WORD PREDICTION TECHNIQUES FOR USER ADAPTATION AND SPARSE DATA MITIGATION

by

Keith Trnka

A dissertation submitted to the Faculty of the University of Delaware in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Computer and Information Sciences

Fall 2010

WORD PREDICTION TECHNIQUES FOR USER ADAPTATION AND SPARSE DATA MITIGATION

by

Keith Trnka

Approved: _____

Errol Lloyd, Ph.D.
Chair of the Department of Computer and Information Sciences

Approved: _____

Michael Chajes, Ph.D.
Dean of the College of Engineering

Approved: _____

Charles G. Riordan, Ph.D.
Vice Provost for Graduate and Professional Education

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: _____

Kathleen F. McCoy, Ph.D.
Professor in charge of dissertation

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: _____

Vijay Shanker, Ph.D.
Member of dissertation committee

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: _____

Sandra Carberry, Ph.D.
Member of dissertation committee

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: _____

Owen Rambow, Ph.D.
Member of dissertation committee

ACKNOWLEDGMENTS

First off, I would like to thank my parents Keith and Dale Trnka for raising me and supporting me all these years. They always encouraged me to learn and experiment as a child, whether it was studying dinosaurs and rocks or trying to see if there is an undiscovered and palatable combination of spices (most such experiments were bitter failures).

Secondly, I am extremely grateful to my dissertation committee for sharing their time and expertise with me. I especially appreciate Kathy's dedication in mentoring me and developing my writing and presentation skills. She has given me a substantial amount of her time and worked along with me to improve my research. I would also like to thank Dr. Rambow for agreeing to be a part of my committee when I can only repay him with baked goods.

I'd like to take the time to acknowledge many of the excellent educators that have inspired me to enjoy learning and experimenting. Even in high school, Dr. Roche (biology) taught me to comprehend detailed writing by forcing me to summarize complex material. Ms. Lyons (chemistry) introduced me to the delightfully systematic and mathematic nature of experimentation. Mrs. McMahon (history) taught me to analyze information more critically by example.

The College of New Jersey also had many excellent professors and instructors. I appreciate the guidance of Dr. Miroslav Martinovic in introducing me to human language technologies, specifically important concepts like TF-IDF, stemming, and question answering. I'm very grateful for the challenging teaching of Dr. Gopalan Sampath (graphics, architecture), who pushed me to expand my limits not just in

programming but also conceptually. Finally, I am truly indebted to Mr. Shawn Sivy for an informative semester of Perl, Apache, and MySQL — his classes were incredibly useful, both in professional work as well as my thesis.

The University of Delaware has an abundance of excellent faculty, and I'm thankful for all of my professors. Specifically, I would like to thank Dr. Vijay Shanker for his contagious enthusiasm for data-driven methods and insightful questions in my talks. I'm also very grateful to Dr. Sandra Carberry for teaching me to comprehend articles quickly and thoroughly, and also for introducing me to several fundamental articles that I otherwise may have overlooked. I would also like to especially thank Dr. Lori Pollock for allowing me to be a part of her excellent research lab.

From my research group, I would like to especially thank Chris Pennington and John McCaw, who helped build our collection of corpora and filled in many gaps in my knowledge, from AAC background to knowledge of relevant work to planning conference trips. I would like to thank several of my former labmates, notably Dr. Ben Breech, Dr. Dave Shepherd, and Dr. Emily Hill. I thank Ben Breech for knowing basically everything that Google couldn't help me with. Dave Shepherd helped me study through prelims and demonstrated the effectiveness of focusing on results and progress. I have to thank Emily Hill for being a wonderful friend all these years and teaching me quite a lot about research in academia.

Finally, I have to thank any of my readers or reviewers, and especially those who have provided me with feedback on papers or presentations over the years. I'd like to thank Randall Munroe for allowing us to use xkcd.com comics. I would also like to thank US Department of Education grant H113G040051 for partially supporting me.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

The field of Augmentative and Alternative Communication (AAC) seeks to help minimize the effects of speech disorders to allow better communication. High-tech AAC devices are electronic devices that address the dual issue of speech impairment and reduced motor control. Communication rate can be improved substantially using word prediction, an application of language modeling to text entry. Word prediction relies on the letters and words that the user has entered so far to suggest likely words that the user is in the process of typing.

Although we can increase language model quality with more training data, these efforts are unlikely to increase the average relevance of training texts. Instead, many documents will be dissimilar to testing data. Even if a few relevant texts are added to the training data, their contribution is marginalized by the abundance of irrelevant texts. We address the problem of varying relevance through adaptive language modeling, specifically topic adaptation and style adaptation. We found that these adaptations increase keystroke savings for both topic and style adaptation individually, and also when topic and style modeling are combined. We have addressed the problem of irrelevant training data with cache models to learn new words and model lexical repetition, and we also integrated a large word list using methods developed for the part of speech ngram model.

We have focused on general-purpose improvements in language modeling and natural language processing. Many of our methods may be applicable to related language modeling problems. However, we have focused only on language modeling improvements which we feel are well-suited for word prediction and AAC.

Chapter 1

INTRODUCTION

You don't use science to show that you're right, you use science to become right.

Randall Munroe, xkcd.com/701

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 (Beukelman and Mirenda, 1998). This affects individuals in all aspects of their lives, from education to the workplace to 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 in order 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 ECO-2, Dynavox's V, and Saltillo's ChatPC are examples of available electronic AAC devices.

Although communication rates differ, 10–15 words per minute (wpm) seems to be a rough upper bound for communicating by selecting letter-by-letter on a keyboard (Copestake, 1997, Newell et al., 1998). Researchers have attempted to develop systems to speed communication rate, oftentimes including word prediction. For example, Newell et al. (1992) surveyed numerous AAC users with the PAL word prediction system and found a wide range of communication rates. One PAL user with cerebral palsy was able to achieve 16 wpm, but two other users with athetoid cerebral palsy produced 1.94–3.64 wpm while two others ranged 0.52–1.16 wpm. Studies with the FASTY system showed similar rates — 1.71–2.87 wpm was achieved in early studies (Beck et al., 2004). A study comparing the EdgeWrite system to the WiViK onscreen keyboard showed a participant with a spinal cord injury produce 11.82 wpm with WiViK and 12.09 wpm with EdgeWrite (Wobbrock and Myers, 2006).[1] Thus communication rates can be extremely slow for this population of users. However, the communication rate of slower users sometimes doubled with the addition of word prediction.

In contrast to these slow rates of communication using AAC, spoken communication produces 130–200 words per minute or more. 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 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

[1] Communication rates reported in characters per minute were converted to words per minute under the assumption of 4 letters plus a space per word, which is consistent with our user study.

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 1.1. 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 ECO-2, Dynavox' V, and Saltillo's ChatPC. The list of predicted words is generated using statistical language

Figure 1.1: Example touchscreen word prediction system. Screenshot taken from our user study using the WordQ onscreen keyboard.

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. Additionally, some systems allow the user to select the specific vocabulary set, for example a vocabulary set for cooking or for a school subject. However, these dictionaries act must be switched manually and only act as filters on the vocabulary,

rather than preferences. For a survey of natural language processing in AAC, see Newell et al. (1998). For an excellent survey of word prediction in AAC, see Garay-Vitoria and Abascal (2004, 2006).

At best, modern devices utilize a trigram model without a topic-specific dictionary and basic recency promotion. However, these models only partially account for the constraints a human would use (Lesher et al., 2002, Brill et al., 1998). The use of a basic language model results in many predictions that are not appropriate for the current conversation, which can be distracting to the user. For example, suppose a user is typing "vowel" in "I would like to buy a vowel" in Figure 1.2. One can imagine that the user seeking to enter the word "vowel" might become very distracted by looking at the offered predictions. Predictions such as "vacation", "voice", and "volunteer" are likely to be somewhat jarring. 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. (2007) 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.

buy a		buy a v		buy a vo	
lot	(F1)	very	(F1)	voice	(F1)
little	(F2)	variety	(F2)	voluntary	(F2)
good	(F3)	vacation	(F3)	volunteer	(F3)
couple	(F4)	van	(F4)	vote	(F4)
big	(F5)	video	(F5)	volume	(F5)
(a) No letters		(b) 'v' entered		(c) 'vo' entered	

Figure 1.2: Inappropriate predictions while typing "vowel". Predictions generated from a trigram model with backoff trained on Switchboard.

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. (1999) demonstrate that bigram and trigram models are not saturated[2] even when trained on 3 million words, while a unigram model is nearly saturated.

For some excellent background reading on language modeling, see Jelinek (1991), Brill et al. (1998), Rosenfeld (2000), and Goodman (2001b).

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 (Rosenfeld, 2000). 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 (Trnka and McCoy, 2007). Wandmacher and Antoine (2006)

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

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.

Corpora of spoken English are much more readily available than AAC corpora and are a reasonable approximation of conversational AAC. However, spoken English corpora 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 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 lack of appropriate corpora affects us in two different ways. Firstly, if we had a substantial amount of appropriate text, we could split the texts into training and testing and develop the algorithms like normal NLP systems. Secondly, if we even had a smaller amount of appropriate text, it could be used for testing (but not training). This would at least provide an evaluation of the system that closely matches real-world usage.

The problem of corpora is that similarity and availability are inversely related, illustrated in Figure 1.3. At one extreme, a very large amount of formal written English is available to use, however, it is very dissimilar from conversational AAC

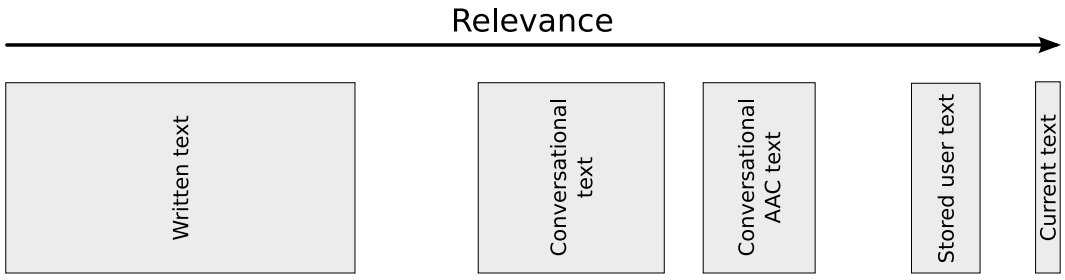


Figure 1.3: The most relevant text available is often the smallest, while the largest corpora are often the least relevant for AAC word prediction.

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 in order 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 Thesis Organization

Chapter 2 lays the groundwork for the rest of the thesis by developing rigorous evaluation methods. We first present our work in evaluation metrics, then present our cross-domain evaluation methodology.

Chapter 3 adapts a language model to the topic of discourse. The chapter focuses on our final topic model and more recent work. For a full treatment of the development of our topic model, refer to (Trnka, 2008c) or Appendix B.

Chapter 4 applies our experiences with topic modeling to adapt to the style of discourse. The style-adapted model builds on top of a part-of-speech (POS) based language model. This chapter discusses both the development of the baseline POS model as well as style adaptation. We present and evaluate many of the alternatives in style adaptation.

Chapter 5 presents two language modeling improvements that are enabled by a part-of-speech-based language model. We first use part-of-speech information to develop a cache model that balances accuracy and coverage of predictions. Then we present a way to more accurately leverage a large word list as a final step in backoff.

Finally, Chapter 6 will present our conclusions and future work.

Chapter 2

EVALUATION METHODS

xkcd.com/309

Evaluation methods guide the process of research in language modeling and other data-driven methods. Research with data-driven methods can be likened to optimization — the researcher experiments with various methods and chooses the ones that maximize the evaluation metrics. In this way, our evaluation decisions are defining the high-level research goals of word prediction. Therefore, we have

designed evaluation methods that we feel approximates real-world usage as closely as possible.

We will describe three different methods of keeping our evaluations focused on users. Firstly, we will use a variety of corpora to try to better approximate the diverse nature of real-world usage in Section 2.1. In Section 2.2, we will examine three common evaluation metrics in detail, focusing on keystroke savings and how to compute it in a way that closely matches real-world benefit. Finally, we will describe methods to vary the training and testing data to approximate situations where the training data is not necessarily appropriate for the user in Section 2.3. We will summarize the decisions made in Section 2.4.

2.1 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. Also, some methods may have varying benefit — for example cache models are more beneficial with longer documents or when the training data is sparse.

Conversational language is arguably the most important use for an AAC device, so 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. However, AAC devices can be used for writing as well, so 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 between written and spoken text. Each of the corpora came with their own formatting conventions: most were in all lower case (except for proper names) but some 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.

The first word of each sentence was converted to lowercase unless it was a known named entity, which would remain capitalized. "I" and contractions such as "I'll" were capitalized in all corpora. 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. We will refer to this as *corpus normalization* — preprocessing corpora such that they all have similar formatting. Corpus normalization is especially important when training and testing on different corpora to ensure that the lexicons of the two corpora match.[1] For further study, see Mikeev (2000) for capitalization and abbreviation disambiguation with a focus on to sentence boundaries. Also, Schwarm and Ostendorf (2002) address normalization in the context of language modeling for meeting transcription — they focus on words that lack a standard transcription, such as numbers and acronyms.

The word and document counts of each corpus are shown in Table 2.1 — the overall collection is roughly half spoken and half written. The average document length (in words) is also shown.

2.1.1 Conversational Speech Transcriptions

The primary target of research in word prediction is conversational usage of AAC devices. The real-time nature of spoken conversations creates a communication rate divide which word prediction attempts to lessen. However, AAC devices are used for a wide variety of conversation.

Ideally, we would like to evaluate word prediction with conversational AAC text, but such a corpus has been unavailable so far. Instead, we will evaluate word

[1] We addressed the problem in a simple manner, but normalization involves many unsolved questions — are "shoulda" and "should've" the same words? (Switchboard sometimes transcribes different forms of the same word differently)

Corpus	Medium	Word count	Documents	Avg. length
AAC Email	email	27,710	117	236
Callhome	spoken	48,407	24	2,016
Charlotte	spoken	187,587	93	2,017
SBCSAE	spoken	237,191	60	3,953
Micase	spoken	545,411	50	10,908
Switchboard	spoken	2,883,774	2,438	1,183
Total spoken	spoken	3,902,380		
Slate	written	4,178,543	4,531	922

Table 2.1: Word and document counts for each corpus

prediction on several conversational speech texts that have speech repairs removed in an effort to bring the corpora closer to what an AAC user would type.

2.1.1.1 Speech Repair Removal

Transcribed conversational text is characterized by frequent speech repairs. Shriberg (1996) suggests that many speech repairs are the result of "getting ahead of oneself". However, AAC communication rate is limited more by the speed of typing words than by the speed of planning a message. For this reason, speech repairs were automatically removed when they could be easily identified. We follow the work of Hindle (1983) in processing simple speech repairs such as backchannels (e.g., uh, um), repetitions, and limited cases of word corrections.

Hindle (1983) describes a method for parsing speech repairs based on the idea of a speech repair consisting of three parts — the reparandum (the part being replaced), the editing signal, and the repair (the replacement). We used pauses, abandonment of words, and backchannels as candidate editing signals, 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

ellipsis marks to signal pauses. Potential editing signals were identified in each sentence and then lexical pattern matching was performed on the words to the left and right to identify speech repairs. In the case of abandoned words (i.e., words which were interrupted by the editing signal), we only require prefix matching. 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. In contrast, 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. We feel that the resulting text is far easier to read and is closer to what we think an AAC user would have said.

2.1.1.2 Switchboard

The Switchboard corpus is a collection of 2,438 English phone conversations recorded by Texas Instruments using a variety of speakers and topics (Godfrey et al., 1992).[2] 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 somewhat unrealistic as a model of unprompted day-to-day conversations, we feel that the large size of Switchboard outweighs any small dissimilarities.

[2] For additional information on data collection and transcription conventions, see http://www.ldc.upenn.edu/Catalog/readme_files/switchboard.readme.html and http://www.isip.piconepress.com/projects/switchboard/

2.1.1.3 SBCSAE

The Santa Barbara Corpus of Spoken American English (SBCSAE) 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 (Du Bois and Englebretson, 200 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.

2.1.1.4 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) (Reppen et al., 2005). Special processing was added for parentheticals and quotations to focus the ngram model on the proper conditioning information. This is especially important when trained on corpora without parentheticals but tested on corpora containing parentheticals. 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.

[3] For additional information on SBCSAE transcription, see http://projects.ldc.upenn.edu/SBCSAE/

2.1.1.5 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.

2.1.1.6 Charlotte

The Charlotte Narrative and Conversation Collection 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.

2.1.2 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 known 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 as was done in Micase.

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 over 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 (e.g., removing titles) to make articles in Slate more natural.

2.1.4 Corpus Statistics

In general, language modeling results will be affected by the amount of training data and the similarity of the training and testing data. This section will quantify two of the problems in word prediction — proper nouns and out-of-vocabulary words (OOVs). Non-cache language models fail to predict OOVs, and therefore keystroke savings is typically lower in corpora with more sparse vocabulary (Trnka and McCoy, 2007). Proper nouns often appear as OOVs, but additionally the (conditional) probabilities of proper nouns from the training data does not often match the testing data.

Proper nouns are problematic in word prediction because the training data used to build a system is unlikely to contain proper nouns typed by the user, not just because the training data may come from a different domain, but also because the training data is likely to be out-of-date for current events. The percentage of proper nouns in each corpus is shown in Table 2.2. The major trend is that more written forms of communication (i.e., Slate, AAC Email) tend to have more proper nouns. Slate in particular has many proper nouns, likely because it discussed current events, which are often centered about a named entity. The AAC Emails tend to have more named entities than the spoken corpora, likely due to a similar trend regarding current events. Switchboard has relatively few proper nouns, which we

feel is due to the topic-prompted nature of the conversations, such as discussing care for the elderly or gardening.

Corpus	Proper Nouns
AAC Email	8.92%
Callhome	8.23%
Charlotte	6.59%
SBCSAE	5.67%
Micase	3.12%
Switchboard	2.10%
Slate	12.03%

Table 2.2: Percentage of proper nouns across corpora.

Out-of-vocabulary words (OOVs) are very difficult to predict, as standard language models are limited to predicting the training set's vocabulary. They are also more difficult to objectively measure with respect to a corpus because they measure the relationship between training and testing data. We measured OOV percentages in two ways: using a large corpus of about a trillion words of web data from Google (Brants and Franz, 2007) for training and also using cross-validation within the same corpus. We used 11-fold cross-validation and the sets were determined randomly, but each set was balanced to contain a similar number of documents and words. The cross-validation sets on Switchboard were additionally balanced based on the distribution of topics. The OOV statistics are shown in Table 2.3.

The OOV percents with respect to web data show an expected trend — formal corpora such as Micase and Slate tend to use less common vocabulary. Otherwise, the words in the other corpora predominantly occur in common web data. In contrast, the cross-validation results are predominantly much higher, due to the effect of training data size on vocabulary tests. The exception is Slate, which has a lower OOV percent with cross-validation as opposed to web data. We believe this trend is explained by the high amount of proper nouns in Slate in the context of

Corpus	OOVs (Google)	OOVs (cross-val)
AAC Email	0.81%	8.48%
Callhome	0.38%	6.86%
Charlotte	0.37%	4.49%
SBCSAE	0.77%	5.76%
Micase	1.35%	4.40%
Switchboard	0.22%	0.52%
Slate	2.12%	1.96%

Table 2.3: OOVs across corpora. Vocabulary is computed in two ways – with respect to a large dictionary (Google corpus) and also using cross-validation

current events. Switchboard shows a particularly low amount of OOVs in the cross-validation test due to the topic-prompted nature of the corpus and also because we balanced the cross-validation sets for topic.

2.2 Evaluation Metrics

The choice of evaluation metrics reflects variation in how we can optimize word prediction. Keystroke savings is the most common evaluation in word prediction, and evaluation with keystroke savings leads to systems that focus on reducing the number of keystrokes. Word prediction systems can also be evaluated in terms of communication rate with real users (the number of words produced per minute). In contrast to keystroke savings, evaluations of communication rate can lead to focused optimization of the user interface and reduction of cognitive overhead. The third evaluation metric which we discuss is perplexity, which leads to designing language models that maximize the probability of generating the testing data.

Intuitively it seems that a language model which increases the probability of the testing data will also reduce the number of keystrokes for word prediction and therefore increase communication rate. However, perplexity and keystroke savings

do not always correlate well (Wandmacher and Antoine, 2007). Additionally, communication rate depends on not just the language model, but also the user and user interface. Studies of communication rate can be further complicated by issues in experimental design.

We will discuss the three evaluation metrics in detail, and our evaluations will focus on keystroke savings, but will also include some perplexity figures.

2.2.1 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:

$$KS = \frac{keystrokes_{\text{normal}} - keystrokes_{\text{with prediction}}}{keystrokes_{\text{normal}}} * 100\% \tag{2.1}$$

Typical values for keystroke savings with reasonable language models range from 45–60%. However, the above equation leaves two questions unanswered: What is a keystroke? How should the number of keystrokes under each entry method be computed?

2.2.1.1 What Counts as a Keystroke?

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. Although most characters only take one keystroke, some such as uppercase letters typically take two keystrokes (depending on use of shift vs caps lock). We follow the current trend that an uppercase letter is counted as a single keystroke. We expect this to affect the results very little; our preprocessing removed most uppercase letters at the start of a sentence and the only remaining uppercase words are

proper nouns, which comprise a relatively small percent of the data, especially for conversational language (see Table 2.2).

The keystrokes used for our evaluation may differ from the keystrokes originally used to enter the text, because we preprocessed the corpora to conform to AAC practices. 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 whether a newline or "speak key" counts towards keystroke savings or not. For the most part, so long as systems being compared either all simulate newlines or all ignore newlines, the keystroke savings measure can still be used for comparison. We are unaware of any prior publications that acknowledge this issue; from personal communication, the field is inconsistent about simulating newlines or not. 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 simulate the typing of a speak key at the end of every sentence. The implication of this decision is that our keystroke savings may appear lower. Because this keystroke is not predicted by our algorithms, the maximum possible keystroke savings given is slightly lower (on the order of 1–2% compared to ignoring a speak keystroke). In contrast to an end-of-sentence key, researchers normally include spaces as measurable keystrokes for word prediction.

2.2.1.2 Simulating the User and Interface

The second major problem in keystroke savings 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 software, 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 and 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". Rather than complicate our assumed interface, we allow a user to undo the prediction selection by pressing backspace; that functionality is implemented in our user study (Trnka et al., 2007, Trnka et al., 2009) as well as in our simulations. In our system, if the exact desired word does not appear in the predictions (i.e., with the desired suffix), the user would need to continue typing letter-by-letter. In addition to the problem of backspacing, research in multi-word prediction faces a similar problem, where the system may select a two-word prediction followed by a one-word prediction when a one-word prediction followed by a two-word prediction may have been fewer keystrokes. These two examples (backspace-editing and multi-word prediction) illustrate some of the potential complications involved in computing keystroke savings. Multi-word prediction is not considered in our studies.

Another aspect of the user interface that affects keystroke savings is the number of predictions offered in the prediction window. In a theoretical simulation, keystroke savings increases with the window size. However, larger window sizes require more perceptual and cognitive overhead in a real system. Researchers typically evaluate word prediction systems with 5–7 predictions. We will evaluate the system with five predictions in the list. However, it should be noted that changes in the language model typically affect all window sizes similarly.

2.2.1.3 Upper Bounds

In trying to improve the state of word prediction, several researchers have noted that it seems extremely difficult to improve keystroke savings beyond a certain point. Copestake (1997) discussed the entropy of English to conclude that 50–60% keystroke savings may be the most we can expect in practice. Lesher et al. (2002) replaced the language model in a word prediction system with a human to try and estimate the limit of keystroke savings. The participants were offered a large list of predictions from their advanced language model and filtered the list. They found that human-filtered predictions could achieve 59% keystroke savings compared to their advanced language model, which achieved 54% keystroke savings. They noted that one subject's filtering achieved nearly 70% keystroke savings on one particular text, and concluded that further improvements on current methods are possible. Garay-Vitoria and Abascal (2006) survey many prediction systems, showing a wide spectrum of savings, but no system offers more than 70% keystroke savings.

We investigated the problem of the limitations of keystroke savings first from a theoretical perspective, seeking a clearly defined upper boundary (Trnka and McCoy, 2008). Keystroke savings can never reach 100% — it would mean that the system divined the entire text they intended without a single key. In a single-word prediction system[4], the minimum input for a perfect system is one keystroke per word (to select the prediction) and one keystroke per sentence. We compute the keystroke savings at this minimal amount of input, which we call the *theoretical keystroke savings limit*. The theoretical keystroke savings limit is essentially a derivative of average word and sentence length for a corpus, but in the units of keystroke savings.

Several examples of the theoretical limit are shown in Table 2.4. In this evaluation, a trigram model was individually trained and tested on each corpus, using

[4] This assumption is reasonable because prediction/completion of multiple words is currently very uncommon in AAC.

cross-validation for all corpora but Switchboard and Slate which were split into a single training and testing set. In contrast to the actual trigram model performance,

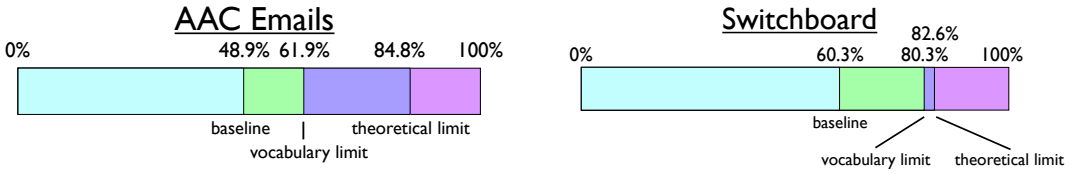
Corpus	**Trigram**	**Vocab. limit**	**Theor. limit**
AAC Email	48.92%	61.94%	84.83%
Callhome	43.76%	54.62%	81.38%
Charlotte	48.30%	65.69%	83.74%
SBCSAE	42.30%	60.81%	79.86%
Micase	49.00%	69.18%	84.08%
Switchboard	60.35%	80.33%	82.57%
Slate	53.13%	81.61%	85.88%

Table 2.4: Vocabulary limit and theoretical limit of keystroke savings. The keystroke savings using a trigram model is shown for reference.

the theoretical limits all fall within a relatively narrow range, suggesting that the achievable keystroke savings may be similar even across different domains.

We can derive a more practical limit by simulating word prediction using a perfect model of all words that occur in the training data. This gold standard will predict the correct word immediately so long as it occurs in the training corpus, but will require full keystrokes for out of vocabulary words (OOVs). We call this measure the *vocabulary limit* or *practical limit* and apply it to evaluate whether the difference between training and testing vocabulary is significant. Although this is similar to measuring OOVs, the vocabulary limit frames the measurement in terms of the impact on keystroke savings.

The vocabulary limit exhibits much greater variation, as it reflects not just the theoretical limit but also vocabulary differences (which are partly dependent on corpus size). Although this measure is an analogue of OOVs, it more concretely illustrates the effect of OOVs on actual keystroke savings — 60% keystroke savings when training and testing on AAC Email would be extraordinary, but less surprising on Switchboard.



(a) AAC Emails (b) Switchboard

Figure 2.1: The relative impact of the limits of keystroke savings compared to baseline models for AAC Emails and Switchboard.

In particular, the limits can help guide research. For example, Figure 2.1 illustrates the limits of Table 2.4. If the actual keystroke savings is close to the theoretical limit, then methods that increase the theoretical limit are likely to be beneficial (e.g., multi-word prediction). If the difference between the vocabulary and theoretical limits is large such as with AAC Emails, then methods to expand the prediction vocabulary may be very beneficial, such as cache modeling (see Chapter 5). And if the difference between actual keystroke savings and the vocabulary limit is high such as with Switchboard, then language models that make better use of the training data may be beneficial, such as topic and style adaptation (Chapters 3 and 4, respectively).

2.2.1.4 Core and Fringe Vocabulary

AAC devices such as PRC's Pathfinder make a distinction between core and fringe vocabulary. Core vocabulary is (loosely) a small set of common words that can be used to communicate. Devices such as the Pathfinder optimize the user interface for the core vocabulary. Specifically, the Pathfinder uses Minspeak (Baker, 1982) to encode the core vocabulary of about 300 words, which generates a word via a short (1-3) sequence of icon selections. Other methods are possible, but many traditional devices are specifically optimized for a small number of words.[5]

[5] This is partly due to the small vocabulary of some of the user base.

Fringe words comprise the remainder of the vocabulary. Devices such as the Pathfinder provide letter-by-letter typing for fringe words and some form of word prediction to speed up the process. We originally focused our evaluation solely on fringe words due to the popularity of the Pathfinder interface, and also due to our focus on topic adaptation (Trnka, 2008c). However, we expect methods such as style adaptation to benefit both common and uncommon words. Additionally, pure typing interfaces with word prediction are becoming more common for AAC. Therefore, most of our evaluations will evaluate using all words, not just fringe words.

For a more detailed discussion of core vocabulary, see (Stubbs, 1986).

2.2.1.5 Summary

Theoretical keystroke savings is the primary evaluation for word prediction systems. We compute keystroke savings as the percentage difference between the number of keystrokes required to type a text letter-by-letter versus with the word prediction system, under the assumption that the predictions are fully utilized.

2.2.2 User Evaluation: Typing Rate

Although word prediction saves keystrokes compared to letter-by-letter entry, some research has questioned whether keystroke savings translates to an increase in communication rate (Venkatagiri, 1993, Koester and Levine, 1994). Such work points to the added cognitive tasks that are required to use the technique. For instance, in order to fully utilize word prediction, after each letter has been entered the user must shift their gaze to the list of predicted words, scan the list, and for each word in the list, decide whether or not it is the word they intend, and act on that decision accordingly by either continuing the scan or selecting the word.

We performed a user study with 33 participants and found that word prediction can substantially increase communication rate despite an increase in perceptual/cognitive overhead, especially with higher theoretical keystroke savings (Trnka et al., 2007, Trnka et al., 2008, Trnka et al., 2009). We tested both a trigram model predictor as well as a basic dictionary-based predictor, both including a unigram cache model. We found that users produced text 61.2% faster and saved 52.6% keystrokes using the trigram model, and users produced text 10% faster and saved 20.9% keystrokes using the more basic predictor. Users tended to trust the trigram predictions more, and therefore achieved more of the potential benefit of the system — 94.4% for the trigram model compared to 79.9% for the basic model. Also, users took longer per key press as they relied on word prediction more.

Communication rate relates to simulated keystroke savings in a complex way. Firstly, the actual keystroke savings that users achieve is typically lower than the potential keystroke savings that the system offers, because users sometimes don't look at the predictions or miss the desired word when it's predicted. Furthermore, as potential keystroke savings increases, users' actual keystroke savings is a higher percentage of the potential — users trust the system more and check the prediction list more often and more carefully. Secondly, the benefit of word prediction on communication rate is dependent on both the actual keystroke savings and the cognitive overhead of looking at the predictions:

$$speedup \approx \frac{1}{(1 - \frac{aks}{100}) * \frac{spk_{new}}{spk_{old}}}$$

where aks is the actual keystroke savings of the user, and spk_{new}/spk_{old} is the seconds per keystroke of predictive text entry relative to normal text entry. For example, the users took 2.92s to press a key with the trigram model compared to 2.23 seconds for letter-by-letter entry, an overhead of 30.9%. However, users saved 52.6% keystrokes using the trigram model and the keystroke savings outweighed the additional cognitive overhead for a speedup of 61.2%. For more details on

communication rate in word prediction and the relationship to keystroke savings, please see (Trnka et al., 2009).[6]

2.2.3 Perplexity

Perplexity is an automatic method of evaluating language model quality, like keystroke savings. Unlike keystroke savings, perplexity is a pure evaluation of the language model itself — there are no settings regarding the number of predictions or how a user might type using word prediction. Perplexity is very common in evaluation of language modeling for speech recognition and also in pure language modeling research (i.e., language modeling without an intended application) (Rosenfeld, 2000, Goodman, 2001a).

Perplexity can be formulated in two equivalent ways. The intuitive way of formulating perplexity is the inverse of the geometric mean probability of each word in the testing data:

$$PP(w_1, \ldots, w_N) = \frac{1}{\sqrt[N]{\prod_i P(w_i \mid h)}} \tag{2.2}$$

where $w_1, \ldots, w_N$ is the sequence of words in the testing data and $P(w_i \mid h)$ is the probability of word i using the language model being tested. The conditioning information h used in the language model can be any representation of the words that have been used so far. However, the product $\prod_i P(w_i \mid h)$ is too small of a number to be computed on standard hardware. Therefore the second representation of perplexity is more common.

The second way of representing perplexity uses log-space to perform the computation and avoid the problem of small numbers:

$$PP(w_1, \ldots, w_N) = 2^{-\frac{1}{N} * \sum_i log_2 P(w_i|h)} \tag{2.3}$$

[6] In this work, the trigram-based predictor is labeled "advanced" and the dictionary-based predictor is labeled "basic".

This formulation has relation to another metric — $-\frac{1}{N} * \sum_i log_2 P(w_i \mid h)$ is the cross-entropy of the testing data with respect to the language model. Although both methods of computing perplexity will theoretically produce the same number, the latter equation is favored due to computational requirements.

Perplexity can range from 1 (each word has probability 1) to infinitely high (when each word in the corpus has minimal or zero probability). Typical values for perplexity range from 100–1000, where lower numbers are better.

One of the significant advantages of perplexity is computational performance compared to domain-specific evaluations. In speech recognition, perplexity is substantially faster to compute than word error rate. Perplexity is also much faster to compute than keystroke savings — keystroke savings requires that the whole vocabulary be partially sorted for every word processed in testing data, which in turn requires probability computation for each word. On the other hand, perplexity only computes probability once for each word in testing data.

Despite the computational advantage of perplexity, it does not correlate well with domain-specific evaluations. In speech recognition, perplexity has been shown to often differ somewhat from word error rate (Rosenfeld, 2000, Goodman, 2001a). Rosenfeld (2000) gives the following rule-of-thumb:

> As a rough rule of thumb, reduction of 5% in perplexity is usually not practically significant; a 10%–20% reduction is noteworthy, and usually (but not always) translates into some improvement in application performance; a perplexity improvement of 30% or more over a good baseline is quite significant (and rare!).

In word prediction, perplexity has also been shown to not correlate well with keystroke savings (Wandmacher and Antoine, 2007).

Although the mismatch between perplexity and domain-specific evaluations is not fully understood, we will provide two examples in word prediction. Firstly, perplexity is impacted by the probability assigned to unknown words (and is one of

the motivating factors for smoothing methods).[7] However, unknown words can't be predicted, so a method can decrease perplexity on unknown words without having an impact on keystroke savings. Secondly, some common words are perfectly predicted (i.e., they are predicted with no letters typed). A change in language modeling can potentially increase the probability of these words at the expense of others and not improve keystroke savings (or even decrease it!).

Despite the flaws with perplexity, it is used extensively in speech recognition and pure language modeling research. Therefore, we will include perplexity as an evaluation metric in some of our evaluations.

2.2.4 Statistical Significance

When evaluating our methods, we not only want to know the average improvement (or degradation), but also whether the change can be explained by random chance or whether it is likely to be a real benefit. In our larger tests, we address this by computing statistical significance using a paired t-test on keystroke savings.

A paired test works by computing before and after metrics on the same data, then taking the difference at the level of the data. For our work, we record keystroke savings on a document level and pair the documents, measuring the change in keystroke savings for each document. Although it may be possible to pair keystroke savings at the word level, this violates the assumption that the data points are independent of each other. Rather than compute the absolute difference in keystroke savings, we compute the percentage change. This follows the intuition that some documents are hard or easy, and a change to the algorithm would probably have a more consistent percentage change rather than absolute change across documents.

[7] Furthermore, the probability of an unknown word depends on estimation of the number of unknown words, which is somewhat arbitrary — estimating that there are twice as many zero-frequency words will lower perplexity without affecting keystroke savings.

This percentage change is the same as the keystroke savings of the improved method with respect to the baseline, rather than computing keystroke savings with respect to letter-by-letter typing and taking a difference.

The mean of the paired distribution is then compared to zero (the null hypothesis) using a t-test. When the results are sigificant, we present the most informative p-values available to us, generated from a table mapping t-values to p-values. In general, most authors will attribute significance when $p \leq 0.05$ (i.e., there is less than 5% chance that the data could be from a distribution with a mean of zero).

The method we have chosen for statistical significance is generally sound, but produces varying findings depending on the corpus. This method is highly dependent on the number of documents in the pairing. Even minor changes in the algorithm often produce significance on corpora with many documents (i.e., Switchboard, Slate) while more drastic changes may not yield significance on corpora with few documents (e.g., Callhome). For example, even though the AAC Email corpus has fewer words than Callhome, due to the larger number of documents, significance is much easier to achieve. Additionally, statistical significance is highly dependent on the variation in the data. If a method (on average) yields a benefit, but not consistently, then the results are unlikely to be significant. In contrast, in a paired test significance can be trivial when the changes are all positive or all negative. In the context of language modeling, this means that changes to the language model which are unlikely to ever reduce performance often produce trivially significant results, whereas changes that sometimes help and sometimes hurt are more difficult to find significant.

2.3 Domain-varied Evaluation

The overall goal of our evaluation is to determine how well word prediction will benefit users. Users communicate in a variety of domains and genres, therefore we evaluate word prediction with a variety of corpora. Not only does this help

protect against over-fitting any one type of corpus, but also we can better estimate how well our system will work for a given user. For example, a user that types a significant number of emails may experience word prediction quality more similar to email or other written texts. An individual that primarily uses their AAC device for conversation would be more likely to see trends similar to the spoken corpora. Wandmacher (2009) also advocate domain-varied evaluation to address the heterogeneous needs of AAC systems.

In addition to the issue of multiple domains/genres, in practice, the language model in an AAC device has been trained on texts very different than the texts it is used to predict. To get a better sense of how our methods will affect actual users, we use three tests 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 language models, 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. Out-of-domain training shows the expected degradation (or improvement) of performance resulting from using word prediction in a more 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 an advanced language model 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.

Unfortunately, the domain-varied evaluation leads to complex tables (e.g., Table 2.7 on page 35). Each row of the table represents one *testing corpus*, while each column represents the relationship of the training data to the testing corpus (row). In general, we will avoid a full-scale evaluation while presenting the development of the language models, and will focus on the evaluations that are most appropriate for the techniques in question.

We evaluated a baseline trigram model with the three domain variations to estimate the benefits of word prediction on various genres and also to evaluate the importance of in-domain data for various genres. This evaluation expands on our previous work (Trnka and McCoy, 2007) by adding perplexity to the evaluation. We evaluated the baseline model using in-domain evaluation first, shown in Table 2.5. 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 (see Table 2.3 on page 18). 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, other corpora have less data per speaker than the AAC corpus. 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 previously balanced the cross-validation sets to have comparable topic distributions.[8] This reduces the chances of encountering a word

[8] This balancing was performed to provide a more reliable evaluation of topic adaptation methods.

in testing that did not occur in any training texts.

Perplexity only correlates partially with keystroke savings — the lowest perplexity and highest keystroke savings occur on Switchboard. However, Slate shows the highest perplexity, yet it shows the second highest keystroke savings. The higher perplexity on Slate is likely due to the high number of OOVs, which degrades perplexity significantly. Additionally, even the perplexity of known words is higher, due to the large vocabulary of Slate. In general, the results show that perplexity across corpora does not correlate closesly with keystroke savings, although previous studies (Wandmacher and Antoine, 2007) have shown that it correlates reasonably well on the same data with slightly different language models.

	In-domain training	
Corpus	**KS**	**PP**
AAC Email	49.449%	427
Callhome	49.028%	279
Charlotte	52.884%	272
SBCSAE	46.793%	384
Micase	51.607%	428
Switchboard	58.839%	147
Slate	54.332%	839

Table 2.5: In domain evaluation with keystroke savings and perplexity.

We also evaluated the baseline trigram model using out-of-domain training in Table 2.6. 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 data size increase ranged from 166 times more data for Callhome (for 5.92% more savings) to 13 times more for Micase (for 1.08% more savings). In contrast to the smaller corpora, the larger corpora showed better keystroke savings using in-domain training. The results on Slate are

	In-domain		Out-of-domain	
Corpus	**KS**	**PP**	**KS**	**PP**
AAC Email	49.449%	**427**	**51.963%**	653
Callhome	49.028%	279	**54.949%**	**209**
Charlotte	52.884%	272	**55.512%**	**254**
SBCSAE	46.793%	384	**50.366%**	**340**
Micase	51.607%	**428**	**52.684%**	488
Switchboard	**58.839%**	**147**	55.174%	272
Slate	**54.332%**	**839**	43.932%	4,671

Table 2.6: Out-of-domain evaluation with keystroke savings and perplexity. In-domain evaluation is shown for reference; the best results per row are bolded.

somewhat unsurprising, as there is less out-of-domain training data compared to in-domain. In the case of Switchboard, the lower keystroke savings of out-of-domain can be attributed to the very different style and vocabulary of the other corpora, which is dominated by the formal written style of Slate.

The perplexity figures in the out-of-domain test do not correlate with keystroke savings, especially comparisons within each row. For example, on the AAC Email corpus, keystroke savings increases by 2.14% despite a 53% increase in perplexity. The disparity can be attributed to the handling of OOVs as well as words that are correctly predicted without typing any letters. The effect of OOVs is especially dramatic in this case — the much larger amount of training data reduces the probability of each unknown word due to both a smaller amount of held-out probability and a larger estimate of the number of unknown words. Additionally, the words that were already OOV with respect to the AAC Email data are likely to be domain-specific and remain OOV with respect to the other corpora.

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 2.7. As with the out-of-domain test,

the highest keystroke savings for each corpus is shown bolded.

	Training domain					
	In		Out		Mixed	
Corpus	**KS**	**PP**	**KS**	**PP**	**KS**	**PP**
AAC Email	49.449%	**427**	51.963%	653	**54.004%**	495
Callhome	49.028%	279	54.949%	209	**55.004%**	**206**
Charlotte	52.884%	272	55.512%	254	**56.029%**	**232**
SBCSAE	46.793%	384	50.366%	340	**50.695%**	**316**
Micase	51.607%	428	52.684%	488	**53.812%**	**408**
Switchboard	**58.839%**	**147**	55.174%	272	58.608%	150
Slate	**54.332%**	**839**	43.932%	4,671	54.167%	970

Table 2.7: Mixed-domain evaluation with keystroke savings and perplexity. In-domain and out-of-domain evaluations are shown for reference; the best results per row are 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. Daumé (2007) similarly found that a naïve mix of two domains can improve performance over each separate model.

Perplexity with mixed-domain evaluation nicely correlates with the changes in keystroke savings from out-of-domain training. For most corpora, the training data changes very little, because the in-domain portion is a fraction of the size of the out-of-domain portion. The effect could simply be the result of a large reduction in OOVs, which would show a dramatic effect in both keystroke savings and perplexity.

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 and McCoy (2007).

2.4 Conclusions

We address evaluation in three main ways. Firstly, we evaluate using a mixture of corpora, and analyze results individually on each corpus. This allows us to analyze high-level trends in word prediction, such as whether techniques are more beneficial to corpora with very large vocabulary (e.g, Micase, Slate) or smaller vocabulary (e.g., Switchboard). Similarly, it allows us to investigate the impact of word prediction techniques on spoken vs. written texts.

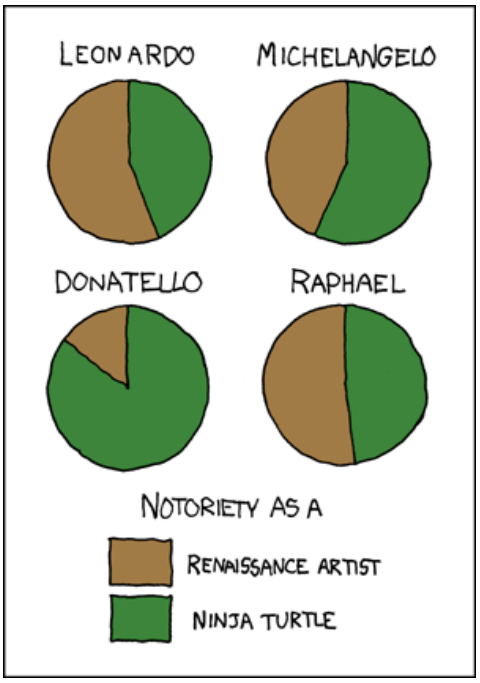
Secondly, we will use keystroke savings as our primary evaluation, and compute keystroke savings in a way that approximates AAC usage. We will also include perplexity evaluations when possible, to relate to work in speech recognition and pure language modeling. However, we note that trends in keystroke savings often differ from trends in perplexity, because changes in probability may not affect keystroke savings, especially with respect to OOVs.

Finally, we will emulate real-world usage by varying the similarity of the training data to actual usage. In contrast to most research, where the training data is held constant, we hold the testing data constant, varying the relationship of the training data to testing from in-domain to mixed-domain to out-of-domain.

We will also include evaluation and analysis that is specific to the individual techniques that we develop. Any technique-specific analysis will be presented along with the technique.

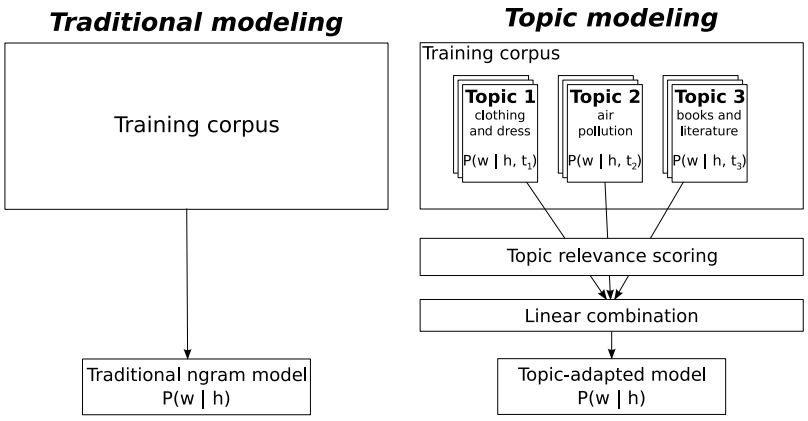
Chapter 3

TOPIC ADAPTATION



xkcd.com/197

Topic-adapted language models have been used to improve both word prediction (Li and Hirst, 2005, Trnka et al., 2006a) and speech recognition (Bellegarda, 2000, Rosenfeld, 1994). Our method of topic adaptation utilizes corpora that are separated by topic, whether annotated by humans (Lesher and Rinkus, 2001), treating each document as a topic (Mahajan et al., 1999), or using automatic clustering to



(a) Traditional ngram model (b) Topic model

Figure 3.1: High-level view of topic modeling and traditional ngram modeling.

define topics (Florian and Yarowsky, 1999). The defining aspect of topic models is that training data is weighted by topical relevance with respect to the current context. In contrast to traditional ngram modeling, the language model is specifically tailored to topics or documents that are similar to the current document. Our approach is illustrated at a high level in Figure 3.1 — first the topics are assigned relevance scores and then the topics are combined using the scores as weights.

We have previously shown that our approach to topic modeling can be successfully applied to word prediction (Trnka, 2008c), both for human-labeled topics and when treating each document as a very specific topic. We will first summarize our previous findings and decisions in Section 3.1.

We then expand on our previous research in three main ways. Firstly, we will re-evaluate our topic models with updated procedures in Section 3.2. The updated evaluation measures perplexity and also includes some evaluation on all words (not just fringe words). Secondly, we will discuss pilot experiments of integrating testing data into the training data in Section 3.3. This is particularly useful when none of the training data is relevant, and helps to customize the model to user data over

time. Finally, we will address the shortcomings of document-as-topic by applying text clustering to produce topics in Section 3.4.

Section 3.5 will compare and contrast our methods and findings with related research. Finally, we will summarize our findings and contributions in Section 3.6.

3.1 Prior Work (Proposal Summary)

The majority of our topic modeling research was completed with the PhD proposal. This section will summarize our previous findings. For further details, see Appendix B.

3.1.1 Basic Structure: Are Unigrams Enough?

Figure 3.1(b) shows our linear combination of topic-specific models, but leaves the history h undefined. In developing a topic-adapted language model, we can set h as an empty history to create a topic-adapted unigram model, or include one or more words of context in h to create a higher order topic-adapted model, such as a trigram model. Certainly a topic-adapted unigram model would not offer predictions appropriate for the context — to build a useful word prediction system, it would need to be combined with a model of context (e.g., unadaptive trigram model). A topic-adapted unigram model reflects the intuition that topic only affects the vocabulary and therefore contextual preferences can be provided outside the adaptation. On the other hand, choosing a topic-adapted trigram model reflects the intuition that topic may be more complex than vocabulary preferences.[1]

To illustrate the difference between the two models, consider the word "bank" in a conversation about savings accounts: In a unigram-like topic adaptation, the adaptation would increase the likelihood of "bank" in all contexts. In a higher-order

[1] The difference between the two models also reflects the difference between domain-specific terminology (which includes multi-word expressions) and vocabulary (which is only a list of words).

ngram adaptation, the adaptation would only increase the likelihood of "bank" in contexts that it belongs in. Furthermore, the higher-order adaptation would only increase the likelihood of "bank" in contexts appropriate for the intended word sense. Additionally, the higher-order ngram approach is more conducive to adaptation of multi-word terms (e.g., "savings account").

We previously explored both methods of topic adaptation (Trnka et al., 2006b). For the unigram adaptation (previously called "hybrid topic modeling"), we adapted a pure unigram model to the topic of conversation. This amounts to having a separate unigram model for each topic and combining these unigram models with topic relevance weights. The resulting topic-adapted unigram model was multiplied with a baseline trigram backoff model. An exponential weight was added to the topic-adapted unigram model and tuned to produce best performance, though tuning resulted in a weight of 0.05. The rough form of the model is shown below.

$$P_{hybrid}(w \mid h) = P_{baseline}(w \mid h) * \left(\sum_{t \in topics} P(t \mid h) * P(w \mid t) \right)^{\alpha} \tag{3.1}$$

where h is an abstract representation of the history, in this case the previous two words for the baseline component (which is also a backoff model) and a bag-of-words representation for $P(t \mid h)$.

For the higher-order ngram model, we trained a separate bigram and unigram model on each topic. The topic-adapted unigram model was constructed by weighting each model with the topic relevance scores, just like the unigram-based topic adaptation. The same procedure was used to compute an overall bigram model. These two models were then combined using backoff. At the time of the previous evaluation, we only used bigrams for the higher-order topic adaptation due to time constraints. The rough form of this model is shown below.

$$P_{topic}(w \mid h) = \sum_{t \in topics} P(t \mid h) * P(w \mid h, t) \tag{3.2}$$

where $P(t \mid h)$ is the same topic relevance score as the previous model and $P(w \mid h, t)$ is a bigram model trained from topic t.

Both versions of topic adaptation displayed an improvement over a baseline trigram model. However, the higher-order topic adaptation yielded greater improvement across prediction list sizes 1–10 (0.8%–1.5%).

3.1.2 Combining Frequencies

In creating an overall language model, we weight each individual topic model and combine them into an overall topic-adapted language model. This process has several options that interact with backoff and smoothing. Although few researchers discuss this problem, we feel that combining frequencies and later performing smoothing is better for smoothing algorithms (i.e., smoothing algorithms have a better sense of how much probability to reserve for unseen events). The rough form of this is shown in Figure 3.2. Adda et al. (1999) also chose to implement topic models this way, albeit for a different task (broadcast news transcription). The decision to combine frequencies rather than probabilities makes our model more complicated to represent mathematically (shown below), but it allows smoothing to work better and allows for better runtime optimization.

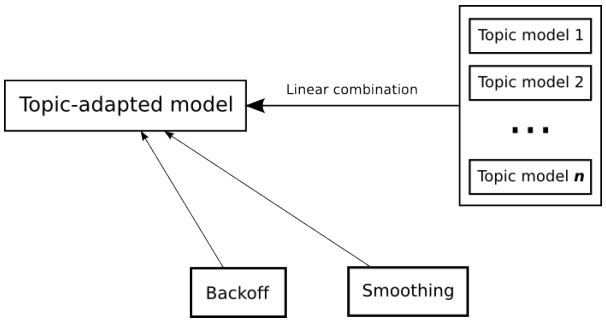


Figure 3.2: Illustration of backoff and smoothing in our system.

$$f(w \mid w_{-1}, w_{-2}) = \sum_{t \in topics} P(t \mid h) * f(w \mid w_{-1}, w_{-2}, t) \tag{3.3}$$

$$f(w \mid w_{-1}) = \sum_{t \in topics} P(t \mid h) * f(w \mid w_{-1}, t) \tag{3.4}$$

$$f(w) = \sum_{t \in topics} P(t \mid h) * f(w \mid t) \tag{3.5}$$

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

$$P'(w \mid w_{-1}, w_{-2}) = \begin{cases} P_{smooth}(w \mid w_{-1}, w_{-2}) & if P(w \mid w_{-1}, w_{-2}) > 0 \\ \alpha_{w_{-1}, w_{-2}} * P_{smooth}(w \mid w_{-1}) & if P(w \mid w_{-1}) > 0 \\ \alpha_{w_{-1}} * \alpha_{w_{-1}, w_{-2}} * P_{smooth}(w) & otherwise \end{cases} \tag{3.6}$$

where $\alpha_{w_{-1},w_{-2}}$ is the held out probability mass from the distribution $P_{smooth}(w \mid w_{-1}, w_{-2})$ and similarly $\alpha_{w_{-1}}$ is the held out probability mass from the bigram distribution.

3.1.3 Smoothing

The choice of combining frequencies and runtime optimization led to some problems with respect to smoothing. Firstly, the interpolation process produces frequency-like real-valued numbers. However, most smoothing methods operate on integers to compute probabilities, for example, counting the number of words that occur once, twice, etc. Therefore we added one to the frequency and took the floor of the resulting number, changing anything in the range 0–0.99 to 1, anything in the range 1-1.99 to 2, etc.

The combination of frequencies using topic relevance weights results in a distribution with strictly less frequency than the original, because the frequencies are being multiplied by weights below one. This creates a problem for smoothing algorithms, which use the frequencies to implicitly determine the sparseness of the

distribution, resulting in overly discounted distributions. We addressed the issue by scaling the results so that the sum of frequencies in the distribution is the same before or after the interpolation. We found that this improved keystroke savings slightly. Another similar approach is to ensure that the sum of the topic relevance scores is equal to the number of topics.

We also had to choose an appropriate smoothing function. We originally wanted to use the smoothing from Katz backoff model (Katz, 1987), but this did not allow us to smooth only the conditional distributions that were necessary for prediction. We experimented with Good-Turing smoothing (Good, 1953) and a variation of Witten-Bell smoothing (Witten, 1991) but settled on our own method. Specifically, we had to deal with the combined problems of 1) only smoothing distributions on-demand (to speed up the system) and 2) dealing with distributions that are too sparse for methods like Good-Turing smoothing. Our smoothing function is shown below:

$$P(w \mid h) = \frac{f(w \mid h)}{f(w \mid h) + \lambda} * \frac{f(w \mid h)}{f(h)} \tag{3.7}$$

where λ is an optionally tunable parameter, which we normally take to be 1. In this smoothing method, a larger percent of the probability is discounted from infrequent words (similar to Good-Turing smoothing). In the worst case (all frequency 1), half of the probability mass is reserved, similar to Witten-Bell smoothing.

3.1.4 Determining Topical 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 will describe variations in identifying the relevant topics. This corresponds to $P(t \mid h)$ in the overall topic model, which we will refer to as a *(topical) relevance score*. The overall approach we take for topical relevance is similarity scoring based on bag-of-words representations of each topic and the current context.

The representation of each topic is a pure unigram model — the weight of each word is proportional to the frequency. The representation of the current context is more complex — the weight of each word is a combination of not just frequency, but also recency and topical salience, discussed in Section 5.1.1. This representation is often called a cache or recency model.[2] After discussing aspects of topical relevance, we will briefly present our findings with different scoring functions in Section 3.1.4.2.

3.1.4.1 Facets of Relevance: Frequency, Recency, and Topical Salience

The most basic aspect of relevance is frequency — a topical word that occurs twice is probably more important than a topical word that occurs once. To illustrate the effects of the other facets, we will use a table format showing the weights assigned to each word in a snippet. In Figure 3.3, "Kathy" has a weight of 2 and other words

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

Figure 3.3: Example cache weights using frequency. The user is currently typing "grad school" in this sentence.

have a weight of 1.

The recency of use of a word also contributes to the importance of the word. If a word was used recently, we would expect that the word is more important to the current topic than less recently used words, due to the shifting nature of the topic of conversation. Similarly, many topical words tend to be reused throughout a discourse. Beeferman et al. (1997) confirmed the intuition that words tend to be reused — they modeled the probability of lexical repetition using an exponential function of the recency of the word. We follow Bellegarda (2000) 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

[2] The naming conventions mimic basic cache models; see Chapter 5.

this method, illustrated in Figure 3.4. 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,

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 3.4: Example cache weights using frequency and exponential decay. The user is currently typing "grad school" in this sentence.

λ is a tunable decay weight. We chose $\lambda = 0.95$ following Bellegarda (2000).

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. 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. However, instead of using Inverse Document Frequency (IDF) to measure topical salience, we use Inverse Topic Frequency (ITF), which specifically weights words that are good discriminators between topics. Another advantage of ITF is that it will be tailored to the particular topic granularity that we use (see Section 3.1.5 for topic granularity). As is common in Information Retrieval, we take the log of the ITF. For the sake of the example, we replaced the ITF scores with approximated IDF scores from Google. In this example, the weight of "Kathy" is 11.189 and the weight of "shared" is 4.256. Although using IDF-like methods should weight function words such as "an" or "the" very low (nearly zero), we prefer to use a list of common words that are independent of topic and assign these words zero weight. 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. We felt that these two additional rules helped protect our scoring against idiosyncrasies in small corpora and when documents are used as topics.

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

Figure 3.5: Example cache weights using frequency, exponential decay, ITF, and stopword elimination. The user is currently typing "grad school" in this sentence.

3.1.4.2 Relevance Functions

We primarily use distributional similarity functions to determine relevance. We convert an arbitrary similarity score sim into the necessary probabilities by normalizing:

$$P(t \mid h) \approx \frac{sim(t, c)}{\sum_{t'} sim(t', c)} \tag{3.8}$$

where t is the topic being scored, h is an abstract representation of the current context, and c is the cache representation of the current context.

We primarily use the cosine measure for similarity scoring, shown below. The cosine function can be thought of as a two-stage method: Firstly, the distributions are normalized to unit vectors, which is similar to converting the frequencies or weights to probabilities. Then the cosine score is the dot-product of the normalized distributions, which ranges from 1 (when all the normalized weights match exactly) to 0 (when there are no common weights between the two distributions).

$$sim_{cosine}(t, c) = \frac{\sum_{w \in t \cap c} f_t(w) * f_c(w)}{\sqrt{\sum_{w \in t} f_t(w)^2} * \sqrt{\sum_{w \in c} f_c(w)^2}} \tag{3.9}$$

where $f_t(w)$ is the frequency of word w in topic t and $f_c(w)$ is the frequency of word w in the cache. In this equation, we view t and c as sets of words from the topic and cache, respectively. This measure takes into account many common aspects of distributional similarity — the amount of overlap in word usage, the amount of non-overlap in word usage (implicitly), and the similarity of the actual weights (see Section B.5.4.1).

Other researchers have found that different similarity scores or classification scores were more useful for their tasks. Lee (1999) found that the Jacquard coefficient was better than cosine in differentiating between artificial and natural verb-object pairs.

$$sim_{Jacquard}(t, c) = \frac{\mid t \cap c \mid}{\mid t \cup c \mid} \tag{3.10}$$

The Jacquard coefficient divides the number of words in common by the total number of words. Like cosine, this ranges from 0 to 1. The primary difference from the cosine measure is that Jacquard's coefficient does not take into account the actual frequencies or weights in the distributions, so it can give a high similarity score even when the weights of the words are very different. We previously applied the Jacquard coefficient to topic adaptation but found that it was worse than cosine similarity.

Seymore and Rosenfeld (1997) found that Naïve Bayes was better than cosine in similarity scoring for topic modeling. Whereas pure similarity scores are approximations of $P(t \mid h)$ in conjunction with normalization, Naïve Bayes computes a true probability. Intuitively, Naïve Bayes is calculating the probability that the current document was produced by the language model of each topic, using the unigram model of each topic to compute the probability. Naïve Bayes implicitly accounts for topical salience — topical words will differentiate the scores (i.e., words where $P(w)$ is very different than $P(w \mid t)$ for any topic). 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:

$$P(t \mid h) = P(t) * \prod P(w \mid t)^{f_h(w)} \tag{3.11}$$

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 the probability of an arbitrary word in the corpus occurring

in topic t (i.e., the number of words in topic t divided by the total number of words) 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. The computations must be performed in log-space and normalized, but otherwise follow the equation above. We evaluated Naïve Bayes in our topic modeling system, but found that it led to much lower keystroke savings than cosine similarty. The poor performance of Naïve Bayes was unexpected, especially since Naïve Bayes is commonly accepted as a baseline method in text classification (Jurafsky and Martin, 2000) and was found to reduce perplexity slightly over TF-IDF for topic modeling for Seymore and Rosenfeld (1997).

In summary, we initially used cosine similarity to produce relevance scores for topic modeling, where the cosine scores were normalized to compute probabilities. We experimented with Jacquard's coefficient and Naïve Bayes following other researchers, but found that cosine was superior for our task.

3.1.4.3 Score Scaling

Florian and Yarowsky (1999) point out one problem with similarity-based approaches — when the current document is long, even irrelevant topics will have a nonzero relevance score and possibly distract the model from more relevant topics. Similarly, when the current document is short, there may only be one or two words that indicate the current topic, which can lead to similar scores for relevant and irrelevant topics alike. Therefore they applied a scaling function, which increases the maximum score to one and the minimum to zero:

$$sim'(t,h) = \frac{sim(t,h) - min_{t'}(sim(t',h))}{max_{t'}(sim(t',h)) - min_{t'}(sim(t',h))} \tag{3.12}$$

This will help accentuate the differences amongst the relevance 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. Florian and Yarowsky (1999)

found that scaling the cosine scores in this way helped reduce perplexity slightly. In our early experiments, we also found scaling to be beneficial and therefore include scaling in the topic model.

3.1.5 Topic Granularity: Human-labeled vs Document as Topic

Our approach to topic modeling leverages data that is segmented by topic. Documents in some corpora are manually assigned to topic labels, such as "air pollution" or "fishing" in Switchboard (Godfrey et al., 1992), which can be used for language modeling (Trnka et al., 2006b, Lesher and Rinkus, 2001). Some researchers use topics of a similar type, but rely on automatic document clustering rather than manual clustering of topics (Florian and Yarowsky, 1999). However, other researchers have used each individual document as a sort of topic (Mahajan et al., 1999). 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.

We refer to this spectrum as 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 as an example of 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).

In previous studies, we implemented and evaluated topic modeling at various granularities (Trnka, 2008a, Trnka, 2008b). At one extreme, we treated each document as a topic (fine-grained). The human-labeled topics in Switchboard were the basis for comparison, representing medium-grained topic modeling. At the other end of the spectrum, we treated each corpus as a general sort of topic, called coarse-grained topic modeling. We found that the human annotated topics of Switchboard

were best for topic modeling, though fine-grained topic modeling was a close second. Furthermore, we experimented with filtering out the least relevant documents in the fine-grained approach. We found that when the relevance scores were better (i.e., with stemming), that topic modeling was best with all documents included (although the least relevant documents only contribute slightly to keystroke savings). See Section B.6.2 for more detail. Coarse-grained topic modeling wasn't as promising, providing some improvement when a mix of in-domain and out-of-domain data is used for training, but was completely detrimental when only out-of-domain data was used.

Our previous studies are partially replicated in Section 3.2, which includes human-labeled topics and document-as-topic. The complete previous work can be found in Appendix B.

3.1.5.1 Improvements for Small Topics (Document-as-topic)

We treated documents as topics for fine-grained topic adaptation. The extreme data sparseness in the topic-specific models requires specific treatment to ensure that the model still works reasonably. The decision to combine frequencies rather than probabilities in our overall model was partially motivated by this sparseness problem. The main remaining issue is relevance scoring. We need to improve the chances of finding overlap between the cache model and the topic model. We found two improvements for document-as-topic modeling: Firstly, stemming both models helps to increase the potential amount of overlap in scoring. Secondly, we found that the sparseness of the distributions sometimes resulted in zero similarity, which we addressed with basic smoothing of the relevance scores.

3.1.5.1.1 Stemming

We applied Porter's stemmer to both the cache of the current document as well as the topic/document model. In basic experiments, we found that this

improved keystroke savings by about 0.2% when documents were used as topics, but decreased keystroke savings slightly with human-labeled topics. The decrease is likely because stemming is washing out some of the finer distinctions in topics. The benefit of stemming in this context is that it provides a simplified model of vocabulary use. The smaller effective vocabulary makes it easier to find overlap between documents, which partially compensates for smaller topics.

3.1.5.1.2 Similarity score smoothing

When a similarity score is zero, the respective topic is effectively removed from the language model. Although it is unlikely to be very useful for topical words, the data may be useful in improving the general-purpose nature of the language model. Also, sometimes even irrelevant documents are the only documents to contain specific vocabulary, and in general we have found that reducing the number of words in the vocabulary reduces keystroke savings, even when the words are uncommon.

We chose to add a percentage of the minimum non-zero similarity score to all scores. This results in a score distribution where no topics are assigned zero weight, but also the topics that previously had zero weight are assigned a smaller weight than any other topic. This unfortunately has a small interaction with similarity score scaling, so we only included the upscaling portion (see Section 3.1.4.3):

$$sim'(t,h) = \frac{sim(t,h) + \gamma * min_{t'|sim(t',h)>0}(sim(t',h))}{max_{t'}(sim(t',h)) + \gamma * min_{t'|sim(t',h)>0}(sim(t',h))} \tag{3.13}$$

We took $\gamma = 0.3$ to ensure that the smoothing did not overly affect the scores. The experimental results found that this smoothing method improved keystroke savings slightly on most corpora and did not decrease keystroke savings significantly on any corpora.

3.1.6 Summary of Prior Work

We adapt to the topic of discourse by first scoring topics for relevance, then by combining models of each individual topic in proportion to their relevance. Topic relevance scores measure the similarity of a unigram representation of the current partial document and each individual topic. We found that cosine similarity was best for relevance scoring, and included inverse topic frequency and recency in the model of the current document. The resulting scores were scaled so that the maximum was one and the minimum was zero, then normalized to create a probability distribution.

The relevance scores were used as weights in a linear combination of frequencies for trigram, bigram, and unigram models — each topic contributes to the frequencies in proportion to its score. After combining frequencies, we re-scale each distribution so that the original frequency sum is preserved, then discount the models to reserve probability for unseen events. The trigram, bigram, and unigram models are finally combined using backoff, which recursively distributes the held-out probability mass of a model to the next lower-order model.

We primarily used human-labeled topics, but also explored the idea of topic granularity — some topics labelings may be more specific or more general. In addition to human-labeled topics, we also explore document-as-topic as an example of a fine-grained topic distribution. The increased sparseness in relevance scoring leads us to adopt stemming and relevance score smoothing to perform topic adaptation with document-as-topic.

3.2 Evaluation

This section evaluates our approach to topic adaptation using both human labeled topics and also using each document as a topic. We have expanded the evaluation to allow for better comparison to other language modeling improvements. For example, style adaptation from Chapter 4 is evaluated over all words. Therefore, we included an evaluation of human-labeled topics on all words. Additionally, we

added perplexity evaluations to attempt to allow better comparisons between our topic model and other research in topic adaptation.

3.2.1 Human-labeled Topics (Switchboard)

This section will train the language models on Switchboard and test the models on each corpus. The human-labeled topics from Switchboard will be used to demonstrate topic adaptation.

Table 3.1 replicates our previous work. In this test, only fringe words are considered in evaluation. Additionally, core words are excluded from the predictions, modeling something like PRC's Pathfinder device. This evaluation demonstrates that topic adaptation can be beneficial, even when the content of the topics is from a very different corpus. In this case, the algorithm can latch onto small aspects of similarity and dissimilarity even if the vocabulary of training and testing data can differ somewhat. The adaptation produces a much larger improvement in keystroke savings when tested on Switchboard, where the topic adaptation is most natural. The absolute values of these results differ slightly from our previous results, but the trends are the same. The differences are most likely due to minor fixes in corpus cleanup as well as minor bug fixes.

The perplexity values in this test are extremely high because this evaluation focuses on fringe words, which is a very diverse set of words. Additionally, we expect that these tests have a much higher percentage of out-of-vocbulary terms, which severely degrade perplexity. The topic adaptation results in an extreme reduction in perplexity, though the reduction in keystroke savings is more modest.

Table 3.2 presents the results run on all words, not just fringe words. Even in the baseline model, most corpora show higher keystroke savings compared to the fringe-only evaluation — Switchboard is a better model of common, core words than less frequent, fringe words (especially on other corpora). Unfortunately, these results are not as promising for topic adaptation. Switchboard and Slate show an

	Baseline		Switchboard topics		Significance
Testing corpus	KS	PP	KS	PP	
AAC Emails	43.75%	123,417	44.13% (+0.38)	52,318 (-57.6%)	$p < 0.001$
Santa Barbara	43.64%	49,144	44.05% (+0.41)	18,463 (-62.4%)	$p < 0.001$
Callhome	50.81%	19,752	50.98% (+0.17)	8,661 (-56.2%)	$p < 0.004$
Charlotte	49.98%	35,513	50.37% (+0.39)	15,686 (-55.8%)	$p < 0.001$
Micase	44.68%	80,648	45.15% (+0.47)	35,353 (-56.2%)	$p < 0.001$
Switchboard*	60.69%	4,132	61.89% (+1.20)	1,868 (-54.8%)	$p < 0.001$
Slate*	36.86%	383,591	37.47% (+0.61)	188,818 (-50.8%)	$p < 0.001$

Table 3.1: Switchboard topics, fringe-only Medium-grained topic modeling using human-annotated topics, trained on Switchboard. The baseline method is a non-adaptive trigram model trained on Switchboard. These results are measured on fringe words only. *cross-validation not performed due to time constraints.

overall improvement in keystroke savings, but the results on other corpora are mixed. The small changes coupled with the smaller corpora result in a lack of statistical significance on many corpora. The perplexity results are more encouraging, but they do not correlate very well with the changes in keystroke savings.

Our topic modeling methods are not designed to provide useful adaptation for core words, leading to smaller changes in keystroke savings when all words are considered. The degradation of keystroke savings on Callhome and Charlotte are likely the result of adapting to words that comprise less than half of the keystroke savings — the language models are optimized for topical words, but may degrade keystroke savings on non-topical words slightly, resulting in an overall loss of keystroke savings. This loss in keystroke savings could likely be eliminated by preventing the topic adaptation from affecting the probabilities of non-topical words. In the future, we may implement this by separately adapting topical and non-topical words. Alternatively, the use of ITF may have been beneficial for fringe word evaluations, but may cause small problems for the prediction of core words.

	Baseline		Switchboard topics		Significance
Testing corpus	KS	PP	KS	PP	
AAC Emails	49.37%	965	49.38% (+0.01)	930 (-3.6%)	not significant
Santa Barbara	48.55%	466	48.62% (+0.07)	391 (-16.1%)	$p < 0.005$
Callhome	54.15%	254	54.02% (-0.13)	241 (-5.1%)	$p < 0.005$
Charlotte	54.43%	301	54.40% (-0.03)	295 (-2.0%)	not significant
Micase	50.36%	650	50.36% (+0.00)	623 (-4.2%)	not significant
Switchboard*	58.84%	147	59.04% (+0.20)	146 (-0.7%)	$p < 0.005$
Slate*	42.44%	5,330	42.70% (+0.26)	4,787 (-10.2%)	$p < 0.005$

Table 3.2: Switchboard-topics, all words Medium-grained topic modeling using human-annotated topics, trained on Switchboard. The baseline method is a non-adaptive trigram model trained on Switchboard. The results are measured on all words. *cross-validation not performed due to time constraints

3.2.2 Document as Topic

This section presents our evaluation of document as topic. Some of the tests were updated with perplexity results, but the trends of perplexity on fringe words is the same — perplexity scores are very large, and topic adaptation yields a large reduction in perplexity.

Table 3.3 shows the results of document-as-topic adaptation on all corpora with in-domain evaluation (i.e., training and testing on the same corpora). This first evaluation only measures keystroke savings on fringe words, similar to Table 3.1. Because of the in-domain evaluation, these results are only directly comparable on Switchboard, where document-as-topic gives a 1.12% improvement in keystroke savings compared to 1.20% improvement with human-annotated topics. The other corpora show significant improvements aside from Callhome, which is not significant. The perplexity results are similar to Table 3.1 — the numbers are very large due to the large set and difficult predictability of fringe words. Topic adaptation again reduces perplexity substantially, but the impact on keystroke savings is smaller. Fine-grained topic modeling gives similar results without the need for annotation,

making it applicable to training on not just Switchboard but other corpora as well.

	Baseline		Documents as topics		Significance
Testing corpus	KS	PP	KS	PP	
AAC Emails	45.90%	6,269	46.32% (+0.42)	3,549 (-43.4%)	$p < 0.001$
Santa Barbara	39.96%	20,893	40.22% (+0.26)	9,446 (-54.8%)	$p < 0.001$
Callhome	40.21%	8126	40.16% (-0.05)	4420 (-45.6%)	not significant
Charlotte	45.81%	19,892	46.11% (+0.30)	9,849 (-50.5%)	$p < 0.001$
Micase	46.95%	20,548	47.52% (+0.57)	10,869 (-47.1%)	$p < 0.001$
Switchboard*	60.69%	4,132	61.81% (+1.12)	1,872 (-54.7%)	$p < 0.001$
Slate*	52.88%	14,781	53.84% (-0.04)	8,425 (-43.0%)	$p < 0.001$

Table 3.3: Document-as-topic, in-domain, fringe only 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

Table 3.4 shows our previous results with out-of-domain evaluation. The topic adaptation improves keystroke savings even when the training data is not very relevant, and is arguably even more beneficial in this test than the in-domain test.

Table 3.5 shows our previous results with mixed training data. Document-as-topic is extremely helpful here. The results of fine-grained topic modeling with mixed-domain training are the best we have for most corpora. Partly, the reason for the benefit is the ability of the model to easily decide which parts of the training data are most relevant, and appropriately weight relevant training data higher. Much of the relevant training data is likely from the same corpus as the testing data, meaning that the adaptation process is facilitating the combination of many diverse sources of text.

3.3 Iterative Re-training with Fine-grained Topics

Topic modeling is less beneficial when all of the training data is irrelevant (i.e., out-of-domain). In a real system, we can avoid this problem by incorporating

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 3.4: Document-as-topic, out-of-domain, fringe only 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

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 3.5: Document-as-topic, mixed-domain, fringe only 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

past user texts into the model. This section explores the possibility of re-training the model with each testing document after evaluation, much like a system would update the model with user data.

The problem in integrating user data is how to combine the user data with the baseline model. If frequencies are combined without weights, then the user data would not contribute much to the overall model and the predictions may still be inappropriate. However, we can use our topic model to appropriately weight user data versus out-of-domain data. If we use documents as topics, then the user text can be added into the model without any additional effort. We expect the topic relevance scores to tune the model more to user texts than irrelevant texts.

The intention of this feature is to help improve keystroke savings when the training data is very different from actual usage. We will perform a pilot evaluation with two corpora: AAC Email and Santa Barbara. The evaluation will approximate the problem with out-of-domain training — training on Santa Barbara to evaluate on AAC Emails and training on AAC Emails to evaluate on Santa Barbara. The topic model will use documents as topics, both in the baseline topic model and the improved topic model. This allows user texts to be integrated like any other document. In evaluation, when a testing document is completed, we will train a model on the document and add it to the set of texts for adaptation. However, note that this evaluation differs from user adaptation — the corpora have multiple speakers.

Table 3.6 shows the results of the trigram baseline, the topic model (with document as topic), and the topic model with iterative re-training (with document as topic). Although topic modeling itself is not beneficial in this test, iterative re-training is very useful — improving over the topic model by 1.34% on AAC Emails and 3.64% on Santa Barbara.

Testing corpus	Baseline	Topic	Topic (iterative re-training)
AAC Emails	44.37%	43.95%	45.29%
Santa Barbara	36.21%	36.17%	39.81%

Table 3.6: Iterative re-training, out-of-domain evaluation. The baseline model is an unadaptive trigram model and the topic models use document-as-topic.

The pilot evaluation shows that iterative re-training can substantially increase keystroke savings when the training data is largely irrelevant. The approach we have taken is relatively simple — a better one might have a hierarchical weighting system, at the top level for all user data compared to external data and then within that by document. Additionally, a more advanced approach might better leverage the consistent style of a user's text, especially for core words. However, in this evaluation, we could only track corpora, not specific users within each corpus.

We conclude that integrating user texts is beneficial, but needs further evaluation. However, the evaluation depends both on the training/testing split of the corpora as well as the order of the testing documents. For example, an evaluation with 2-fold cross-validation will likely be more favorable for iterative re-training compared to 11-fold cross-validation. Furthermore, the feature has no benefit whatsoever in leave-one-out testing. We would like to address these problems in future work. The cross-validation problem can be avoided with out-of-domain training, but our topic simulations unfortunately still separately evaluate each set, reducing the benefit. We plan to address this by improving the evaluation code for topic adaptation. We hope to initially address the testing data order by evaluating the potential impact of order on the improvement and measuring upper and lower bounds on the effect.

3.4 Automatic Clustering for Topic Modeling

One of the disadvantages of our initial approach to topic modeling is the reliance on a corpus with human-labeled topics — a relatively uncommon resource. The use of a document as a topic allows us to shed that requirement, but the approach is computationally demanding every time word prediction is necessary. In this section, we will explore the use of clustering to automatically derive topics that mimic human-labeled topics. This has the advantage of not requiring laborious human annotation and is adaptable to any type of corpus. In general, our approach with clustering is similar to (Florian and Yarowsky, 1999), but we apply a flat clustering.

3.4.1 Clustering Method

Topic modeling is designed for a flat grouping of documents, in contrast to methods that fully leverage a hierarchical clustering (such as Florian and Yarowsky (1999)). Our formulation of topic adaptation is naturally suited to a flat clustering method like k-means. However, k-means clustering can depend on the initial assignment of documents to clusters. Instead we applied agglomerative clustering like Florian and Yarowsky (1999), and afterwards devised a flat clustering by undoing the most dissimilar merges. The number of merges undone is determined by our desired number of clusters.

In contrast to traditional clustering research, we have multiple corpora to cluster, and we also normally evaluate using cross-validation. The two problems are similar — given a training and testing set, should we build clusters that involve data outside of the training set? These problems are discussed in Section 3.4.1.3. In addition to the ordinary problem of choosing a clustering method and similarity metric, maintaining a consistent interpretation of cluster size across corpora proved to be challenging. We will discuss the meaning of cluster size in Section 3.4.1.2.

For the similarity metric used in clustering, we started with cosine similarity between unigram representations of each document (or cluster). However, we found that this had a tendency to merge single documents into an iteratively built centroid of the corpus. When we flattened this clustering, one cluster contained the majority of the documents and the other clusters were all singletons. Topic adaptation with this type of clustering is unlikely to provide any benefit. We suspected that common words were responsible for this behavior. Therefore we applied inverse document frequency (IDF) in our similarity metric to focus the clustering on the most topically salient words. However, this only partially addressed the issue. Similarly, we tried stemming the unigram distributions, which also only helped somewhat. Additionally, we tried smoothing each document's distribution with the distribution of the overall corpus. This also had some benefit, but not enough. To fully address the issues, we needed to add a weight to prefer merges between smaller clusters, following Florian and Yarowsky (1999). The weighting scheme will be discussed further in Section 3.4.1.2.

3.4.1.1 Re-creating Clusters for Switchboard

To verify that our clusters were of acceptable quality, we attempted to recreate Switchboard's human-defined clusters with automatic clustering. We performed agglomerative clustering as normal and split from the top down until we had the same number of clusters as the human-defined topics.

We performed a subjective evaluation of the cluster quality in two ways. The first method iterated over all clusters, and considers all pairs of documents within the cluster. If the two documents also share the same topic label, we increment the number of matches. Afterwards, we normalize by the maximum possible matches. We additionally track the frequency of topics within each cluster and display a listing. The second evaluation repeats the procedure, but instead iterates over topics and checks for matches in the clustering.

We found that the human-annotated topics and our automated clusters agreed often — results were over 80% overlap for most topics. Additionally, the "confused" documents often displayed natural tendencies, such as confusion between family life and school.

3.4.1.2 What's the Size of a Topic?

Although the question of clustering on Switchboard is reasonably clear (we know how many clusters to use), clustering on other corpora is much less clear. We can choose 70 clusters to mimic human-labeling on Switchboard, which has over 2,000 documents. Should a corpus of 4,000 documents have 70 clusters? 140 clusters? Or fewer? As a thought experiment, we can create an artificial corpus by taking Switchboard and duplicating each document. We would like our clustering method to derive 70 clusters in such an artificial scenario. The size of a topic in the number of documents should be dependent on how specific or general the documents are with respect to the overall corpus. Intuitively, there is a "right size" for a topic, such as the specificity of encyclopedia articles.

We take a first step towards characterizing topical specificity by leveraging similarity thresholds in clustering. We will create a threshold across corpora, which we hope will create topics of similar specificity regardless of corpus. However, for this approach to work, the scale of the similarity scores must be the same across corpora. Although we would like to use the similarity threshold from Florian and Yarowsky (1999), their similarity score varies across corpora, shown below.

$$sim(c_1, c_2) = \left(\frac{1}{D_{c_1}} * \frac{1}{D_{c_2}} \right)^{\alpha} * cos(c_1, c_2) \tag{3.14}$$

where c_1 and c_2 are the clusters that are being considered for merging, D_{c_i} is the number of documents in cluster i and α is a tunable weight. In this equation, the normal cosine score is being weighted to prefer merging smaller clusters (to

counteract the problems we encountered). We tried several variations of a weighted cosine score, but found the following weighted cosine to work well:

$$sim(c_1, c_2) = \left(1 - \frac{D_{c_1} - 1}{D_* - 1} * \frac{D_{c_2} - 1}{D_* - 1}\right)^{\alpha} * cos(c_1, c_2) \tag{3.15}$$

where D_* is the total number of documents. The weight component for each cluster ranges from nearly one when the cluster contains all but one document, to zero when the cluster contains only one document.

To briefly summarize, agglomerative clustering worked very poorly with just a pure cosine similarity, even with tweaks such as IDF, stemming, and smoothing. The solution we settled on was to include a weight which prefers merging smaller clusters. However, we had to take care to both apply size to the weighting and also to maintain a score that ranges from 0–1, rather than a range based on the size of the corpus. The effect of this effort is that similarity scores have a much more consistent meaning across corpora.

We automatically found the threshold t on the similarity score such that using t to split apart the hierarchy produced the same number of clusters on Switchboard as the number of human-labeled topics on Switchboard. In this manner, we have effectively converted the humans' decisions regarding topic specificity into a similarity threshold. We then applied this threshold to all corpora to derive a number of clusters that maintains a comparable specificity to the original Switchboard topics. We evaluated these clusters in topic modeling, described in Section 3.4.2. Additionally, we experimented with clustering where the threshold was 25% greater and 25% smaller to get a better sense of whether more specific or more general clusterings were better for word prediction. The actual number of clusters for each corpus is shown in Table 3.7. The number of clusters correlates somewhat with the number of documents, but is also a function of the actual words in the corpora. The only truly unfortunate clustering is Callhome — topic adaptation will likely have no effect with only one topic.

		Number of clusters		
Corpus	Number of documents	$t = 75\%$	$t = 100\%$	$t = 125\%$
AAC Emails	117	3	8	13
Santa Barbara	60	9	13	16
Callhome	24	1	1	2
Charlotte	93	4	5	11
Micase	50	7	11	14
Switchboard	2,438	59	70	84
Slate	4,531	259	372	476

Table 3.7: Number of clusters as a function of the number of documents in each corpus and the similarity threshold.

3.4.1.3 Problems in Clustering

In a collection of corpora, the question is whether we should cluster the entire collection or each individual corpus. We decided to individually cluster each corpus rather than create a single clustering of all corpora. Especially for a collection with diverse vocabulary, we feel that clustering each corpus individually is the right decision. In contrast, an overall clustering may present issues to domain-varied evaluation. Due to the diversity of domains of our corpora, it is likely that the clusters would fall along corpus lines. Additionally, due to the runtime complexity, clustering all corpora together would be time prohibitive. Therefore we decided to cluster each corpus separately.

Ideally we would like to build our clusters from only the training data used in each iteration of cross-validation. However, that requires 11 different clusterings for each corpus. Instead, we decided to perform a single clustering for each corpus. Although this allows the structure of the clusters to reflect both training and testing data, we feel that the extra 1/11 of the corpus (the testing data) would not significantly affect the clustering.

We also acknowledge a problem with cross-validation that is similar to our previous work with human-labeled topics — the training sets are not balanced by

cluster. It could be that one cluster is highly represented in set 1, and not represented at all in set 2. This imbalance could reduce the potential benefit of cluster-based adaptation. We would like to ensure that the cluster distributions are reasonably balanced across sets. However, we have not addressed the problem yet, but we propose a method for future work. The sets might be balanced for any topic-like problem by balancing sets such that the unigram distributions are maximally similar.

3.4.2 Evaluation

Our evaluations with clustering focus purely on in-domain training, as the clusters are predominantly meaningful within a corpus. In the future we may extend this to include out-of-domain or mixed-domain training. Most of the results in this section are evaluated on all words, not just fringe words. One evaluation of fringe words is included for comparison to our previous topic modeling studies.

Tables 3.8 and 3.9 evaluate topic modeling with clustering, using clusters that mimic Switchboard topics' specificity. The results compare favorably to human topics (i.e., Table 3.1 on page 55). The only directly comparable result is on Switchboard — 1.14% improvement for clusters compared to 1.20% for human-labeled topics with fringe evaluation. The evaluation on all-words is a little less favorable (compared to Table 3.2 on page 56) — 0.13% improvement for clusters on Switchboard compared to 0.20% with human-labeled topics. The results are directly comparable to the in-domain test of fine-grained topic modeling (Table 3.3 on page 57). Clusters are better for topic modeling on Switchboard, but document-as-topic is better on most other corpora. We can think of clustering as a spectrum of topic granularity, the baseline model is the most general clustering (one group of data) and document-as-topic is the most specific grouping. The results in comparison to the baseline and document-as-topic suggest that the clusters are too general for most corpora. Certainly, having a single cluster on Callhome is not useful. The following two evaluations will consider more general clusters in Table 3.10 and more specific

clusters in Table 3.11. These followup evaluations measure keystroke savings over all words, not just fringe words.

	Baseline		**Clusters** ($t = 100\%$)		**Significance**
Testing corpus	KS	PP	KS	PP	
AAC Emails	49.45%	427	49.57% (+0.12)	437 (+2.3%)	$p < 0.05$
Santa Barbara	46.79%	384	46.80% (+0.01)	339 (-13.3%)	not significant
Callhome	49.03%	279	49.03% (+0.00)	247 (-11.5%)	not significant
Charlotte	52.88%	272	52.84% (-0.04)	277 (+1.8%)	not significant
Micase	51.61%	428	51.55% (-0.06)	443 (-3.5%)	$p < 0.002$
Switchboard*	58.84%	147	58.97% (+0.13)	149 (+1.4%)	$p < 0.001$
Slate*	54.33%	839	54.72% (+0.39)	817 (-2.6%)	$p < 0.001$

Table 3.8: Clustered topics at $t = 1.0$, in-domain, all words Medium-grained topic modeling using clusters for topics, trained and tested on each corpus. The baseline method is a non-adaptive trigram model trained and tested on the same corpus. These results are measured on all words. *cross-validation not performed due to time constraints.

Table 3.10 shows topic modeling with a fewer number of clusters, attempting to create clusters that are 25% more general than the original Switchboard topics. The results show that more general clusters are less beneficial across the corpora. Although the trend is slight, we feel this echoes our findings with using very large topics (corpus as a topic) in Appendix B.

Table 3.11 shows topic modeling with a greater number of clusters, attempting to create clusters that are 25% more specific than the original Switchboard topics. The more specific topics show some improvements and some degradation compared to normal clusters. The evaluation on Switchboard brings the results much closer to performance with human-annotated topics.

In general, our evaluations suggest that clustering is a reasonable way of avoiding the need for human annotation while improving keystroke savings. Smaller clusters seem to be better than more general clusters. Our comparisons with fine-grained topic modeling suggest that the optimal topic size is much smaller than

	Baseline		Clusters ($t = 100\%$)		Significance
Testing corpus	KS	PP	KS	PP	
AAC Emails	45.90%	6,269	46.15% (+0.25)	3,677 (-41.3%)	$p < 0.003$
Santa Barbara	39.96%	20,893	40.01% (+0.05)	9,298 (-55.5%)	not significant
Callhome	40.21%	8,126	40.19% (-0.02)	2,193 (-73.0%)	not significant
Charlotte	45.81%	19,892	45.86% (+0.05)	10,300 (-48.2%)	not significant
Micase	46.95%	20,548	47.31% (+0.36)	11,318 (-44.9%)	$p < 0.001$
Switchboard*	60.69%	4,132	61.83% (+1.14)	1,891 (-54.2%)	$p < 0.001$

Table 3.9: Clustered topics at $t = 1.0$, in-domain, fringe words only Medium-grained topic modeling using clusters for topics, trained and tested on each corpus. The baseline method is a non-adaptive trigram model trained and tested on the same corpus. These results are measured on fringe words only. *cross-validation not performed due to time constraints.

	Baseline		Clusters ($t = 75\%$)		Significance
Testing corpus	KS	PP	KS	PP	
AAC Emails	49.45%	427	49.46% (+0.01)	448 (+4.9%)	not significant
Santa Barbara	46.79%	384	46.79% (+0.00)	342 (-10.9%)	not significant
Callhome	49.03%	279	49.03% (+0.00)	247 (-11.5%)	not significant
Charlotte	52.88%	272	52.84% (-0.04)	278 (+2.2%)	not significant
Micase	51.61%	428	51.51% (-0.10)	446 (+4.2%)	$p < 0.001$
Switchboard*	58.84%	147	58.95% (+0.11)	149 (+1.4%)	$p < 0.001$

Table 3.10: Clustered topics at $t = 0.75$, in-domain, all words Medium-grained topic modeling using clusters for topics, trained and tested on each corpus. The baseline method is a non-adaptive trigram model trained and tested on the same corpus. These results are measured on all words. *cross-validation not performed due to time constraints.

	Baseline		Clusters ($t = 125\%$)		Significance
Testing corpus	KS	PP	KS	PP	
AAC Emails	49.45%	427	49.56% (+0.11)	433 (+1.4%)	$p < 0.03$
Santa Barbara	46.79%	384	46.81% (+0.02)	341 (-11.2%)	not significant
Callhome	49.03%	279	48.91% (-0.12)	267 (-4.3%)	$p < 0.002$
Charlotte	52.88%	272	52.86% (-0.02)	277 (+1.8%)	not significant
Micase	51.61%	428	51.54% (-0.07)	445 (+4.0%)	$p < 0.001$
Switchboard*	58.84%	147	59.01 (+0.17)	147 (+0.0%)	$p < 0.001$

Table 3.11: Clustered topics at $t = 1.25$, in-domain, all words Medium-grained topic modeling using clusters for topics, trained and tested on each corpus. The baseline method is a non-adaptive trigram model trained and tested on the same corpus. These results are measured on all words *cross-validation not performed due to time constraints.

the human-labeled topics of Switchboard, but also larger than a single document. In comparison to document-as-topic, cluster-as-topic can be thought of as a two-stage process for topic relevance scoring. The first stage is clustering, which groups similar texts together. Then the second stage identifies relevant clusters. The effect is similar to a more generalized topic relevance score — although a cluster may be deemed relevant based on keyword overlap, there may be documents in the cluster that share no keywords with the current document. The only requirement is keyword overlap with other documents in the cluster. Therefore, clustering can be considered as a generalization process that allows us to not just predict words that cooccur with words in the current document, but also more loosely related words are likely to be more probable.

In contrast to other methods, we feel that clustering leads to a much faster system and has the potential for higher keystroke savings. We have demonstrated some principles for clustering, but further work is necessary to develop a better approach. We feel that clustering for topic adaptation merits further research, both in clustering methods as well as defining a corpus-independent measure of cluster specificity.

3.5 Related Work

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. See Bellegarda (2004) for a survey of adaptive methods. 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.

3.5.1 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 (Seymore and Rosenfeld, 1997, Mahajan et al., 1999, Florian and Yarowsky, 1999, Clarkson and Robinson, 1997). 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 (Clarkson and Robinson, 1997, Gildea and Hofmann, 1999, Iyer and Ostendorf, 1999, Adda et al., 1999). The Naïve Bayes method fits more closely with traditional research in topic classification, used by (Seymore and Rosenfeld, 1997). The cosine score approach takes an Information Retrieval mindset and has been applied to topic modeling by (Seymore and Rosenfeld, 1997, Mahajan et al., 1999, Florian and Yarowsky, 1999). 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 (2001) 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 (1997) 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. 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. (1998) 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. (1998a) 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 (1994) (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. (1999) 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.[3] 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 3.1.5) — individual document are often too small to have reliable unigram distributions for topic identification, so stemming the distribution

[3] 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.

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 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 (1997). 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 (1999) 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 3.1.4.3. 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 (Clarkson and Robinson, 1997, Gildea and Hofmann, 1999, Iyer and Ostendorf, 1999, Adda et al., 1999). 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 (1997) 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 (1999) 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 discrepancy between perplexity and word error rate is explained in Chen et al., 1998 (1998b). 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 (1999) 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. (1999) 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 3.1.5). 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 (1993) 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. (1999) 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.

3.5.2 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. (1990)[4]. 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, 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. (1990) used 100 dimensions for document retrieval in a relatively small corpus (1000–2000 documents, 5000–7000 terms). Dumais et al. (1997) applied LSA to cross-language document retrieval using 330 semantic dimensions. Landauer et al. (1998) experimented with 2–1032 dimensions in applying LSA to the TOEFL vocabulary test and found that performance peaked with 300–325 dimensions. The

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

optimal choice of k may vary from one application to another, but researchers generally select values in the hundreds.

(Wandmacher and Antoine, 2007, Wandmacher et al., 2007) 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[5] and train the LSA model on 100 million words of news text using the Infomap toolkit.[6] 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.[7] 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 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

[5] http://www.speech.sri.com/projects/srilm/ as of 11/2010

[6] http://infomap-nlp.sourceforge.net/ as of 11/2010

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

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 (2000, 1998a, 1998b) 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. 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 (1998b) 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) (Bellegarda, 1998a). 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) (Bellegarda, 1998b). 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.[8] 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%

[8] 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.

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 (2000) 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 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 (2006, 2007). 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.

Hofmann (1999) builds on research in LSA to develop probabilistic latent semantic analysis (pLSA). They represent the generation of a word as a two stage

process: First, a set of (latent) topics and associated probabilities are generated from the document. Then, a given word is generated according to the topic distribution. Like LSA methods, the algorithm only describes words out of context, similar to unigram distributions. However, unlike LSA, pLSA does not utilize singular value decomposition for dimensionality reduction. Instead, pLSA uses the EM algorithm along with the structure of the probabilistic model to produce latent topics. They provide a basic evaluation of pLSA for language modeling, however, pLSA is only compared to unigram-like methods. pLSA reduces perplexity by 50-70% over the unigram model, however, in the context of our language model, many of the improvements can be accounted for by simple ngram models of context. For example, consider the word pair "language modeling". Although pLSA would substantially reduce the perplexity of "modeling" in this context, a simple bigram model would also achieve the same result, if not better.

Blei et al. (2003) point out many of the deficiencies of pLSA, notably that the generation of topics from documents does not generalize to new documents, preventing the model from being used as a true generative language model. They instead create a new model, latent dirichlet allocation (LDA). LDA addresses issues in both LSA and pLSA to form a true generative model. Unlike LSA, LDA does not require any extra effort to create a generative model. In contrast to pLSA, LDA is applicable to documents outside of the training corpus. However, like both LSA and pLSA, LDA views the generation of words without context, solely as the product of the latent topic distribution. They found that LDA reduced perplexity over the unigram baseline by about 50% whereas pLSA reduced perplexity by closer to 35%. In general, LDA tends to require fewer latent topics than pLSA — the results improve more slowly with additional topics.

Hsu and Glass (2006) apply LDA to topic adaptation for lecture transcription. Their language model is structured first as a hidden markov model (HMM) and

then as latent dirichlet allocation for topical words. The HMM is similar to the part of speech ngram model of Chapter 4 except that the word classes are learned automatically rather than using part of speech tags. One of the word classes is reserved for topical words, and the LDA model is used for generating these words. They do not provide an evaluation of the LDA component, but find that a mixture of models appropriate for topic and style together yield a perplexity reduction of about 16%.

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 (Li and Hirst, 2005, Matiasek and Baroni, 2003, Lau et al., 1993, Rosenfeld, 1994, Rosenfeld, 1996). 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 (2005) 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 (Lau et al., 1993). 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 (we evaluate this for our system in Section D.1). 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 (2006) 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. (2002) 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. (1993) by using average mutual information to rank candidate trigger pairs. Matiasek and Baroni (2003) 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%.[9] 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 (1996) might fare better. Trost et al. (2005) 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),

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

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 (2007) 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 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. (1993) 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 (1994) extended Lau et al. (1993)'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 (1996) 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. (2000) 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.

3.5.4 Other Topic Adaptations

Other researchers such as Berger and Miller (1998) and Lesher and Higginbotham (2005) 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. (1999), 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.

3.6 Conclusions

In summary, we have demonstrated an effective topic adaptation for word prediction. Topic adaptation is viewed as a two-stage process: First, the current partial document is compared against the set of topics and each topic is scored for topical relevance. Secondly, word ngram distributions from each topic are combined according to their relevance score, allowing all topics to have an effect, but focusing on the most relevant topics. We found that it was best to combine frequencies rather than probabilities to prevent over-smoothing the model.

We found that cosine similarity with inverse topic frequency was the best method for topical relevance. We also found that exponential decay was useful for domains where the topic may shift throughout a document.

We found that we could use human-annotated topics, but our methods were applicable to clustering as well as treating each document as a very specific topic. We found that clustering had interactions with our evaluation framework, and that clustering across multiple corpora made it difficult to maintain a consistent granularity for the topics. Therefore, we devised a similarity score for clustering that was independent of corpus size. In contrast, the difficulties with document-as-topic reflected the sparseness of the unigram models for each topic. Therefore, we found it beneficial to apply stemming in relevance scoring for small documents. We also found it beneficial to add a small weight to all topics to ensure that none were eliminated from the model.

In contrast to other researchers, we have focused on a language model that explicitly models topic adaptation. Also, our topic model differs from latent semantic models and trigger pair models by allowing topic to affect contextual distributions

(i.e., trigrams, bigrams) rather than purely creating a unigram-like distribution to combine with an unadaptive source of context.

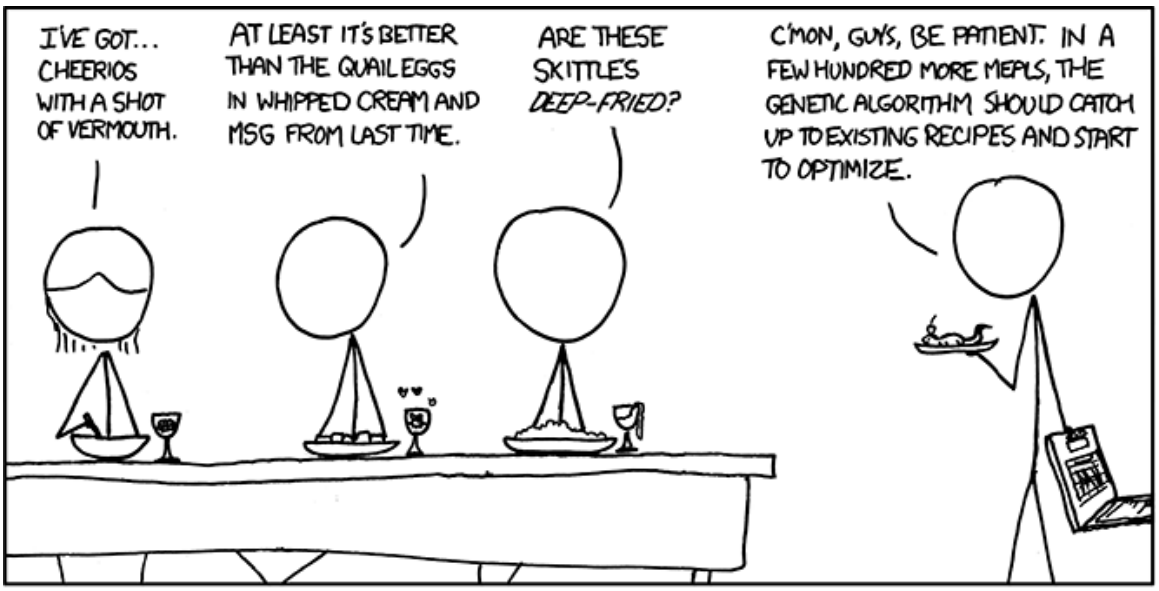
In the context of our larger work, topic modeling addresses some of the problems in mixed-domain evaluation — when a large mixture of corpora are used for training, less relevant documents/corpora can decrease the benefit of more relevant texts. Topic adaptation addresses the problem by dynamically tuning weights for each piece of training data. Therefore, topic adaptation helps create a language model that reflects a more appropriate weighting of in-domain versus out-of-domain training data. Additionally, dynamic adaptation accounts for the shifting meaning of in-domain and out-of-domain — even within the same corpus, the set of topics that are relevant for one testing document may not be relevant for another testing document.

In the context of word prediction, topic adaptation has many benefits. Firstly, the predictions are more appropriate to the user and should increase keystroke savings and decrease any cognitive disruption from irrelevant words, especially in systems that only use word prediction for fringe words. Secondly, in contrast to traditional AAC vocabulary sets, topic adaptation requires no user intervention to predict words in a given topic. Additionally, topic modeling never removes words from the predictions (unlike vocabulary sets). This also benefits practitioners and device manufacturers by eliminating the development need for vocabulary sets. Thirdly, topic adaptation allows device manufacturers to include a more diverse mixture of training data for their language models. This may also translate to a benefit for practitioners, who may be able to import topic-specific vocabulary from textbooks more easily. Finally, topic adaptation provides a mechanism for user adaptation by storing user texts and including them in the adaptive framework, which will derive a meaningful combination of user texts and static training data.

In the future, we would like to evaluate document-as-topic and cluster-as-topic with mixed domain evaluation on all words. We expect that these evaluations will show more benefit than in-domain evaluations, and also more benefit than Switchboard topics on other corpora. We would also like to expand the iterative re-training work to include full-scale evaluations. Also, we feel that the clustering methods could be improved significantly, and improvements in clustering may lead to cluster-as-topic as the superior model, both in terms of computational time and keystroke savings.

Chapter 4

STYLE ADAPTATION



xkcd.com/720

Style is an aspect of communication that reflects needs of the communication medium, the relationship between the speaker and listener, the environment, and other factors. For example, spoken and written language differ at a basic level due (in part) to different communication speeds — a writer has more time for message planning, while a speaker can catch up with message planning and then require a

speech repair. Similarly, a reader has more time to consider writing, thus allowing the writer to employ more complex language without the communication breakdown of spoken language. Biber (1988) analyzes the variation between speech and writing in terms of dimensions of linguistic variation (e.g., formal/informal, interactive/non-interactive) which are described in terms of collections of linguistic features (e.g., number of passives, nominalizations). Similarly, Michos (1996a) and Kessler et al. (1997) describe style as a three-level process: At the most general level, style can be treated as categories or genres, such as email or face-to-face chat. The overall style of a document affects various facets or features of the document at the second level, such as formality, grammatical complexity, directed vs undirected speech, etc. Therehe facets of style then influence the third level — actual communication or realization. For instance, a formal style might have more passive verbs than an informal style, but fewer contractions. The realization of style in natural language generation can also be treated as a lexical feature and addressed by unifying lexical features with the document features (Hovy, 1987).

Kessler et al. separate the bottom effects of style into three types of cues —character-level (e.g., punctuation) cues, lexical cues, and structural/grammatical cues. Although character-level cues may be useful in differentiating certain kinds of writing (e.g., technical/programming vs news), the focus of our system is conversational language without significant punctuation. In contrast, lexicalized style varies even in spoken language — academic prose might contain more descriptive words than more colloquial discussions. However, topic adaptation may serve as an effective model of lexicalized style. For example, consider the stylistic difference between "insect" and "bug" in the context of topic adaptation. A topic model would tune to words in similar contexts as "insect" or "bug", increasing the likelihood of formal words of similar topic to "insect" or informal words of similar topic to "bug".

In contrast to character-level cues of style and lexical cues of style, we will

focus purely on the grammatical aspect of style. We will ignore lexical aspects of style in this chapter because topic adaptation may already account for them in part, and also to focus on aspects of style that are purely orthogonal to topic to eventually combine topic and style adaptation into a single model.

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). The following example demonstrates stylistic variation despite very similar content. The first snippet comes from emails in our research group, while the second is on the same topic but from a formal paper.

> 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.

In terms of the three levels, the most general level is simply a description of the type of document (i.e., technical email vs technical article). At the second level, the email is informal and directed (i.e., the reader is known to the writer) whereas the paper snippet is formal and less directed. Not all features of the styles differ however; both are technical in nature. At the bottom level, the realization is different — the email contains more pronouns whereas the article contains more passive verbs. The article also contains much longer noun phrases, both in terms of pre-modifiers (i.e., nouns, adjectives) and prepositional phrases. We will focus on the grammatical aspect of style using part of speech tags. For example, VBN (past

participle) is more probable in formal writing due to passive constructions and "NN NN" is also more common in the technical articles. PRP (personal pronoun) is more common in emails and also "TO VB" ("to" followed by a base form verb) is more common in emails.

We plan for the style-adapted language model to tune word prediction to discourse needs (e.g., formality, sentence complexity, time constraints), complementing a topic model which focuses on adapting to the content of conversation. Specifically, we will use part of speech tags to model the grammatical effects of style, which is complementary to the lexical variations modeled by topic adaptation. Our experience with topic adaptation 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 build a part of speech ngram model that tunes the transition probabilities based on stylistic similarity scores. The equation below illustrates the overall process:

$$P_{style}(w \mid h) = \sum_{s \in styles} P(s \mid h) * \sum_{tag \in POS(w)} P(tag \mid tag_{-1}, tag_{-2}, s) * P(w \mid tag) \tag{4.1}$$

At the outermost layer, we are performing a linear combination of style-specific part-of-speech models using $P(s \mid h)$ as weights. The combination affects the transition probabilities $P(tag \mid tag_{-1}, tag_{-2}, s)$. We only adapt the transition probabilities (and not emission probabilities) because we are focusing only on grammatical variations due to style. The part of speech base for the model will be explained further in Section 4.3.

Regardless of the structure of the model, we first need to part of speech tag our corpora. We will describe our efforts in tagging the corpora in Section 4.1. Section 4.2 will present proof-of-concept work for style adaptation before we actually

develop the model. After demonstrating that style can be identified using POS tags, we will describe the efforts in developing the baseline model — an unadaptive part of speech ngram model — in Section 4.3. Section 4.4 will demonstrate topic modeling in the POS ngram model. The methods will be largely similar to previous developments in topic adaptation, but the purpose of this model is two-fold: Firstly, we wanted to make sure the POS ngram model (an unknown entity) was working properly by applying topic adaptation (a known entity). Secondly, we wanted to create a topic model that is suitable for combination with the style model. After finding that topic adaptation can indeed work in the POS ngram model, Section 4.5 will describe our approach to style adaptation. We will also include a pilot experiment of combining topic and style adaptations. Finally, Section 4.6 will describe related work and Section 4.7 will summarize our findings with style adaptation and POS modeling.

4.1 Part-of-speech Tagging the Corpora

We first tagged our corpora using the Stanford maxent tagger (Toutanova et al., 2003), which was trained on Penn Treebank (Marcus et al., 1993). We selected a small sample of documents for evaluating the quality of the tagger — one document from each of the corpora with long documents and 2–3 documents from each of the corpora with short documents. The documents were tagged using the Stanford tagger and we marked each tag as acceptable or unacceptable. For example, "yes" as a single-word utterance might be tagged as a singular noun (NN), which is an unacceptable tagging.[1] In contrast, certain errors were more reasonable, such as mistagging a gerund (VBG) as a singular noun (NN). We labeled these taggings as acceptable, because a part of speech model will predict VBG and NN in reasonably similar contexts (i.e., the distinction between gerund and singular noun is not as important

[1] The most appropriate tag seems to be UH (interjections, backchannels, etc).

for our task as the difference between other tags). These sorts of errors were generally uncommon, but we labeled them as acceptable taggings and focused on more problematic mistaggings.[2]

The off-the-shelf tagger produced accuracies ranging from 92% to 99% depending on the corpus. In general, the tagger performed well on written or formal text (Slate, AAC Email, Micase) and poorly on spoken language. The discrepancy is due to the training data of the tagger — Penn Treebank is written text, whereas the spoken corpora often contain sentence fragments and make extensive use of ellipsis.

After our initial evaluation of the tagger, we classified all sources of error. The vast majority of the errors were due to backchannels (e.g., "okay", "mhm"), single-word utterances (e.g., "yes"), interjections (e.g., "gosh!"), and contractions (e.g., "I've", "woulda").

We addressed contractions by building a list of contractions and their expanded forms, then sending the expanded form to the tagger and restoring the original after tagging. For example, "I've" was expanded to "I have" by our preprocessing code, then (hopefully) tagged as "I/PRP have/VBP", then the postprocessing code changed it to "I've/PRP+VBP". Similarly, "woulda" was replaced with "would have" in the preprocessor, then (potentially) tagged as "would/MD have/VB", then postproccessed to "woulda/MD+VB". Although our concatenation of tags is not standard for Penn Treebank, we felt that it was necessary. The Brown Corpus also made this decision to concatenate tags for contractions and similar lexemes.

We constructed the lists of contractions by examining the 200 most common words with apostrophes and the 200 most common words ending in "a" from our corpora. For the most part, we only added rules for contractions that were mistagged.

[2] Note however, that our classification of acceptable or not marks a distinct difference from traditional evaluation of part of speech taggers.

The contractions we expanded were of the following forms: "X've", "X'll", "X're", and "X'd". Contractions of the form "Xn't", "X's", and "I'm" were tagged correctly so we didn't process them specially. We additionally built a list of words and expansions such as "woulda" based on our corpus analysis.

The remaining category of problems for the tagger comprised mostly words which should be tagged "UH". The "UH" tag is meant for backchannels as well as interjections and some dialogue acts (e.g., "yes") We categorized these words to try and understand the problems. The first category of words contained interjections which could also be considered dialogue act signals (e.g., "gosh!", "wow!"). The second category was used in response to a question (e.g., "yes", "no", "alright"). The third category contained ambiguous backchannels (e.g., "yeah", "mhm", "sure") — unambiguous backchannels had already been removed in our corpus cleanup. The last category were words that could not be used as interjections, but could be treated as dialogue act signals (e.g., "hello", "thanks", "hey", "oops"). These categories can all be modeled reasonably using the "UH" tag.

We collected statistics on all words that were used as single-word utterances, and surveyed the most frequent 200. Although these words are used in other contexts (not just as a single-word utterance), for simplicity we only constructed the list from single-word utterances. When we built rules to tag them as "UH" in the postprocessor, we allowed the rule to apply to the first word of any utterance, rather than purely single-word utterances.

The result of our preprocessing and postprocessing rules raised the minimum tagging accuracy from 92% to 98% on the sample documents. The remaining errors in tagging were beyond the scope of simple preproccessing and postprocessing, such situations that require better domain knowledge. In addition to these efforts, we later realized that there were discrepancies between our tokenizer and the Penn Treebank tokenization scheme. We addressed this issue by additional processing

that kept track of how to piece the Penn Treebank tokenization back into our original tokenization.

We have adapted an existing part of speech tagger to our corpora in a simple fashion, but we would like to note some research in the field of domain adaptation. Daumé (2007) evaluates several methods of domain adaptation along for several tasks, where a small amount of annotated data is available in the target domain. This is roughly similar to the small amount of corrections we performed on specific documents, but using more data in the target domain. In contrast, Blitzer et al. (2006) develop domain adaptations that use completely unlabeled data in the target domain and leverage unsupervised learning that focuses on the similarities and differences from the source domain. McClosky (2010) performs domain adaptation for syntactic parsing using a mixture of source domains (similar to our mixtures of topic and style). In the future, we may be able to leverage the body of work in domain adaptation to improve the baseline part of speech tagger for more reliable taggings with little additional annotation effort.

4.2 Proof-of-concept for Style Adaptation

We performed two proof-of-concept analyses before setting out to build the part-of-speech model. Firstly, we measured several words and POS distributions on some corpora and made binary comparisons between the distributions. This analysis is presented in more detail in Appendix C while Section 4.2.1 summarizes the important points. Secondly, we measured the perplexity of each corpus using POS transition probabilities from each other corpus. We extended the analysis by applying these models to text classification. Section 4.2.2 summarizes our efforts and findings in this objective analysis.

4.2.1 Subjective Analysis

In our subjective analysis, we compared a small collection of 33 research emails contained about 8,800 words with a corpus of 4 research papers containing 15,500 words. We sought to control for topical variations by sampling only descriptive emails and papers about our research. The example at the beginning of this chapter is taken from these corpora. We applied the Stanford maxent tagger without any modifications[3] to these corpora and measured several distributions from each corpus.

We measured and compared several distributions from each corpus: word unigrams, POS unigrams, POS bigrams, and coarse-grained POS unigrams/bigrams (i.e., using a more general tagset). The more general POS classes simplified the noun tags to a single tag, as well as the verbs and adjectives to their own classes. For each event in a distribution, we measured the ratio of its probability in one corpus compared to the other. This gives a sense of how much more likely events (e.g., words, tags) are in each corpus. We also computed a normalized difference between the two probabilities, which is somewhat analogous to inverse topic frequency. We focused our analysis on frequent words with a high normalized difference score.

We found that our analyses were able to identify several characteristics of the two styles. The word unigram distributions showed some stylistic indicators — words such as "you" or "I" are much more common in emails than technical articles. Also, descriptive words were more common in technical articles. The part of speech unigram comparison yielded many useful indicators — pronouns (both personal and Wh-pronouns) were more likely in emails. Parenthetical expressions were more likely in emails (the tagger has special tags for right and left brackets). Symbols were also much more common in emails, most likely due to code and shell

[3] This work was performed before developing out preprocessing and postprocessing for the tagger.

snippets. Technical articles had a few distinguishing tags — foreign words (i.e., Latin) were more common as well as comparative adverbs. The effects of the POS unigram distributions unfortunately obscured some of the POS bigram analysis. We focused on the coarse-grained POS bigrams to help reduce the impact of POS unigram differences. Technical articles showed a preference for modified nouns — both noun-noun and adjective-noun expressions were more common. Phrasal verbs were much more likely in emails (e.g., "write up", "figure out")

The subjective analysis demonstrated that we could observe several stylistic indicators solely from comparison of part of speech distributions. Although some trends were observed in word unigrams, POS unigrams also largely captured the same trends. For more detailed results, see Appendix C or view them online at http://www.cis.udel.edu/~trnka/work/style/.

4.2.2 Objective Analysis

The second proof-of-concept analysis we performed involved comparing the sequences of part of speech tags between the various corpora. Whereas the previous section analyses the various distributions separately, this analysis combines the transition probabilities into a trigram backoff model, producing the probability of generating a part of speech tag given the previous two tags. This section is effectively testing $P(tag \mid tag_{-1}, tag_{-2}, s)$ from Equation 4.1 without actually implementing the style adaptations.

The first step in this analysis is is to compare the transition probability models. We used perplexity for this, which is inversely proportional to the probability of generating the tag sequence of one corpus from the tagged data in another corpus. Cross-validation is used when the training and testing corpora are the same, to avoid misleadingly low perplexities. Slate was left out of the analysis for time reasons. Unlike word-based perplexity, the numbers are significantly smaller, because there

are vastly fewer tags than words. The resulting perplexities from this analysis are shown in Table 4.1.

	Training corpus					
Testing corpus	**AAC**	**SB**	**Call.**	**Char.**	**MI**	**SWB**
AAC Emails	12.5	17.3	17.2	17.1	16.9	19.2
Santa Barbara	14.6	10.3	12.6	11.9	11.0	11.4
Callhome	14.8	11.5	10.8	11.0	10.6	10.0
Charlotte	12.3	10.8	10.7	9.5	10.5	9.9
Micase	15.8	12.7	13.9	13.7	10.8	11.4
Switchboard	14.3	11.8	11.1	11.1	10.3	9.4

Table 4.1: Perplexity of each corpus with respect to each other corpus, using only transition probabilities. Columns hold the training data constant and rows hold the testing data constant. For example, a transition probability model trained on Switchboard yields perplexity 11.4 when testing on Micase.

We will start by analyzing trends within each column. This analysis is treating the testing data as a variable while the quality/sparseness of the transition probability model (i.e., the result of the training data) is held constant. At a quick glance, the various columns roughly have a range of values corresponding to the size of the training data (e.g., the AAC column values are relatively high). Secondly, the perplexity reflects the grammatical similarity of the corpora. The cells in which training and testing data are from the same corpus typically display the lowest perplexity in each column (i.e., highest probability).

Looking at the overall table, the corpora fall into three groups: AAC Email, Micase, and the rest of the corpora. This reflects the style of the corpora: AAC Email is written and thus has different characteristics than the other corpora. Micase is academic, and the language is also very unique. The other corpora are predominantly colloquial spoken corpora.

The second part of the analysis was to approximate style adaptation by performing text classification — classifying the documents by corpora using the transition probability model. When using purely the transition probability model, we achieved a classification accuracy of 96.1%. In perspective, a predict-majority baseline achieved 88.1% accuracy, as Switchboard contains many more documents than the other corpora.

In summary, we feel that part of speech transition probabilities can identify many of the differences between different styles. They also show promise in probabilistically differentiating the corpora. The actual style-adapted language model (Section 4.5) will build on the work here in text classification.

4.3 Baseline Part-of-speech Modeling

In contrast to a word ngram model, a part of speech (POS) ngram model does not check the previous words, but rather the part of speech of the previous words. A POS trigram model can be written as

$$P(w \mid h) = \sum_{tag \in POS(w)} P(tag \mid tag_{-1}, tag_{-2}) * P(w \mid tag) \tag{4.2}$$

where $POS(w)$ is the set of possible tags for word w, tag_{-1} is the POS tag of the previous word, and tag_{-2} is the tag of word w_{-2}. The first component $P(tag \mid tag_{-1}, tag_{-2})$ is the transition probability or prior probability. The second component $P(w \mid tag)$ is the emission probability or posterior probability. This part of speech model is very similar to existing models (e.g., (Brants, 2000)).

The relevant questions for creating such a language model are:

- where do tag_{-1} and tag_{-2} come from?
- how is $P(tag \mid tag_{-1}, tag_{-2})$ computed from frequencies?
- how is $P(w \mid tag)$ computed from frequencies?

The set $POS(w)$ is the set of all POS tags and we let the transition and emission probabilities decide how much to affect the overall model.

Under the assumption that we can obtain the tags of the previous two words, we will compute $P(tag \mid tag_{-1}, tag_{-2})$ as a trigram model with backoff. In other words, we will compute it like our word ngram model (Equation 3.6), but using POS tags instead of words. We use a special start-of-sentence symbol to condition the model on the beginning of the sentence, similar to our word ngram model.

The emission probability model is computed in a simple manner without any backoff. We compute the emission probability as $f(w, tag)/f(tag)$, and then discount it slightly using the smoothing method from Section 3.1.3. Note that backing off to a word unigram model is not normally necessary because all tags are considered with non-zero probability — the entire vocabulary of the word unigram model is accounted for in the POS model.

The remainder of this section is organized as follows. First, we will discuss the Viterbi algorithm to find the previous two tags in Section 4.3.1. This will also be used to find the tagging of a sentence after it is completed, when needed for style adaptation or cache modeling. Section 4.3.2 will address the problem of tagging unknown words — even if the user types a new word, we need to know its tag to be able to predict subsequent tags. We will briefly discuss runtime optimization in Section 4.3.3. Section 4.3.4 will briefly evaluate whether we really need to consider all possible taggings of the subsequent word to generate useful predictions. Finally, we will compare the POS ngram model to our baseline word ngram model in Section 4.3.5.

4.3.1 The Uncertain Context of POS: Viterbi Algorithm

We require the tags of the previous two words of context for the language model. The most natural solution is to use some sort of part of speech tagging algorithm. We will use the Viterbi algorithm for Markov model tagging.

The Viterbi algorithm is a dynamic programming approach for finding the optimal sequence of tags for a sequence of words. It maintains a list of candidate taggings for the sentence and stores the probability of each candidate tagging:

$$P(t_1, \ldots, t_n) \approx \prod_{i=1}^{n} P(t_i \mid t_{i-2}, t_{i-1}) * P(w_i \mid t_i) \tag{4.3}$$

The Viterbi algorithm stores the optimal tagging of words ending in each possible tag. For example, the previous sentence ends with "tag". Viterbi processing of the sentence would find the optimal tagging ending with "tag/NN" and also the optimal tagging ending with "tag/VB". It also stores the optimal tagging ending in "tag/NNS", even though the emission probability $P(tag \mid NNS)$ is zero or almost zero. Each of these optimal candidate taggings recursively depends on the optimal taggings for the previous word and so on. Although the process can be thought of recursively, instead it proceeds iteratively from left to right and updates the table after each word.

The iterative nature of the algorithm allows us to update the Viterbi table as words are typed, maintaining T candidate taggings of the (partial) sentence, where T is the number of POS tags encountered in training. The Viterbi algorithm doesn't commit to any particular tagging until the end of the sentence — although there may be a clear best tagging so far, the tagging may change based on subsequent words. Rather than simply pick the best tagging so far for word prediction, we allow them all to affect the model in accordance with the probability the Viterbi algorithm assigns them:

$$P(w \mid h) = \sum_{(tag_1, \ldots, tag_{-2}, tag_{-1}) \in candidates} P(tag_1, \ldots, tag_{-2}, tag_{-1}) * \sum_{tag \in POS(w)} P(tag \mid tag_{-2}, tag_{-1}) * P(w \mid tag) \tag{4.4}$$

where *candidates* is a set containing the optimal taggings of the partial sentence ending in w_{-1} from the Viterbi algorithm and $P(tag_1, \ldots, tag_{-2}, tag_{-1})$ is the probability of each tagging, as computed in Equation 4.3. One important note is that none of the probabilities can be zero if we hope to handle unknown words at all. In the case where smoothing sets $P(w \mid t)$ to some small probability when w is an unknown word, the Viterbi algorithm as presented will assign unknown words to tags based on the context in which they appear.

We will briefly discuss some approximations we made in Section 4.3.1.1 and then evaluate the effectiveness of the Viterbi algorithm for our task in Section 4.3.1.2.

4.3.1.1 Approximations

We found that running tests with a full Viterbi algorithm was much too slow (e.g., 3 hours and 45 minutes for the smallest corpus). Therefore we pruned the candidate lists, transforming it into a beam search. We set a simple probability threshold for candidates to be retained. The table below shows the speed of prediction on a small corpus with no threshold as well as two other thresholds, showing a small tradeoff between runtime and keystroke savings. We will use $t = 0.01$ for the majority of our results.

Run type	**Testing time**	**KS (W = 10)**
Viterbi (no pruning)	13,476s	51.731%
Viterbi ($t = 0.001$)	1,050s	51.727%
Viterbi ($t = 0.01$)	569s	51.712%

Table 4.2: Effects of pruning Viterbi candidates.

In general, the multiplication of many probabilities should be implemented as an addition of log-probabilities to avoid numbers too small to represent in standard

hardware. However, we found it easier to use normal probabilities with multiplication, and normalize the probabilities of the candidates to sum to one after every update of Viterbi. For the sake of prudence, we also implemented the algorithm using log-probabilities. For this to work effectively, we converted the threshold for pruning into log-space. We found that the standard implementation degraded keystroke savings slightly and also slowed the runtime of the system by about 20%. When we added the re-normalization step after each word, the keystroke savings was identical to not using log-probabilities, but was still slow. Therefore we decided to continue using our original implementation rather than the standard implementation with log-probabilities.

4.3.1.2 How Good is Viterbi?

The quality of the Viterbi algorithm is important to the effectiveness of the part of speech ngram model. Therefore, we compared the keystroke savings of our POS model under several different settings:

- Viterbi tagging
- greedy tagging — using a beam search with only the top candidate at every step
- cheat-history — looking up the previous two tags from the corpus rather than using a tagger
- cheat-lookahead — how well we'd do if we knew the tag of the next word

The results from Table 4.3 are from the AAC Email corpus. The quality of word prediction when Viterbi is used to tag the history is remarkably close to actually knowing the history. In re-running these tests, we used the Viterbi pruning threshold $t = 0.001$ (Section 4.3.3), so the difference between Viterbi and greedy tagging is slightly understated. However, with a normal pruning threshold $t = 0.01$,

Run type	Keystroke savings (W=5)
greedy tagging	48.188%
Viterbi	48.202%
cheat-history	48.368%
cheat-lookahead	57.853%

Table 4.3: Keystroke savings with Viterbi compared to reference implementations.

the Viterbi and greedy tagging results have identical keystroke savings and Viterbi offers only slightly better tagging accuracy. The difference might be more significant for other corpora, especially those with many out of vocabulary words. Looking back at the theoretical tests, the cheat-lookahead results give some sense of just how much better the transition probabilities might be. From these results, we conclude that our implementation of the Viterbi algorithm is sufficient for our task.

4.3.2 Tagging Unknown Words: Suffix Tagger

The Viterbi algorithm struggles somewhat with unknown words — the emission model $P(w \mid t)$ provides only minimal help in this situation and the algorithm is forced to tag words based only on their context. However, typically by the time the right context of the word is known, it no longer has a significant impact on word prediction — normally the left context is the only available information.

Rather than purely tag unknown words based on the left context, we developed a basic morphological model which we call the *suffix tagger*. The suffix tagger is designed to input a word without any context and output a probability distribution over part of speech tags. However, the suffix tagger considers all possible suffixes, rather than using a stemmer or morphological analyzer — the suffixes learned by the algorithm can be longer or shorted than real suffixes.

We learned tagging rules in a decision-tree-like manner, with the full language model data from training available to learning. The algorithm starts with the null suffix and considers all possible letters to add to the suffix. Letters are added

from *right-to-left* with respect to the words in which they occur. Every possible next letter to add is considered. The distribution of tags with the two suffixes is compared — if adding a letter to the suffix reduces the entropy of the distribution by more than 10%, then the longer suffix is added and processed recursively. We measured frequencies of word forms for the suffix tagger, not word occurrences, to ensure that the results were focused on normal words rather than common words with irregular morphology. The recursion terminates when any of the following are true:

1. the suffix doesn't reduce entropy enough
2. the suffix exceeds a length threshold (we used 5)
3. there are less than 10 words with the suffix

We found that these rules sufficiently minimized over-fitting in the learning process. Figure 4.1 provides some examples of distributions learned by the suffix tagger.

```
-able: JJ (0.882), NN (0.118)
-ally: RB (1.000)
-ance: NN (0.818), NNP (0.182)
-ding: VBG (0.806), NN (0.194)
-ence: NN (0.900), VB (0.100)
-ents: NNS (0.929), VBZ (0.071)
-ible: JJ (0.909), NNP (0.091)
-king: VBG (0.812), NN (0.188)
-lity: NN (0.900), NNP (0.100)
-ning: VBG (0.722), NN (0.278)
-ties: NNS (1.000)

-ble: JJ (0.839), NN (0.097), RB (0.032), NNP (0.032)
-ing: VBG (0.786), NN (0.170), JJ (0.028), VBP (0.005), ‘‘+NNP (0.004), ...
-lly: RB (0.920), NNP (0.040), JJ (0.040)
```

Figure 4.1: Example rules learned by the suffix tagger.

The suffix tagger can be applied to any word, and returns the tag distribution associated with the most specific suffix possible. In the Viterbi algorithm, we need

the probability $P(w \mid t)$ but the suffix tagger instead provides us $P(t \mid w)$ for a particular word. We can convert from one to the other using Bayes' rule:

$$P(w \mid t) = \frac{P(t \mid w) * P(w)}{P(t)} \tag{4.5}$$

In the Viterbi process, these probabilities are normalized anyway, and the word probability $P(w)$ is an invariant — therefore we omit it. In the process of normalizing the distributions for known words $P(w \mid t)$ we feel that the majority of the effects of $P(t)$ are negated, so we also omit that term. We simply put $P(t \mid w)$ into Viterbi and let the normalization process deal with any conversions.

Table 4.4 shows a small-scale evaluation of the suffix tagger on AAC Emails. Although the improvement is minor, the suffix tagger has many other uses and the benefit is probably more important in an out-of-domain test.

Run type	**Keystroke savings**	**Tagging accuracy**
Without suffix tagger	48.188%	89.88%
With suffix tagger	48.217% (+0.029)	91.53% (+1.65)

Table 4.4: Improvement from the suffix tagger in the Viterbi algorithm.

The suffix tagger is similar to the suffix trees from TreeTagger (Schmid, 1994), which is a decision-tree tagger. Brants (2000) also uses a similar approach, but allows all suffixes to affect the model, rather than only the most specific one. Charniak et al. (1993) suggest an approach based on a stemmer, but we prefer learning the relevant suffixes automatically instead. Cucerzan and Yarowsky (2000) include similar suffix information, but also use information about the distribution of inflections and derivations of a given word — for example, a new word "bananadoor" may not be tagged well based on suffix alone, but if we know about other forms (e.g., "bananadoors"), we can produce better guesses of the tag. Miller et al. (2007) demonstrate that this information can be beneficial in adapting part of speech taggers to new domains (specifically biomedical texts).

4.3.2.1 When Suffixes Fail: Proper Nouns and Symbols

The major failing of the suffix tagger is that suffix alone doesn't contain enough information for tagging. Proper nouns and the associated tags, for example, are partially indicated by the first letter.

We tried three different algorithms to account for proper nouns. Firstly, we tried to account for the differences with simple substitution rules. In training the model, we grouped the proper nouns with the non-proper nouns (i.e., we replaced NNP with NN in any tags). Then in applying the model, if the word began with an uppercase letter, we replaced NN with NNP in any tags suggested by the model. This approach decreased keystroke savings, because the endings of the proper nouns need to be treated as a separate category compared to non-proper nouns.

The second approach was to learn a distribution based on whether the first letter was uppercase or not. For example, the distribution for uppercase letters should have high probabilities for NNP, NNPS, and NNP+POS, whereas the lowercase distribution should have high probabilities for all other tags. This distribution was multiplied with the distribution based on the suffix and then re-normalized. Unfortunately, this approach also decreased keystroke savings. The problem was that the prefix distributions would essentially "outvote" the suffix distributions in some cases, causing lowercase words to be more often mis-tagged as the overriding category for lowercase words — nouns.

The final approach learned a separate tree for uppercase words and non-uppercase words. This addressed the issues in the previous approach. In a test on Switchboard, the prefix-split trees did not affect keystroke savings but raised tagging accuracy from 95.59% to 95.63%. This approach is also used by TnT (Brants, 2000). We omitted the prefix modifications from most of our evaluations due to only minor benefits, but we may improve the method in the future by also separating out symbols and numbers (i.e., first classifying words into groups that share orthographic

rules).

4.3.3 Runtime Optimization

Although we implemented several runtime optimizations involving memory-runtime tradeoffs, this section will briefly highlight a tradeoff between keystroke savings and runtime, yielding a major decrease in runtime (67%) for a minor decrease in keystroke savings (0.02%).[4]

The process of generating predictions in a part of speech model is quite different than a word ngram model. In a word ngram model, the backoff process allows us to first add predictions or completions from the trigram model, then if we need more words, we can check the bigram distribution, then the unigram distribution. In this way, we can avoid processing large sets of words. Unfortunately, the POS ngram model is significantly different — words aren't added in a backoff process at all. Instead, predictions are generated by iterating over the set of possible tags for the next word (and their associated transition probabilities). Each of the tags produces a list of candidate words, up to the size of the prediction window.

The optimization first requires that we sort the tags by transition probability and process them in that order. We seek to find a threshold on the transition probability and skip processing any tags below the threshold. The threshold is chosen dynamically — the first tag which produces a full prediction window worth of words will set a threshold based on the last element in the list. The fundamental basis of our optimization is that most words only have high emission probability for one tag. If we stretch the assumption to say that words can only occur under one tag, then no word less probable than the last element in the first full list can affect the predictions.

[4] These numbers are specifically for the AAC Email corpus, which we used for quick testing.

The first implementation set the threshold to the emission probability only — if any transition probability is lower than that, no word from that tag can ever be placed above the word that determined the threshold (and therefore it can't appear in the list). This sped up processing significantly (over 50% runtime reduction), but eroded keystroke savings more than we'd like (about 0.4%). The second implementation set the threshold to the transition times emission for the lowest word in the first full list of candidates for a given tag. This also allows us to exit the emission probability loop early, and resulted in our overall 67% reduction in runtime with 0.02% reduction in keystroke savings. The minor loss in keystroke savings is due to non-zero emission probabilities for the same word in multiple tags.

4.3.4 Do We Really Need to Iterate Over All Tags to Generate Predictions?

We were curious about the source of our keystroke savings with respect to the various tags, specifically with respect to the lists sorted by transition probability. Therefore, we decided to evaluate the possibility of using only the most likely tag to generate predictions, rather than the normal method of allowing all tags to affect predictions (though pruned somewhat). We found that generating predictions from only the most likely tag produced only 19.9% keystroke savings compared to 51.9% keystroke savings in a test on AAC Emails. This echoes the findings of comparing Viterbi to looking up the tag of the next word — the POS model is somewhat uncertain about the tag of the next word, and therefore needs many tags to generate predictions.

4.3.5 Comparison to Word Ngram Model

Table 4.5 compares the POS ngram model and the word ngram model across corpora with domain variations. The first row of numbers in each group is the

POS ngram model and the second row is the word ngram model. The vocabulary/practical and theoretical limits of word prediction are also shown below. The results on the AAC Email corpus are slightly out of date (e.g., the in-domain results are about a percent higher in more recent tests), but the trends haven't changed with updated code.

In general, these results show a clear trend that the word ngram model is better for word prediction than the part of speech ngram model. The notable trend is that the word ngram model is generally more sparse, and takes better advantage of large amounts of data — the gap between the two models increases with more data. We expect that the poor performance of the part of speech model is due to collocations, multiword expressions, and similar phenomena. For example, in predicting "United S-", the word ngram model has a much stronger source of context ("United") whereas the POS ngram model only knows that the previous word is tagged NNP — the second word could just as easily be a person's last name in the model. The tagging accuracy isn't exactly the problem — it is normally over 90% (although in this case, it's not accuracy but rather agreement with another tagger).

We analyzed the differences between the models in more detail. Our first analysis compares the keystroke savings of both methods by part of speech tag — we measure keystroke savings over singular nouns (NN) separately from prepositions and subordinating conjunctions (IN) and so on. An excerpt of this analysis is shown in Table 4.6. The improvement (or degradation) in keystroke savings of the word ngram model compared to the POS ngram model is shown in parentheses. The word ngram model is much better at predicting many categories of words (reflecting it's overall success), but the part of speech model is better about some other categories. The word ngram model is much better at predicting proper nouns (NNP), cardinal numbers (CD), and certain conjunctions (PRP+VBP). Despite varying differences

	Domain testing		
Corpus	In-domain	Out-of-domain	Mixed-domain
AAC Emails (POS/Viterbi)	47.017%	46.384%	**48.090%**
Baseline (word trigram)	49.467%	51.945%	**53.993%**
practical max	67.794%	74.165%	76.176%
theoretical max	79.928%	79.928%	79.928%
Santa Barbara (POS/Viterbi)	44.583%	44.875%	**46.063%**
Baseline (word trigram)	46.807%	50.364%	**50.698%**
practical max	68.004%	73.187%	73.491%
theoretical max	74.998%	74.998%	74.998%
Callhome (POS/Viterbi)	47.133%	49.082%	**50.098%**
Baseline (word trigram)	49.033%	54.936%	**55.006%**
practical max	67.899%	75.512%	75.530%
theoretical max	76.711%	76.711%	76.711%
Charlotte (POS/Viterbi)	49.957%	49.520%	**50.589%**
Baseline (word trigram)	52.881%	55.485%	**56.005%**
practical max	72.440%	77.229%	77.404%
theoretical max	78.330%	78.330%	78.330%
Micase (POS/Viterbi)	48.572%	48.000%	**49.726%**
Baseline (word trigram)	51.581%	52.655%	**53.779%**
practical max	72.462%	76.320%	77.050%
theoretical max	79.679%	79.679%	79.679%
Switchboard (POS/Viterbi)	**53.427%**	48.967%	52.555%
Baseline (word trigram)*	**58.816%**	55.145%	58.584%
practical max	77.581%	77.878%	78.115%
theoretical max	78.349%	78.349%	78.349%
Slate (POS/Viterbi)	**49.705%**	41.292%	49.127%
Baseline (word trigram)*	**54.323%**	43.919%	54.158%
practical max	79.448%	71.306%	79.583%
theoretical max	81.961%	81.961%	81.961%

Table 4.5: Baseline (word trigram) compared to Viterbi/POS results. These results measure keystroke savings on all words (not just fringe) and do not include dictionary backoff. Domain-varied testing of several corpora at window size 5. Highest value per row of actual results is shown in bold.

between the two models, NN is the most common tag, so the word ngram model produces better keystroke savings overall.

Tag	POS model	Word model
NN	45.150%	47.275% (+2.12)
IN	51.608%	54.323% (+2.72)
NNS	44.021%	46.142% (+2.12)
RB	52.810%	55.422% (+2.61)
JJ	41.809%	43.620% (+1.81)
DT	58.229%	57.166% (-1.06)
VB	48.872%	47.484% (-1.39)
PRP	44.732%	46.500% (+1.77)
NNP	43.835%	47.406% (+3.57)
VBP	54.357%	56.752% (+2.40)
VBG	46.732%	45.514% (-1.22)
CC	62.641%	56.688% (-5.95)
VBD	44.795%	45.096% (+0.30)
VBN	34.264%	34.668% (+0.40)
VBZ	42.667%	39.937% (-2.73)
TO	61.714%	59.045% (-2.67)
MD	64.866%	65.932% (+1.07)
CD	21.982%	26.847% (+4.87)
PRP+VBP	54.995%	59.942% (+4.95)

Table 4.6: Keystroke savings for words in different part of speech tags using the POS and word ngram models.

We analyzed the results one one step further and found more specific examples in which each model was better. Figures 4.2 and 4.3 show examples of words in which each model performs better. The figures also show the contexts in which each method performs better. For example, Figure 4.2 shows one of the reasons the part of speech model predicts better on coordination conjunctions (CC) — it predicts "and" better. The figure shows that the POS model achieves nearly 70% keystroke savings on "and" — the maximum is 75% if we predict the word before any letters have been typed (i.e., 1 keystroke required to produce "and" and a

space). Compared to the word ngram model, the POS model saves 257 keys or 8.5% keystroke savings using the word model as a baseline. The lines below the summary of the word "and" describe the benefit of the POS model in various contexts. For instance, when "and" is the first word in a sentence, the POS ngram model saves 34 keys compared to the word ngram model. The POS ngram model reduces the required keystrokes by 50% relative to the word ngram model. We can deduce that the POS ngram model predicts "and" before the 'a' has been typed whereas the word ngram model requires the first letter to complete "and".

The part of speech model increases keystroke savings substantially when "and" is used and the first word in a sentence. The POS model reduces 34 of the total 257 keystrokes in this context. The remaining contexts all each save two keystrokes or less, because they are less frequent in the testing data. However, in three of the examples, "and" follows an NNP or NNPS. In these situations, the POS ngram model has a significant advantage over the word model — the word model probably didn't encounter "AAC Keys and" or "Dave and" in the training data, whereas the training data probably contains instances of "NNP CC" and "NNPS CC".

Figure 4.2 shows some examples in which the word ngram model is better. Many of the examples of "you" occur are after "of". This is one of many situations in which a word ngram model has a strong advantage — "of" is common enough so that data sparseness is not an issue. Additionally, "of" displays different characteristics from the set of prepositions and subordinating conjunctions (IN) as a whole. The part of speech model does not account for the differences between "of" and the set of IN words, and therefore might predict determiners and other words before "you". The examples of "that" are also similar — several of the contexts are specific uses of "that" following a VBP or VB, but the sequence of words isn't appropriate for all verbs. Effectively, the word ngram model can roughly handle the different

subcategorization frames of verbs and only predict words appropriate for the first argument.

```
and: 69.914% (911 / 3028 keys) [8.487% (-257 keys) over word ngram]
   [and]: 34 words, 50% savings over word ngram (-34 keys)
   infrared stuff [and]: 2 words, 50% savings over word ngram (-2 keys)
   Cerebral Palsy [and]: 2 words, 50% savings over word ngram (-2 keys)
   AAC Keys [and]: 2 words, 50% savings over word ngram (-2 keys)
   Dave [and]: 3 words, 40% savings over word ngram (-2 keys)
a: 40.753% (897 / 1514 keys) [6.209% (-94 keys) over word ngram]
   after [a]: 2 words, 50% savings over word ngram (-2 keys)
   if [a]: 2 words, 50% savings over word ngram (-2 keys)
   to get [a]: 2 words, 50% savings over word ngram (-2 keys)
   to do [a]: 4 words, 33% savings over word ngram (-2 keys)
   click on [a]: 1 words, 50% savings over word ngram (-1 keys)
to: 62.071% (1073 / 2829 keys) [2.651% (-75 keys) over word ngram]
   I got [to]: 2 words, 67% savings over word ngram (-4 keys)
   feel free [to]: 3 words, 50% savings over word ngram (-3 keys)
   that page [to]: 2 words, 50% savings over word ngram (-2 keys)
   will have [to]: 2 words, 50% savings over word ngram (-2 keys)
   not only [to]: 2 words, 50% savings over word ngram (-2 keys)
```

Figure 4.2: Examples by word where the POS ngram model is better than the word ngram model.

Fazly (2002) and Fazly and Hirst (2003) have had more success in applying a part of speech model. The main difference is that they are using BNC, which uses a more specific tagset and contains mostly written data. They report 51% keystroke savings with 5.6 million words of training and 53% with 81 million words of training. Copestake (1997) applied a part of speech ngram model to AAC and found that it improved keystroke savings over a unigram model. They initially used Penn Treebank for training the transition and emission probabilities, but found that it was necessary to train the transition probability model on a small sample of in-domain texts. Additionally, they found it beneficial to treat the most frequent 50 words in the data as their own tag (e.g., "'the" would be tagged as "THE" rather

```
you: 62.610% (510 / 1364 keys) [7.991% (-109 keys) over POS ngram]
   if [you]: 24 words, 50% savings over POS ngram (-24 keys)
   some of [you]: 6 words, 50% savings over POS ngram (-6 keys)
   any of [you]: 4 words, 50% savings over POS ngram (-4 keys)
   most of [you]: 4 words, 50% savings over POS ngram (-4 keys)
   do [you]: 4 words, 50% savings over POS ngram (-4 keys)
that: 64.085% (677 / 1885 keys) [2.971% (-56 keys) over POS ngram]
   I know [that]: 11 words, 50% savings over POS ngram (-11 keys)
   to say [that]: 5 words, 45% savings over POS ngram (-5 keys)
   I'm sure [that]: 4 words, 50% savings over POS ngram (-4 keys)
   for everyone [that]: 3 words, 50% savings over POS ngram (-3 keys)
   to understand [that]: 2 words, 50% savings over POS ngram (-2 keys)
with: 64.757% (363 / 1030 keys) [5.437% (-56 keys) over POS ngram]
   to work [with]: 4 words, 50% savings over POS ngram (-4 keys)
   will work [with]: 3 words, 50% savings over POS ngram (-3 keys)
   to communicate [with]: 3 words, 50% savings over POS ngram (-3 keys)
   the problem [with]: 2 words, 50% savings over POS ngram (-2 keys)
   helping me [with]: 2 words, 50% savings over POS ngram (-2 keys)
```

Figure 4.3: Examples by word where the word ngram model is better than the POS ngram model.

than "DT" in their system). Garay-Vitoria and Julio-Abascal (1997) applied a parser-based prediction system and found that it improved slightly over unigrams (which it should, because it models context). Their parsing model is similar to a part of speech ngram model in that it predicts the part of speech of the subsequent word. We suspect that our part of speech model might be improved by expanding the tag set to model some of the finer-grained distinctions between words and their part of speech. In the future, we would like to experiment with a more detailed tag set like Fazly (2002) or at least treating frequent words as different from their normal tags following Copestake (1997).

In summary, the word ngram model generally offers better predictions than the POS ngram model because the Penn Treebank tagset isn't specific enough to capture all grammatical preferences, let alone any semantic preferences. Especially in the case of common words, the word ngram model can learn the difference between appropriate use of the word's tag compared to the word itself, without suffering sparseness problems. The main advantage of the part of speech model is that it is less sparse — it is able to predict grammatically appropriate words even without having seen the sequence of words in training. In a full system, the POS model might be best suited as a replacement for a unigram word model in a backoff process following (Niesler and Woodland, 1996)— the POS model can generate the same set of vocabulary, but instills a better sense of context than the word unigram model. We would like to implement this in the near future.

Although the part of speech ngram model generally provides worse predictions than the word ngram model, we only intend the model to serve as a baseline to develop the style, cache, and dictionary models. The style model will be evaluated in terms of the improvement over the baseline part of speech model, not in terms of the absolute keystroke savings of the model. Therefore we can develop our style-adapted model independently of any improvements to the baseline part of speech

model.

4.4 Topic-adapted POS Model

We implemented topic adaptation with the part of speech ngram model for two reasons. Firstly, we wanted to ensure that adaptation was possible in the POS model. And secondly, we wanted to create a topic model to combine with the style model. The combination of topic and style is natural when both are represented as adaptations in a part of speech ngram model. The primary difference from our previous work with topic adaptation is that the topic-adapted part of speech model will only affect emission probabilities (i.e., the amount of context affected by adaptation is less), shown below:

$$P(w \mid h) = \sum_{tag \in POS(w)} P(tag \mid tag_{-2}, tag_{-1}) * \sum_{i} P(topic_i \mid h) * P(w \mid tag, topic_i) \tag{4.6}$$

where $P(w \mid tag, topic_i)$ is the emission probability measured from $topic_i$ and $P(topic_i \mid h)$ is the normalized topic relevance score. This formulation glosses over one of the previous decisions in topic adaptation — we combine topic-weighted frequency distributions and convert to probabilities afterwards (i.e., Section 3.1.2).

This section will first discuss the redesign of the previous topic model to function in a part of speech model. Then we will evaluate the topic model in various ways — in addition to previous evaluation methods, we will include tagging accuracy and a per-tag breakdown of the effects on keystroke savings.

4.4.1 Word-based Topic Adaptation Revisited

We developed the model using the smallest corpus for testing (AAC Email). However, this corpus doesn't have defined topics. Instead, we decided to use documents as topics for development. Also, we evaluated results on all words, not just fringe words, because the overall purpose of the POS ngram model is for all words.

We wanted to compute the topic relevance scores using the emission probabilities to mimic our intention of computing stylistic relevance scores using transition probabilities. Therefore, we computed scores for each topic as follows:

$$P(topic_j \mid h) = \prod_i P(w_i \mid t_i, topic_j) \tag{4.7}$$

This model is effectively computing the probability of generating the partial document history by using the emission probability model to generate words from the corresponding tags. The tags are available as the result of the Viterbi algorithm.

Unfortunately, we found that this reduced keystroke savings compared to an unadaptive part of speech ngram model. The score is similar to Naïve Bayes, which previously performed poorly for relevance scoring. Therefore, we instead implemented cosine scoring like our previous topic model. We found that topic adaptation using cosine scoring improved keystroke savings slightly over the baseline — 48.22% vs 48.27%.

Inverse topic frequency (ITF) was very helpful in the topic-adapted word ngram model. ITF was the most beneficial feature for topic adaptation in the POS ngram model — raising topic adaptation results to 48.50%. The relative benefit of ITF compared to basic topic adaptation suggests that by default, cosine scoring does not differentiate enough between relevant and irrelevant topics.

We previously found that stemming was beneficial when documents were used as topics. We applied stemming in the POS ngram code, but found that stemming resulted in only a minor increase in keystroke savings. We double-checked the results by displaying similarity scores whenever they were computed, expecting a working stemmer to increase both the minimum and maximum unnormalized scores due to increased overlap. Both the minimum and maximum scores increased, therefore we believe the stemmer is working as intended.

We also experimented with polarizing the scores — scaling the maximum to one and the minimum to zero before converting to probabilities. However, we didn't

find a benefit in these tests. Smoothing the similarity scores also helped in previous tests, but we didn't find any benefit in the POS model, even though allowing scores to be zero can reduce the vocabulary size slightly. We also re-evaluated the way the sum of frequencies of a distribution is preserved. For simplicity, we initially scaled the normalized relevance scores by the number of topics to attempt to preserve the original total frequencies of each distribution. However, this is different than ensuring that the distributions sum to the same before and after adaptation. We implemented the second approach to compare but found it slightly worse. Some of these differences may be the result of evaluating on all words rather than only fringe words.

4.4.2 Evaluation

Table 4.7 shows the results of topic modeling with the part of speech model on AAC Email data. Topic modeling is beneficial, especially with ITF. Stemming adds a minor benefit to keystroke savings (not shown), and likewise a minor effect on perplexity. This evaluation also shows tagging accuracy, which is agreement with the Stanford tagger. Topic adaptation generally improves tagging accuracy, which is an interesting side benefit. Tagging accuracy increases because topic adaptation skews the emission model used by the Viterbi algorithm. This skew can potentially help limit the Viterbi algorithm to taggings of words that are appropriate for the current topic (like a partial-WSD problem). We also evaluated with automatic clusters, but the fine-grained results were more beneficial.

Table 4.8 shows a similar evaluation but with the Switchboard corpus. This test reproduces many of our previous findings in topic modeling — any kind of topic modeling can offer a benefit over a baseline model, and human topics are better than document-as-topic. The unique difference in this test is that clustering shows better results than the human topics when there are as many clusters as topics. Strangely, the more specific clusters (84 vs 70) show no improvement over

Run type	KS	Tagging acc.	PP
unadapted	48.22%	91.53%	398.2
topic	48.27%	91.59%	395.9
topic + ITF	48.50%	91.60%	385.4
topic + ITF + stem.	48.50%	91.55%	384.7
topic + ITF + clusters	48.44%	91.52%	388.2

Table 4.7: Topic modeling with the POS ngram model on AAC Emails. Evaluation measures keystroke savings, tagging accuracy, and perplexity.

the normal-sized clusters, which is different than the results on word ngram models. Also, the tagging accuracy decreases slightly for most of the topic-adapted results. It's possible that the different taggings are subtle differences.

Run time	KS	Tagging acc.
unadapted	53.391%	95.45%
topic (itf, stemming, documents)	53.907%	95.50%
topic (itf, human topics)	54.321%	95.43%
70 clusters (itf, cluster=1.0 threshold)	54.335%	95.43%
84 clusters (itf, cluster=1.25 threshold)	54.321%	95.44%
59 clusters (itf, cluster=0.75 threshold)	54.329%	95.42%

Table 4.8: Topic modeling on Switchboard in a POS ngram model. Keystroke savings at window size 5 and tagging accuracy are shown.

4.4.2.1 Per-tag Breakdown

We also analyzed the effects of topic modeling by part of speech tag, partially shown in Table 4.9. Topic modeling provides a benefit for most tags, although more for some than others. The biggest benefits seem to be for nouns (NN, NNS, PRP, NNP). The benefit on personal pronouns (PRP) and numbers (CD) is somewhat unusual, but it could be a cache-model-like effect: the token increases its own likelihood via the topic adaptation. The degradation on prepositions and subordinating conjunctions (IN) is unusual. Larger gains may be possible if the topic model were

allowed to affect different tags to different degrees — perhaps a stronger effect on nouns and verbs and a weaker effect on function words would improve the overall combination more.

Tag	Unadapted	Topic (itf)
NN	44.58%	45.20% (+0.62)
IN	51.61%	51.52% (-0.09)
RB	52.52%	52.71% (+0.19)
JJ	41.26%	41.54% (+0.28)
DT	57.65%	57.64% (-0.01)
NNS	43.83%	44.53% (+0.70)
VB	48.99%	49.14% (+0.15)
PRP	44.19%	44.62% (+0.43)
NNP	43.56%	44.05% (+0.49)
VBP	54.79%	54.88% (+0.09)
VBG	46.64%	46.79% (+0.15)
CC	63.09%	63.19% (+0.10)
VBD	44.69%	44.88% (+0.19)
VBN	34.12%	34.18% (+0.06)
VBZ	42.78%	43.27% (+0.49)
TO	61.66%	61.70% (+0.04)
MD	64.46%	64.71% (+0.25)
NN+,	13.36%	13.36% (+0.00)
CD	21.45%	22.56% (+1.11)
PRP+VBP	53.67%	53.86% (+0.19)

Table 4.9: Keystroke savings of topic modeling, analyzed by part of speech tag.

4.4.3 Summary

In summary, we have successfully recreated our topic adaptation methods in the part of speech model. Some of the results are more beneficial than the word model (partly because the POS model leaves more room for improvement). Some of the results show less benefit, which may be attributed to evaluating on all words rather than only fringe words.

4.5 Style-adapted POS Model

We developed the style-adapted part of speech model similarly to the topic-adapted model — instead of weighting by topic, we weight by style. Instead of tuning the emission probabilities, we tune the transition probabilities. The equation form of style adaptation is shown below.

$$P_{style}(w \mid h) = \sum_{s \in styles} P(s \mid h) * \sum_{tag \in POS(w)} P(tag \mid tag_{-1}, tag_{-2}, s) * P(w \mid tag) \tag{4.8}$$

We will start by taking a corpus to be a style and tuning the transition probabilities based on the relevance scores. We will identify style in the same way as the objective analysis in the proof of concept section (Section 4.2.2) — the score of each corpus will be the probability of generating the partial document history. Because each corpus is a style, evaluations will focus on mixed-domain testing, where all available corpora are used for training.

This section will first present some differences in the development of the model, then will iteratively develop the style adaptations based on the results of evaluation and from out work in topic adaptation. Although not all of the problems in style adaptation are identical to topic adaptation, many of the issues are similar and will be addressed with similar solutions.

The most notable difference in developing the style and topic models is that the cache representation for the two models is very different. There are two alternatives: Firstly, we could iteratively train unigram, bigram, and trigram trigram part of speech transition probability models on the cache data. This would be the closest model to the cache of the topic model. The alternative is to simply store the entire tagged document as it is typed and compute probabilities afterwards. We implemented the latter approach, which re-used more existing code from the POS model. The disadvantage of this approach is that decay modeling is more difficult. However, the style of discourse is applicable to the entire document and does not change in

the same way that the topic of discourse shifts. Therefore, we implemented style adaptations without any kind of decay in the cache representation.

The decision to combine frequencies or probabilities is also different in a style-adapted model. For one, the size of each style (a whole corpus) is much larger than the size of each topic (1/70th of a corpus in the case of Switchboard). Additionally, the POS transition probability model uses a much smaller parameter space than the word trigram model, lessening the data sparseness issue. Also, we need to smooth each transition probability model to compute the stylistic relevance scores. For all these reasons, we decided initially to combine probabilities for style adaptation, rather than combining frequencies and smoothing later as with topic adaptation.

We conducted a small-scale pilot test for the first iteration of our style model, using two of the smaller corpora. These two corpora were chosen in part because they have different stylistic features — the AAC Email corpus contains fewer sentence fragments and is less directed than the Santa Barbara texts. The results are shown in Table 4.10. The style model yielded a noticeable benefit on AAC Emails, but at the cost of a noticeable loss of keystroke savings on Santa Barbara (the larger corpus). We investigated the problem by watching the normalized relevance scores

Corpus	**No style, mixed-domain**	**Style, mixed-domain**
AAC Emails	47.755%	48.069% (+0.314)
Santa Barbara	44.714%	44.518% (-0.196)

Table 4.10: Style adaptation pilot test

during simulation. Although sometimes one corpus would get a weight of 0.6-0.8, the scores were normally between 0.45-0.55, meaning that the corpora were given nearly equal weight despite the expectation that the styles were very different. In contrast, the baseline model does not weight the corpora equally — the baseline model effectively weights the corpora in proportion to their sizes (giving about 90%

of the weight to Santa Barbara). Essentially by default the system is weighting AAC Emails higher than the baseline model and Santa Barbara lower, which explains the gain in keystroke savings on one corpus at the expense of the other.

Therefore, we decided to multiply the weights by an analogue of the corpus sizes. We did not want the scores to become overly skewed, so instead of using the raw number of words (or documents) in each corpus, we dampened the sizes somewhat. Table 4.11 show the results — adding in size-weighting reduced the problem, but also reduced the benefit. The normalized relevance scores reflected the size weighting — they were often 0.9 for Santa Barbara and 0.1 for AAC Emails, with minor fluctuations.

Corpus	**No style**	**Style**	**Style with size-weighting**
AAC Emails	47.755%	48.069% (+0.314)	47.768% (+0.013)
Santa Barbara	44.714%	44.518% (-0.196)	44.711% (-0.003)

Table 4.11: Style adaptation pilot test with size weighting

We pursued the issue a little further by comparing style adaptation on corpora of more similar size without size-weighting — Micase and Switchboard (which is about 5 times larger than Micase). Even if the two corpora are of very different sizes, at the least the transition probability models should be well saturated in both corpora. Table 4.12 shows the results — again the style adaptations benefit the smaller corpus more and degrade performance on the larger corpus. Effectively the results reflect the differing biases of the baseline model (size-weighted) and the style model (size is ignored). Although unfortunate, we have not found a way of evaluating style adaptation without some effects from the relative corpus sizes. However, we expect that the effects of corpus size will be less important when testing on a more diverse collection of corpora, using a mixed domain test. We feel that

Similar size corpora		
	not style-adapted	**style-adapted**
Micase	49.120%	49.240% (+0.120)
Switchboard	53.392%	53.296% (-0.096)

Table 4.12: Style adaptation with corpora of larger sizes

the style adaptation itself becomes more important than corpus size in a test with more corpora, based in part on the findings of our corpus studies (Section 2.3).

We decided to perform a full corpus test of style adaptation, shown in Table 4.13, following the expectation that style adaptation would fare better when there are many styles to choose amongst.[5] In a larger mix of corpora, style adaptation is much more beneficial — keystroke savings improves on all corpora to some degree. This demonstrates the ability of the style model to weight grammatical style appropriately, and demonstrates more benefit when there are more styles to choose amongst. Compared to the smaller-scale tests of style thus far, the baseline model in these cases is effectively more biased against the appropriate style (due to an increased amount of out-of-domain data). Although style adaptation is successful in this case, the effects of corpus size are still apparent — larger corpora benefit less because the unadapted baseline model is more biased towards in-domain data for larger corpora. On the other hand, in-domain data for our target population (AAC users) will be smaller than most corpora, and so the larger benefit on smaller corpora is suitable for our system.

Although the style adaptations appropriately tune the transition probabilities to similar styles with many different styles, in the smaller tests (e.g., Table 4.10) we found that style adaptation weighted the corpora somewhat evenly. In other words, the style-adapted model may suffer from the same problem as the topic-adapted

[5] Slate is omitted from the table due to system issues during the evaluation.

Corpus	POS baseline	Style-adapted
AAC Emails	48.090%	48.264% (+0.174)
Santa Barbara	46.063%	46.256% (+0.193)
Callhome	50.098%	50.315% (+0.217)
Charlotte	50.589%	50.738% (+0.149)
Micase	49.726%	49.774% (+0.048)
Switchboard	52.555%	52.623% (+0.068)

Table 4.13: Style adaptation on all corpora but Slate.

language model — cosine scores without any other processing only adapted a little to topics (e.g., Section 4.4). We addressed this problem with topic adaptation in two ways — scaling/polarizing the scores and applying inverse topic frequency.

We first experimented with polarizing or scaling the stylistic relevance scores. This method scales the maximum score to one and the minimum score to zero (or nearly zero). The intuition behind polarizing the scores assumes that the relevance scores yield the correct ranking of styles but the values underestimate the relative benefit of the corpora. For example, in the test with AAC Emails and Santa Barbara without size-weighting, we assume that the higher score is typically assigned to the correct corpus. Indeed this intuition has been shown in our tests of using the relevance scores for style classification in Section 4.2.2 — the relevance scores can be used to classify documents with an accuracy of 96%. The main difference in style adaptation is that the relevance scores are decided on the basis of partial documents rather than complete documents.

Based on the assumption that the relevance scores can correctly rank the relative benefits of corpora, polarization seeks to increase any above-average scores and decrease any below-average scores. We achieve this by linearly scaling the scores such that the maximum is one and the minimum is zero. We also included a small smoothing factor to ensure that no corpora were assigned zero weight. The formula

we applied for adjusting the weight of corpus i is shown below:

$$w_i' = \left(\frac{w_i - min}{max - min} \right) + 0.5 * min \tag{4.9}$$

The maximum weight is scaled to one while the minimum weight is scaled to zero. Then a fraction of the minimum weight is added to all scores to ensure no weights are zero. We evaluated the effects using all corpora.[6] Table 4.14 shows the results compared to the style-adapted baseline (i.e., the improvements shown are only due to polarization). The polarization method is beneficial on all corpora — in some cases, the benefit of polarization is greater than the base benefit of style adaptation, similar to our findings with inverse topic frequency in Section 4.4.

Corpus	**Style-adapted**	**Polarized style-adapted**
AAC Emails	48.264%	48.305% (+0.041)
Santa Barbara	46.261%	46.360% (+0.099)
Callhome	50.321%	50.454% (+0.133)
Charlotte	50.746%	50.875% (+0.129)
Micase	49.788%	49.857% (+0.069)
Switchboard	52.634%	52.753% (+0.119)

Table 4.14: Style adaptation with polarization in a mixed-domain test. The baseline is the basic style-adapted model.

The main disadvantage of polarization is that it lacks intuitive interpretation — we could scale the relevance scores for more or less polarization, but we only have an empirical basis for the tuning rather than a linguistic basis. In contrast, we can achieve a similar effect (higher scores for more relevant styles and lower scores for less relevant styles) by adapting inverse document frequency (IDF) to style adaptation. It may even be the case that polarization and inverse style frequency together can achieve better results — our work with topic adaptation included both scaling and inverse topic frequency.

[6] Slate is omitted due to system issues, but is included in subsequent tests.

We developed *inverse style frequency (ISF)* to mimic inverse document frequency and inverse topic frequency. ISF will focus the relevance score on those transition probabilities that best differentiate between different styles. For example, in a comparison between emails and technical article texts, the tag sequence "VB DT" is equally probable in both styles. However, "VB IN"[7] is much more probable in emails. Therefore "VB IN" should affect the relevance score more and any occurrences of "VB DT" should affect the scores very little.

In contrast to inverse document frequency or inverse topic frequency, the computation of inverse style frequency is not straightforward, because the relevance scores use the transition probability model rather than a pure unigram model. We addressed the problem by computing a different ISF distribution for each transition probability distribution (trigram, bigram, and unigrams). The unigram ISF is shown below:

$$ISF(tag) = log\left(\frac{num\_styles}{SF(tag)}\right) \tag{4.10}$$

where $SF(tag)$ is the style frequency of the tag — the number of styles containing the tag. For the unigram ISF, the computation is identical to IDF or ITF, but using tags. In the case of using corpora as styles, ISF will only vary when one or more corpora do not contain the tag. This is relatively unlikely for most tags, but tags resulting from contractions such as "PRP+VBP" may show zero frequency in formal language.

We use a similar computation for the bigram distribution:

$$ISF(t_1\ t_2) = log\left(\frac{num\_styles}{SF(t_1\ t_2)}\right) \tag{4.11}$$

In contrast to the unigram ISF values, we expect to see much more variety in the bigram distribution. For example, in the small-scale comparison of emails and

[7] A base form verb followed by a preposition or subordinating conjunction (e.g., "say that", "work on")

articles, "VBP PRP" (e.g., "think we") occurred often in emails but never in papers and "NN VBP" occurred often in papers but never in emails. The trigram ISF is similar:

$$ISF(t_1\ t_2\ t_3) = log\left(\frac{num\_styles}{SF(t_1\ t_2\ t_3)}\right) \tag{4.12}$$

In the case of the trigram ISF, we expect that many more tag triples are unique to one corpus or another.

The style relevance scores are the average generation probability of the tag sequence of the partial current document with respect to each style's transition probability model. The transition probability model is a trigram backoff model. The inverse style frequencies are applied at the level of backoff — if the trigram probability is used, it is first multiplied by the trigram ISF score. If the bigram transition probability is used, it is multiplied by the bigram ISF score. If the unigram transition probability is necessary, it is multiplied by the unigram ISF score. The resulting relevance score is much greater than the original, but only the relative scores are important, not the absolute values.

We evaluated inverse style frequency on all corpora (Table 4.15), but found that it degraded keystroke savings compared to the baseline style adaptation. In many cases, style adaptation with ISF is even worse than the unadaptive baseline. We suspect that the loss of keystroke savings reflects the poor match between ISF and large units of style such as corpora. The tag pairs and triples that have non-zero ISF (i.e., tag pairs/triples that do not occur at all in at least one corpus) may be abnormally infrequent, especially in a mixture of overlapping styles.

Inverse style frequency has intuitive appeal, but our implementation of ISF yielded poor results when using corpora as styles. However, the method works well when we use documents as styles in the following section, due to the increased sparseness of documents. In retrospect, ISF and similar methods rely on data sparseness — on zeros in the data. Although we can rely on zero-frequency words for topic

	Baseline	Style	Style (ISF)
AAC Email	49.570%	**49.649%**	49.599%
SBC	46.062%	**46.271%**	45.974%
Callhome	50.095%	**50.335%**	50.089%
Charlotte	50.592%	**50.746%**	50.440%
Micase	49.730%	**49.797%**	49.374%
Switchboard	52.513%	**52.592%**	52.154%
Slate	**49.052%**	48.831%	48.400%

Table 4.15: Style adaptation with inverse style frequency in mixed-domain evaluation. The maximum for each row is bolded. The baseline results differ from Table 4.14 due to minor changes in corpus cleanup and the baseline model.

identification, we can't rely on zero-frequency tags for style identification with large units of style. Instead, with part of speech tags, we need a measure of the variation of the probability of a given tag, tag pair, or tag triple. In the future, we would like to explore alternate methods for capturing an ISF-like value, for example we could inspect the set of probabilities $P(t \mid t_{-1}, t_2, style_i)$ for each tag triple across styles. The variation in the transition probabilities might be measured by the standard deviation of the probabilities.

4.5.1 Document as Style

Previously, we have used corpora as styles. Certainly, each corpus has a somewhat unique style — Slate is a formal written news corpus, AAC Emails are semi-formal email discussions, and Micase contains academic prose. However, there are stylistic variations even within each corpus. Many corpora contain documents from the same people — AAC Emails contains data from only two people, Switchboard has several users which participated in multiple conversations, Slate has recurring writers, and so on. The different users within a corpus tend to have different personal styles of writing. Biber and Conrad (2009) describe this as the difference between register and style — registers are functionally motivated variations in

linguistic features, arising from situational needs whereas styles reflect personal linguistic preferences that serve much lesser (if any) functional purpose. Our research with corpus as style reflects only Biber and Conrad's registers, but if we treat each document as a (potentially) different style, we may be able to both model register-based variations as well as the stylistic preferences of different people. Although in general, we might not need units of style as small and specific as individual documents, most corpora do not have speaker/writer identification labels. Clustering may be possible, but following our findings with clustering and documents in topic adaptation, we would prefer to use documents. Additionally, we would like to have a smaller-scale test to aid in the development of style adaptation, rather than run a full-scale mixed-domain evaluation to test the effects of each idea.

The most notable difference in using documents as styles is that the transition probability models are less saturated and contain varying sizes. However, the variation in document size is more manageable than the variation in corpus size, both due to lesser extremes and also due to the much larger number of documents than corpora. We followed previous findings in topic adaptation with documents to combine transition model frequencies rather than probabilities. Table 4.16 shows the difference between combining probabilities and frequencies — combined probabilities fared worse than the unadaptive baseline model whereas frequencies improved keystroke savings.

In contrast to our findings with corpus-as-style, we found that inverse style frequency was an improvement for document-as-style. The difference in findings is likely due to both the sparseness of documents as well as the large number of documents. The ISF score for corpora effectively only has eight different values — one value per possible style frequency. In contrast, the ISF scores for documents have thousands of possible values, providing a much finer grained distinction between different measures of stylistic salience. Additionally, the sparseness of the documents

(in contrast with corpora) leads to many more zeros in the frequencies for part of speech tags, pairs, and triples across documents, which allows the style frequencies to have more values between the minimum and maximum.

In contrast to inverse style frequency, polarization produces slightly less benefit, although it still provides benefit over the baseline model. The result is somewhat difficult to interpret because polarization lacks the intuitive model of inverse style frequency. We combined the two approaches (polarization and ISF) to better understand the results and also to seek a better method. Unfortunately, we found that polarization did not increase keystroke savings or tagging accuracy over ISF alone, but decreased them very slightly. It could be the case that polarization skews the ISF-based relevance scores too much, leading to improvement on some documents but a greater loss on (partial) documents where relevance scores are unable to properly rank the styles. Regardless of the cause, the difference on this corpus is too minor to analyze. We have come to the conclusion that inverse style frequency is more appropriate for document-as-style whereas polarization is more appropriate for corpus-as-style.

AAC Emails (fine-grained)	**Keystroke savings**	**Tagging accuracy**
baseline (unadapted)	47.017%	89.48%
style/probabilites	46.677%	88.39%
style/frequencies	47.058%	89.46%
style/frequencies+isf	47.083%	89.46%
style/frequencies+polarize	47.077%	89.36%
style/frequencies+polarize+isf	47.081%	89.27%

Table 4.16: Developmental evaluation of document-as-style on AAC Emails.

Our findings with document-as-style suggest that it may be a viable way of modeling variations in style due to both situational differences (e.g., spoken/written,

directed/undirected) as well as personal preferences. In the case of our pilot experiments, we have only demonstrated the utility of document-as-style to model variations within a corpus. Style adaptations in these tests may reflect the different preferences of the two writers in the corpus. We would like to further explore the possibility of modeling style using documents, especially in tests with a variety of corpora.

4.5.2 Evaluation

We evaluated corpus-as-style adaptation with several combinations of frequencies vs probabilities, ISF vs polarization, and size-weighting. Table 4.17 shows the results of the unadapted part of speech ngram model, the baseline style-adapted model, the style-adapted model with polarization, and the same model also with size-weighting. The results with frequencies and ISF are omitted for brevity, and also because those methods were not useful for corpus-as-style. Style adaptation with polarization alone is generally best, but size-weighting improves results on larger corpora (Switchboard and Slate). The effect of size-weighting on the larger corpora is intuitive — the weights for Switchboard and Slate increase by size-weighting, thus increasing keystroke savings by tuning the adaptation more to in-domain data for those corpora. These results suggest that the baseline style relevance scores could be improved further — an ideal relevance score would not need size-weighting to improve results on any corpora. However, the issue could be that stylistic indictors are not encountered early enough in Switchboard and Slate documents, so size weighting may start the scores off more advantageously.

The main question is whether these results are good or not. Perhaps another method of style adaptation could do better, or perhaps not. The following section will explore an approximate upper bound for style adaptation.

	Baseline	Style	Style (pol.)	Style (sw-pol.)
AAC Email	49.570%	49.649%	**49.691%**	49.592%
SBC	46.062%	46.271%	**46.362%**	45.946%
Callhome	50.095%	50.335%	**50.457%**	50.150%
Charlotte	50.592%	50.746%	**50.879%**	50.614%
Micase	49.730%	49.797%	**49.859%**	49.756%
Switchboard	52.513%	52.592%	52.704%	**52.799%**
Slate	49.052%	48.831%	48.994%	**49.170%**

Table 4.17: Final gains of corpus-as-style using mixed-domain evaluation. The maximum for each row is shown in bold.

4.5.2.1 Framing Benefits: How Good is Good?

The results from style adaptation are somewhat underwhelming compared to topic adaptation on fringe words. However, style adaptation provides a much more uniform benefit across corpora when evaluated on all words. But the important question is whether our results are good relative to the possible theoretical benefit of style adaptation.

Therefore, we developed an approximate upper bound for style adaptation.[8] The best of the various style models is the model from the appropriate corpus. Therefore, we will measure the keystroke savings when only the appropriate corpus is used for the transition probabilities. Effectively, this is an evaluation of the system if the relevance scoring perfectly classified every document into the appropriate corpus. This should provide a approximate limit to the potential benefit of style adaptation.

The approximate style limit compared to the baseline model and style adaptation with polarization is shown in Table 4.18. For most corpora (those except Switchboard and Slate), style adaptation is actually better than the style limit.

[8] Many thanks for Vijay Shanker and Oana Tudor for the suggestion!

On Switchboard, the style adaptation achieves 70% of the limit, though the size-weighted results from Table 4.17 are better than the limit. The style model performs worse than the unadaptive baseline on Slate, and even the size-weighted results on Slate are under the limit.

	Baseline	**Style-limit**	**Style-adapted**
AAC Email	49.57%	49.28% (-0.29)	**49.69%** (+0.12)
SBC	46.06%	46.17% (+0.11)	**46.36%** (+0.30)
Callhome	50.10%	50.00% (-0.10)	**50.46%** (+0.36)
Charlotte	50.59%	50.78% (+0.19)	**50.88%** (+0.29)
Micase	49.73%	49.72% (-0.01)	**49.86%** (+0.13)
Switchboard	52.51%	**52.78%** (+0.27)	52.70% (+0.19)
Slate	49.05%	**49.22%** (+0.17)	48.99% (-0.06)

Table 4.18: Approximate limit for style benefit using mixed-domain evaluation, compared to an unadaptive baseline and style adaptation with polarization. The maximum for each row is shown in bold.

The approximate style adaptation limit is clearly not an upper bound for style adaptation. In particular, the limits for AAC Email and Callhome suggest that the transition probability model is far from saturated using only the in-domain data. The results on Callhome suggest that the other similar spoken corpora may be filling in the gaps in the in-domain transition probability model. In contrast, the style of AAC Email is very different from other corpora, but the results suggest either that the transition probabilities are not saturated with in-domain data alone or that the other corpora may allow some adaptation to the two writers in the corpus. The style limits for Switchboard and Slate are more intuitive — there is a substantial amount of in-domain data, and using only in-domain data for the transition model is better than an unweighted mixture of data from all corpora (the baseline).

In summary, style adaptation outperforms the approximate limit for most corpora, especially smaller corpora. The limit is higher than adaptation for larger corpora because of the problems with style adaptation and larger corpora — both

the unadaptive baseline and the style limit are likely assigning higher weight to in-domain data than the style adaptation for larger corpora. The results demonstrate that out-of-domain data is important for style adaptation, both to mitigate any sparseness in the in-domain transition model and also to potentially allow for limited adaptation to varying style within each corpus.

4.5.3 Combining Topic and Style

We designed the topic adaptations of Section 4.4 and the style adaptations of Section 4.5 to work together by affecting different parts of the part of speech ngram model. Style tunes the transition probabilities and topic tunes the emission probabilities. We performed pilot evaluations of combining topic and style and evaluated them in the context of our reasonable gold standard.

The *reasonable gold standard (RGS)* is analogous to the approximate limit of the previous section — it measures an approximate upper boundary on the possible improvement from combining multiple language modeling improvements. To compute the gold standard, we first run each of the individual improvements that we seek to combine. The simulation creates log files that contain the number of keystrokes required to type each word with and without word prediction. After the improvements have all been simulated, the reasonable gold standard analysis is run on the log files, comparing the keystroke savings of each prediction method on a word-by-word basis. 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).

Table 4.19 shows the individual style and topic models as well as the combined model and the reasonable gold standard. The actual combination of topic and style achieves 48.557% keystroke savings compared to the RGS of 48.842% savings. The combination of topic and style clearly improves over topic and style individually (and close to an additive gain of the respective improvements), but falls somewhat short of the potential benefit.

Run type	**Keystroke savings**
unadaptive baseline	48.217%
style (f+isf)	48.284%
topic (itf)	48.499%
style (f+isf) + topic (itf)	48.557%
RGS of style + topic	48.842%

Table 4.19: Pilot study of topic and style combination using AAC Emails. Document-as-style and document-as-topic were used for these tests.

The reasonable gold standard also provides two useful pieces of information for each model. The first is the win percent — the percent of the words for which a given model yields the maximum keystroke savings amongst the input models. The second is the exclusive win percent — the percent of the words that a given model is the only model that produces the maximum savings. The win percent can be used to evaluate the general-performance quality of model in a mixture. The exclusive win percent characterizes the unique contribution of each model, similar to a leave-one-out evaluation.

Table 4.20 provides this additional information. The win percent is relatively uninformative for these models because they all provide quality general-purpose improvements to the language model. In contrast, a model like a cache model would have a much lower win percent. The exclusive win percentages show that the topic adaptation contributes the most to the combination model, whereas the style model contributes benefits less often. The higher exclusive win percent of topic adaptation

is the result of two things — topic adaptation provides more benefit over the baseline than style adaptation (especially for document-as-style) and the differences in keystroke savings are complementary to style adaptation and the baseline. Unfortunately, the exclusive win percent for the unadapted baseline model in non-zero, meaning that topic and style adaptation both degraded keystroke savings in some limited cases. We would prefer to develop adaptive models that never degrade performance relative to an unadaptive baseline.

AAC Emails (fine-grained)	**unadapted**	**style (f+isf)**	**topic (itf)**
keystroke savings	48.217%	48.284%	48.499%
win percent	96.911%	97.207%	98.261%
exclusive win percent	0.141%	0.520%	2.241%

Table 4.20: Analysis of topic and style combination.

The combination of topic and style is beneficial, but style is more beneficial in a mixture of corpora. In future work, we plan to combine topic and style using mixed-domain evaluation. We would also like to further investigate the difference between the keystroke savings of the combined model and the reasonable gold standard.

4.6 Related Work

Biber and Conrad (2009) address variations in language from three different perspectives: register, genre, and style. Their register and style perspectives are similar to our treatment of style — a particular register or style affects linguistic features of the texts such as formality, directedness, etc. The difference between the register and style perspectives is that the register perspective is functionally motivated — the variations in language are a functional result of situational characteristics such as the communication medium (e.g., email, phone, face-to-face). In contrast, the style perspective is less functional; individuals simply have preferences

for various linguistic features. The genre perspective is different than register and style, reflecting the formatting and structure of texts. For example, letters have a general structure — address, salutation, body, closing, and signature. Research articles typically contain a hierarchical document structure with labeled sections.

We are only interested in the register and style perspectives of Biber and Conrad (2009) for this work. We will not address the genre perspective because document structure and formatting may be inconsistent and unreliable in our corpora. What we refer to as "style" subsumes their register and style. In particular, we are primarily interested in modeling regularities in language. Although the presence or absence of functional variations is linguistically interesting, we model variations at the realization level rather than as a generative process stemming from the situational characteristics. The difference between register and style is only relevant in our work in the context of using corpora versus documents as the units of style. When we use corpora, we expect that stylistic preferences are not captured by the model, whereas document-based units are able to capture variation due to linguistic features of the register and also the linguistic preferences of the speaker.

The remainder of the related work can be analyzed in terms of Biber and Conrad's (2009) distinctions — we will first describe work in natural language generation, which predominantly addresses register explicitly by modeling the influence of situational characteristics (e.g., spoken vs written) on linguistic features (e.g., haste) and then the influence of linguistic features on realization (e.g., generation of referring expressions). In contrast, genre classification research focuses on the low-level aspects of style (e.g., sentence length, part of speech distributions, word/character distributions). This work has evolved over time — early work focuses more on the register perspective and includes more grammatical features, whereas newer work tends to focus more on identifying genre via formatting and more basic information. Finally, we will discuss the scarce related work in style adaptation for language

modeling. Language models generally focus on a mix of the register and style perspectives, similar to our work. However, most existing work models style implicitly; very few specific efforts have been taken to develop methods to identify stylistic variations for language modeling.

4.6.1 Style in Natural Language Generation

Hovy (1987) 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 (1993) 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 (1992) 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. (1993) focus on lexicalized style in near-synonyms for machine translation, attempting to preserve the tone of translated text.

4.6.2 Style in Genre Classification for Information Retrieval

Research in Information Retrieval seeks to automatically detect text genre (Karlgren and Cutting, 1994, Michos et al., 1996a, Kessler et al., 1997). 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 (1994) 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% accuracy for 2 classes, 73% accuracy

for 4 classes, and 52% for 10-15 classes. Kessler et al. (1997) 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 are the realizations of facets at various linguistic levels in the document. They 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 (1994) 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. (1996a) 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[9] 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

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

identifiers identify each feature, and in turn how each feature can identify each category of style. Michos et al. (1996b) 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. (2000b) 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. Intuitively, smaller documents can have sampling errors; they contain very few genre signals.

Wolters and Kirsten (1999) 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. (2001) 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. (2000a) 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 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 (2002) 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. (2003) 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.

4.6.3 Style in Author Discrimination

Graham et al. (2005) address the problem of documents with multiple authors — they seek to identify stylistic inconsistencies resulting from author changes within a document. In contrast to genre or author classification, they only seek to identify changes from one author to another rather than additionally identifying each author. They approximate the problem by concatenating Usenet postings and identifying the paragraph boundaries that are legitimate changes in author (about 25% of the boundaries). They compute various statistics of the adjacent paragraphs and train a neural network to decide whether the boundary marks an authorship change. They found that the most useful features were part of speech unigram distributions and punctuation distributions. Distributions of function words were also helpful. The primary difficulty was the small amount of text used for comparison — although features such as character ngrams can be effective in genre classification, they were ineffective in this task due to sparseness. Similarly, traditional style metrics such as word and sentence lengths were overly sparse for their task. They conclude that high-level features are best for the task due to the small amounts of text being compared.

Feiguina and Hirst (2007) and Hirst and Feguina (2007) address a similar problem of authorship discrimination for short texts with different texts. Compared to Graham et al. (2005), they add features derived from partial parses of the input. Primarily, they add bigram distributions of syntactic labels to existing features. Their syntactic labels include not just part of speech but also grammatical constituent labels (e.g., noun phrase). They linearize the partial parse using a depth-first traversal of the sequence of parse trees to build the bigram model. Their model captures not just bigrams of part of speech (e.g., determiner followed by singular noun) but also higher level aspects of grammar (e.g., verb followed by a clause). They found that both syntactic label bigrams and lexical features[10] were useful. Although they combined all of the features with a support vector machine, they found that classification using syntactic label bigrams required less training data to achieve reasonable performance compared to the lexical features. In contrast, our style identification uses only ngrams of part of speech; ngrams of non-terminal syntactic labels are not included in our system. However, they do not compare the benefit of part of speech bigrams with syntactic label bigrams, making it difficult to say whether bigrams including non-terminal labels were useful.

4.6.4 Style in Language Modeling

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). Several researchers use the EM algorithm as the basis for adapting a linear combination of the training data (Clarkson and Robinson, 1997, Gildea and Hofmann, 1999, Iyer and Ostendorf, 1999, Adda et al., 1999). This adaptation is based on the probability that each individual language model assigns to

[10] This includes average word length, average sentence length, measures of vocabulary (e.g., hapax legomena/dislegomena, type/token ratio), function word distributions, and punctuation distributions.

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 (e.g., tuning weights to match word trigram generation probability). These automatic optimization methods predominantly seek to model topic variation, and are successful at reducing perplexity. However, even the implicit modeling of style lacks the information in our part of speech ngram model. For common words, our adaptation may be similar — for example, the tuning of determiners followed by prepositions. The reason the adaptation may be similar is that all possible pairs of determiners and prepositions should have reasonably accurate probability estimates. In contrast, word ngram models are unsuitable for style adaptation for open-class words (e.g., the variation of noun-noun compounds). Word ngram models that rely solely on the EM algorithm for adaptation may tune the language model for specific noun-noun terminology (e.g., "language model") but cannot tune the language model to the stylistic variation of noun-noun compounds in general.

Research in topic modeling that uses only word unigrams to identify topics may also implicitly include limited stylistic adaptation. For one, such methods may account for lexical aspects of style (e.g., words of varying formality). Also, these methods may adapt to style via topic-style correlations (e.g., sports documents may tend to be informal compared to science documents).

Among the researchers that use the EM algorithm to adapt to topic, Clarkson and Robinson (1998) are the only to acknowledge any stylistic adaptation of their language models. However, like other research with the EM algorithm and word ngram models, the stylistic adaptations are only implicit in the optimization algorithm. In contrast, we explicitly model stylistic relevance and additionally leverage a part of speech ngram model to create more general stylistic features.

Hsu and Glass (2006) seek to address the mismatch in topic and style for lecture transcription. They have accurate corpora of lecture style, but these texts

do not match the topic of individual lectures. On the other hand, they also have textbooks and lecture slides to match the topic, but these texts do not match the style of the lectures. They address the problem by measuring class-based ngram models in a highly adaptive framework, where the classes are learned automatically and loosely correlate with part of speech classes. They reserve a special class for topical words (which are learned from textbooks and lecture slides) and the non-topical classes are learned mostly from the lecture transcriptions. In contrast to our work, we seek to automatically identify relevant styles in our training data. Additionally, we use a more specific set of classes (they use 10), which can allow our model to better model linguistic features.

4.7 Conclusions

In summary, we have successfully adapted a part of speech ngram model to the grammatical style of discourse. We adapt the tag transition probabilities to the most grammatically similar training data, using corpora as our primary unit of style. Transition probabilities can accurately represent many linguistic features of stylistic variation such as preference for pronouns, complex noun phrases (e.g., more adjectives/noun modifiers, prepositional phrases), and passive versus active voice.

Our style-adapted model applies two of the lessons learned from topic adaptation: Firstly, by default most similarity scores do not discriminate enough between different units of adaptation (both for topic and style). Therefore, we found polarization/scaling useful to increase the differences between relevance scores. In contrast, we were unable to translate the idea of inverse document frequency to style adaptation for corpus-as-style, because zero-frequency tags/pairs/triples are uncommon when measured over entire corpora. Secondly, we have demonstrated that documents can also be used as the basis for style adaptation. In this case, we only explored within-corpus style adaptation rather than adaptation in a large

mixture of styles. However, we found that we could adapt to stylistic variations within a corpus, such as different speakers' individual grammatical preferences.

The evaluations demonstrate that style adaptation is viable and beneficial in a mixture of many styles, such as mixed-domain evaluation. The adaptation framework tunes the transition probabilities to better match the overall style of each document, especially for smaller corpora. We found that the benefit of style adaptation exceeded the benefit of finding only the most relevant style (i.e., Table 4.18 on page 147). This suggests that the in-domain transition probabilities alone are not sufficient for maximizing keystroke savings, both due to sparseness and also due to potential modeling of user variations.

We have developed a novel means of language model adaptation — adapting a part of speech ngram model to grammatically relevant training data. Additionally, the overall benefits of style adaptation exceeded optimistic baselines for most corpora. Our style model accounts for an orthogonal aspect of linguistic variation compared to our topic model. Therefore, we combined topic and style adaptations into a single part of speech ngram model and found that the two adaptations could work together.

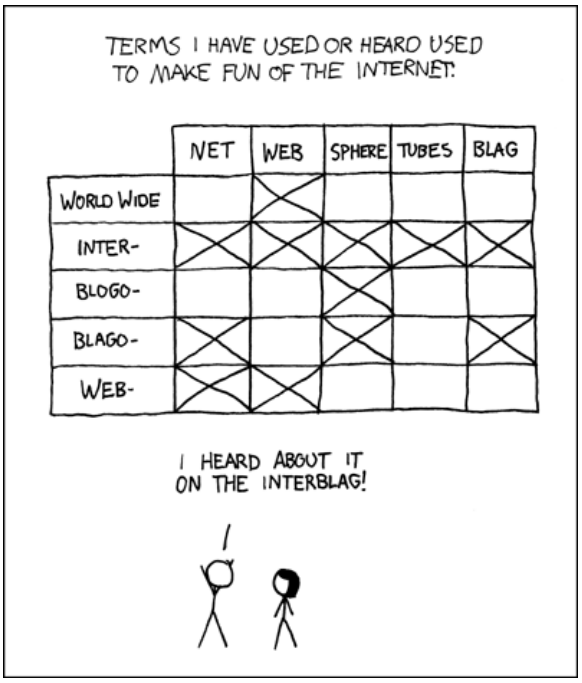
In the context of a word prediction system, this allows system developers to include a wide variety of styles in the training data and our algorithm can appropriately select the most relevant styles. Therefore devices can include training data for many different types of communication without degrading the language model. From the perspective of the user, style adaptation allows them to utilize the same device in many different situations without manual configuration for specific types of language. Additionally, our adaptation better allows users to maintain their own personal style of language without requiring additional keystrokes to use grammatical styles different than the overall training data.

In the future, we would like to expand on our style experiments to compare

corpus-as-style and document-as-style in mixed domain training. We would also like to better adapt the intuition behind inverse style frequency to corpus-as-style. Although not specific to style adaptation, we would also like to improve the baseline part of speech model by including special tags for common words.

Chapter 5

HANDLING UNKNOWN WORDS: CACHE AND DICTIONARY MODELS



xkcd.com/181

Topic and style adaptation rely on the varying relevance and irrelevance of the training data with respect to the testing data or actual usage. However, the training data may not contain all of the right words — the gap between the vocabulary and theoretical limits from Table 2.4 on page 23 is one such example. This gap

ranges from about 25% on Callhome to about 2% on Switchboard, and can only be addressed by methods which expand the prediction vocabulary. In some cases, we can create a large mixture of training data to increase the prediction vocabulary (i.e., mixed-domain evaluation), but even a large mixture will encounter words that do not appear in the training data.

This chapter will focus on two different methods for expanding the prediction vocabulary — cache modeling and dictionary modeling. Cache models are language models trained iteratively on the current partial document and applied to generating better predictions. They reduce the problem of unknown words because many new words are re-used. Although the first occurrence of a new word must still be typed letter-by-letter, subsequent uses can be predicted. Section 5.1 describes the cache techniques we applied to word prediction.

In contrast, dictionary models are unadaptive methods for vocabulary expansion. They process a large list of words and seek methods to augment the predictions with the words from the list. Dictionary methods fundamentally presume that the vocabulary of a large word list is often larger than the vocabulary of a large corpus, or at least that the word list contains some unique words. Section 5.2 presents our work with dictionary methods.

5.1 Cache Modeling

Cache modeling is acknowledged as one of the most useful language modeling methods for improving a baseline model (Goodman, 2001a). We define cache models as a model able to generate predictions by training on the current partial document alone.[1] Cache models can be implemented in many different ways — unigram word models, trigram word models using deleted interpolation, emission probability models, and so on.

[1] In contrast with (Wandmacher and Antoine, 2006), which includes not just the current partial document but also past testing documents.

The purpose of cache adaptation is both **vocabulary expansion** and modeling **lexical repetition**. The idea of vocabulary expansion is that novel words can be learned after their first occurrence, reducing the problem of out of vocabulary words (OOVs). Secondly, cache models seek to model lexical repetition (Beeferman et al., 1997) or lexical cohesion (Halliday and Hasan, 1976), promoting the prediction of words which have been used before. In contrast to vocabulary expansion, models of lexical repetition seek to re-order the existing predictions based on recency.

The problem of vocabulary expansion is often addressed with simple solutions. In particular, unigram backoff models are common for cache modeling — predictions from the cache are only generated if there is extra space in the prediction window after the baseline model has generated predictions. Although such a model would often not contribute to the predictions until several letters of the prefix have been typed, it cannot decrease keystroke savings and can benefit the user substantially in the case of longer words. Even such basic models can offer improved keystroke savings, especially when OOVs are longer words and/or when OOVs are often repeated in a document. We previously evaluated such a model on several corpora with a baseline trained on Switchboard and found that even this basic model can increase keystroke savings by 1–3% for most corpora (see Appendix D.2). We also use this basic model as a baseline in the development of a better cache model (e.g., Table 5.1 on page 170).

Li and Hirst (2005) present an evolution of the unigram backoff cache for proper nouns. The cache is constructed only for words that begin with an uppercase letter and only generates predictions if the first letter of the prefix is uppercase. Predictions from the proper noun cache are given priority over the baseline model — in cases where the word begins with an uppercase letter, the baseline model only generates predictions when the cache model is unable to fill the prediction

window completely. The intuition is that the cache is a much better model of named entities than the baseline model, because the words are not only relevant to the document but they are also implicitly appropriate for the context. They found that this method improved keystroke savings by 8.5% on nouns on the BNC corpus. We previously implemented their method and found that the approach improved keystroke savings by 2–3% on most corpora using fringe-word evaluation and trained only on Switchboard (see Appendix D.1).

There are many problems with basic cache models. The main problem with unigram backoff cache models is that many letters of the prefix may be necessary before any predictions from the cache model are useful. In general, the solution is to combine the probabilities from the baseline and cache models, but this is challenging (Kuhn and Mori, 1990, Jelinek et al., 1991). These solutions also address lexical repetition.

Models of lexical repetition generally require some combination of baseline model probabilities with cache model probabilities. Typically, such models also address the problem of predicting novel words as well. Jelinek et al. (1991) measure trigram, bigram, and unigram models on the static training data and also the most recent 200–1,000 words in testing. They combine the individual models using deleted interpolation, reducing perplexity by about 23% and word error rate by about 10%. In perplexity tests, they found that the trigram cache model resulted in much more benefit than a unigram cache, but in word error rate (WER) tests, the trigram model only reduced WER slightly compared to the unigram cache. Wandmacher and Antione (2006) applied a similar model 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 (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., they never reset the cache.

Compared to our work, this is a combination of iterative re-training (Section 3.3) and cache modeling. In this section however, we will only focus on the partial current document and not any stored user texts or previous testing documents.

Kuhn and de Mori (1990) started with a part of speech ngram model, much like our baseline model from Chapter 4. They measured an emission probability model on the partial current document and combined it with the baseline emission probability using a linear interpolation. They experimented with two different ways of weighting the cache and baseline emission models — first they tried static weights, where they ran tests with weights of 0.1, 0.2, ... , 0.9. They also developed a model of weighting the cache that conditionally depends on each part of speech tag. The intuition behind this method is similar to Li and Hirst's (2005) — certain classes of words are less likely to be predicted with a baseline model and more likely to be predicted with a cache model. For example, content words such as nouns or adjectives might be better modeled using the cache whereas function words such as determiners are less likely to vary. They found that deleted interpolation produced very high weights on many tags, but closer to 50/50 for others. However, the model with per-tag weights only slightly reduced perplexity over the more basic weighting scheme. Overall, they found that the best weighting of the cache components reduced perplexity by 68%.

In contrast to these methods, Carlberger (1998) focused more on the idea of recency promotion — a recency weight was added into a large linear combination of features for word prediction for Swedish. The recency value was computed as the product of the baseline POS model probability of the word based on the previous two tags, a frequency-like value of the word in the cache, and an additional weight based on whether the word was a function or content word. Their approach is unique and seems to predominantly help for recency promotion but not as much for vocabulary expansion. In evaluating their system with leave-one-out evaluation of

techniques, they found that the recency promotion component was responsible for 0.6% of the 46% keystroke savings for all components.

Beeferman et al. (1997) devised a more complete model of lexical repetition. In particular, they decompose repetition into forces of attraction and repulsion — each word increases the chance of seeing the word again (attraction) but temporarily decreases the chance of reuse (repulsion). They modeled the effect as a function of distance using a mixture of two exponential models. In general, they study trigger pairs (arbitrary word pairs with an attraction relationship), but we are predominantly interested in self-triggers only. Effectively, self-triggers are a representation of the cache modeling problem as a word trigger problem. Rosenfeld (1996) applied trigger pairs extensively with exponential models, but found that the most common triggers were self-triggers, which we will instead model using cache techniques rather than trigger methods.

Regardless of whether a cache model focuses on vocabulary expansion or lexical repetition, one of the main problems is data sparseness — the model is only trained on part of a single document. This makes models such as a word trigram less desirable. Although a bigram or trigram cache would offer predictions that are both relevant to the document and appropriate for the context, such a model would rarely generate predictions. In contrast, a unigram cache would offer predictions relevant to the document and would have many suggestions, but the predictions would not be appropriate for the context. We feel that Kuhn and de Mori's (1990) approach balances the need for contextually-appropriate predictions with the need for a model that can be measured effectively on very small amounts of text. Therefore, we will apply their model, shown below:

$$P(w \mid w_{-1}, w_{-2}) = \sum_{tag \in POS(w)} P(tag \mid tag_{-1}, tag_{-2}) * [(1 - \lambda) * P(w \mid tag) + \lambda * P_{cache}(w \mid tag)] \tag{5.1}$$

where λ weights the relative value of the cache model compared to the baseline emission probabilities. The model is an extension of our baseline part of speech ngram (Section 4.3) where the baseline emission probabilities $P(w \mid tag)$ are interpolated with the emission cache model $P_{cache}(w \mid tag)$. The cache probabilities are computed as $f_{cache}(w, tag)/f_{cache}(tag)$ and recomputed after every sentence is completed. We found that the model left a few questions unanswered:

- Should the cache model be used in Viterbi for tagging?
- Should the cache model be smoothed? And how?
- How should λ be computed?

We decided not to use the cache model in the Viterbi algorithm because it could potentially lead to an algorithm that tags novel words greedily. For example, if a novel word "bananadoor" were used as VB[2] in the first instance and NN[3] in other instances, it would be tagged VB in all instances if the cache model were used in Viterbi.

We implemented smoothing in the model just like the baseline POS model — each of the emission model distributions were smoothed separately using our previous formula (Equation 3.7). Unfortunately, we also need to estimate the number of words that occurred zero times for each tag in the cache model to compute the probability of an unknown word. If we use standard methods, we would assume that the number of words occurring zero times in a tag t is approximately equal to the number of words that occur only once. However, this would grossly underestimate the number of words that occur zero times for a tag $N_{0,tag}$. Therefore we used the baseline language model to help compute $N_{0,tag}$. At first, we thought to simply copy the estimate of $N_{0,tag}$ from the baseline model, but again this will lead to an

[2] Verb, base form

[3] Noun, singular

underestimate. Therefore we decided to preserve the total known number of words from the baseline model — we added the number of word forms in the baseline to $N_{0,t}$ from the baseline and subtracted any known words in the cache model. As a result, the probability of unknown words will be similar in both models.

We will discuss various methods of weighting the baseline vs cache model in Section 5.1.1. Section 5.1.2 presents a method that derives novel words using morphology in an attempt to expand the vocabulary further and also emulate trigger pair models.

5.1.1 Interpolating with the Baseline Method

We initially implemented cache adaptation using equal weights for the baseline and cache models. We evaluated this method using in-domain testing and compared to a unigram backoff cache, shown in Table 5.1. In all cases, a simple unigram backoff cache improves keystroke savings over the baseline POS model — in some cases by very little (i.e., Switchboard) and in other cases by quite a bit. The POS cache improves in all cases over the unigram backoff cache, and the improvement is substantial even for Switchboard.

	Baseline	**Unigram cache**	**POS cache**
AAC Emails	48.217%	48.718%	**49.368%**
Santa Barbara	44.586%	45.613%	**48.064%**
Callhome	47.129%	48.218%	**49.362%**
Charlotte	49.957%	50.593%	**52.116%**
Micase	48.582%	50.310%	**53.511%**
Switchboard	53.392%	53.433%	**54.493%**
Slate	49.609%	49.706%	**51.173%**

Table 5.1: POS cache and unigram backoff cache using in-domain evaluation. Baseline POS model is shown for reference. The maximum for each row is shown in bold.

Although the POS cache is beneficial with even weights, we would prefer to

use adaptive weights for the cache model. Weights should reflect the value of each model, but the cache model's value fundamentally increases as the cache becomes saturated. Furthermore, the value of the cache may be different depending on the type of document (e.g., speech with frequent topic shifts vs news stories) and the value may vary depending on part of speech tags (e.g., function words and content words). The following two sections explore two different weighting schemes.

5.1.1.1 Adaptively Weighting the Cache's Value

In contrast to Kuhn and de Mori (1990), we want the weight of the cache to evolve as the document as typed, reflecting the increasing value of the cache as the document grows and also implicitly modeling the greater value of cache models on longer documents.

We compute weights by win-counting based on probability — the model that assigns the highest probability to the previous word being typed has "won" the prediction of the word. The weights are the normalized number of wins for each model. They must be initialized to non-zero values at the beginning of a document — we chose to initialize each model to 10 wins. We briefly experimented with other initializations but found that they decreased keystroke savings.

We would like the relative weight of the cache and baseline models to accurately reflect the changing utility of the cache model. At the beginning of a document the cache model provides no predictions and has low utility, but as the document is typed the cache model becomes more useful. Our win-counting method underestimates the value of the cache model — it effectively averages the value of the cache over the entire document so far. Although this value will increase over time, we expect the cache model to have more wins on the most recent words. If we compute the wins over the most recent words, we expect to weight the cache model more accurately than considering older wins. Therefore, we added win decay to the wins — more recent wins are better indicators of cache utility. We chose to decay

the weights exponentially — multiplying the win counts by 0.95 after every word. We found that decay improved the cache model, especially on Micase, which has longer documents (not shown).

Table 5.2 compares the adaptively-weighted cache with the evenly-weighted cache using an in-domain evaluation on all words. The adaptive method improved keystroke savings over the even weights by a small amount for all corpora, but had a larger benefit for Slate and AAC Emails. This may reflect a relatively higher proportion of out-of-vocabulary words (OOVs) in these corpora relative to their size. It may also reflect a higher proportion of proper nouns, which are likely to be rare in the training data (possibly OOVs) and also tend to be repeated often within a document.

	Unadaptive POS cache	**Adaptive POS cache**
AAC Emails	49.368%	49.556% (+0.188)
Santa Barbara	48.064%	48.083% (+0.019)
Callhome	49.362%	49.396% (+0.034)
Charlotte	52.116%	52.129% (+0.013)
Micase	53.511%	53.574% (+0.063)
Switchboard	54.493%	54.538% (+0.045)
Slate	51.173%	51.319% (+0.146)

Table 5.2: POS cache with even weights compared to adaptive weights with weight decay, using an in-domain evaluation on all words. The improvement is shown in parentheses.

The adaptive weights are more beneficial than static weights. Interestingly, the weights tended to favor the baseline model, in contrast with Kuhn and de Mori's weights which tended to weight the cache model more. The difference may reflect differences in deriving the weights. In particular, our method attributes wins to the baseline model even when the cache model provides no predictions or completions. Therefore, we evaluated a version where wins were only counted if the cache model could produce predictions, but found that it decreased keystroke savings.

5.1.1.2 Weights as a Function of POS Tag

Kuhn and de Mori (1990) found that training different weights for each part of speech tag was slightly beneficial.[4] Unfortunately, we found that weights conditioned on tags caused some problems with win-counting. To summarize, we were unable to find any method of conditioning weights on part of speech tag that increased keystroke savings compared to the more basic adaptive weights.

Part of the problem of adapting weights to part of speech is that the exact tag of each word is not available until the end of the sentence. Originally we counted wins for any tags in which the cache model was defined for the word/tag pair. We found that this approach performed poorly, and instead only accumulated wins after a sentence was tagged.

Secondly, we found that there was a major problem in counting wins by tags. Specifically, the weights are undefined until we encounter at least one instance of each tag. Even after encountering the first instance of a tag, we must take care to avoid assigning full weight to one model or the other. We addressed the problem of undefined weights by defaulting to the unconditional scores if the tag was not yet encountered. We addressed the second problem by initializing the wins to five each just before counting the first win for each part of speech tag. In this way, we avoid the knee-jerk reaction from the first win. Although this was an improvement, the results with the conditional weights were still inferior to the unconditional weights.

Again we suspected that sparseness could be an issue so we instead modeled the conditional weights as modifications of the overall weights — the wins for a particular tag would scale the basic weights up or down. We modified them according to the following formula and then normalized afterwards:

$$weight(cache \mid tag) = weight(cache)^{\frac{E(wins(cache|tag))}{wins(cache|tag)}} \tag{5.2}$$

[4] It is interesting to note that the results were not substantially different even with very different weights for different tags.

where $weight(cache)$ is the normalized *un*conditional win count for the cache model. $E(wins(cache \mid tag))$ is the expected value of $wins(cache \mid tag)$ under the assumption that the conditioning information of the tag is irrelevant. In the case of no decay, the expected value is the normalized unconditional win count of the cache times the total number of times $f(tag)$ in the cache. If the cache model wins more often than expected, then the exponential weight is less than one, which increases the weight (because it is less than one). As with previous methods, the win counts per tag were initialized to five wins for each method. We found that this method was the best of our attempts at designing weights that were conditioned on each tag, but the results were slightly worse than basic adaptive weights.

We evaluated the best approach on all corpora with in-domain evaluation, shown in Table 5.3. The conditional weights (POS-specific adaptive) offer a smaller improvement over the baseline compared to unconditional weights (Basic adaptive). Although we intuitively feel that conditioned weights have more potential than unconditional weights, our experiences suggest that implementation of conditional weights is complicated in a dynamic adaptation (in contrast to statically tuning weights to held-out data).

	Unadaptive	**Basic adaptive**	**POS-specific adaptive**
AAC Emails	49.368%	**49.556%**	49.515%
Santa Barbara	48.064%	**48.083%**	48.044%
Callhome	49.362%	**49.396%**	49.364%
Charlotte	52.116%	**52.129%**	52.099%
Micase	53.511%	**53.574%**	53.516%
Switchboard	54.493%	**54.538%**	54.509%
Slate	51.173%	**51.319%**	51.299%

Table 5.3: POS-specific adaptive weights compared to basic adaptive weights and even weights, using an in-domain evaluation. Weight decay was not applied to the POS-specific weights due to implementation issues. The maximum for each row is shown in bold.

5.1.2 Automatic Derivation of Unknown Words

Our cache model predicts novel words in context fairly well, balancing sparseness issues with context using the part of speech model. However, much of the research of the model (e.g., weights) focuses on balancing the benefit to novel words with the benefit or detriment to predicting words encountered in training.

We designed a model to derive alternate forms of words to expand the vocabulary even further than a basic cache. For example, when a novel word "tweeting" is encountered in testing, the model might also add words like "tweet", "tweets", and "tweeter" to the cache. As with the purpose of cache modeling, the model was motivated both by vocabulary expansion and by lexical cohesion. The benefit to lexical cohesion is that words such as "tweets" are more likely after seeing words such as "tweet" earlier in the document. In the literature of trigger pairs, Rosenfeld (1996) found that the most common triggers were self-triggers and triggers of alternative forms of the same lemma. The goal with respect to vocabulary expansion is to hopefully add some novel words to the language model *before* their first occurrence in testing. The goal with respect to recency promotion is to better approximate trigger pairs, allowing the occurrence of a word to increase not only its own likelihood but also the likelihood of other forms of the word.

The goal of the inflection/derivation model is to produce the following probability and a set of $suffix_2$ given a word w which is represented as a base and suffix:

$$P(suffix_2 = inflection(base_i \in d_j) \mid suffix_1 = inflection(base_i \in d_j)) \quad (5.3)$$

In other words, given that we encounter one word (which is represented as a base with a suffix), we want to compute the probability that another form of the same base is likely to occur in the same document. Note that we are not interested in deriving all possible forms of $base_i$ but only those forms that we expect to encounter

in the same documents. For example, a document containing "literate" may be likely to include "literacy" but less likely to include "literately".

The model is constructed by applying Porter's stemmer in training to build a frequency distribution $f(base, suffix)$. This distribution is simply a more structured form of the word distribution $f(word)$. At the end of each document, we iterate over each *base* this distribution and increment a distribution $f(suffix_1, suffix_2)$ for any cooccurring suffixes of each base form, measuring that the two inflections of the base occurred in the same document.[5] We compute probabilities without any discounting:

$$P(suffix_2 \mid suffix_1) = \frac{f(suffix_1, suffix_2)}{f(suffix_1)} \tag{5.4}$$

Our model reflects the intuition that a given suffix $suffix_2$ may be valid for all words using another suffix $suffix_1$ — the validity of applying a suffix can be approximated by inspecting known suffixes of a base form.

When a word is added to the cache, we apply Porter's stemmer to the word to determine the base and suffix. If the suffix is non-empty, we check the model $P(suffix_2 \mid suffix_1)$ and postulate any suffixes that are above a threshold t_1.[6] We use $t_1 = 0.1$ following trial-and-error.

We then apply the suffix tagger from Section 4.3.2 to obtain a probability distribution over part of speech tags for the postulated words. We threshold this distribution, ignoring any tags below a second threshold t_2. We take $t_2 = 0.2$. In general, t_1 is a control on how outlandish the suggested words can be and t_2 is a way to control the amount of data we flood the cache with. We update the emission

[5] In future work, we'd like to replace this with some sort of "degree" of co-occurrence.

[6] We did not want to inflect/derive words with no suffix because we felt the model would produce too many erroneous words. We suspect that suffixes identified by Porter's stemmer are likely to be legitimate. In contrast, Porter's stemmer may be more error-prone in identifying an empty suffix.

cache model by the probability of generating the word times the probability of the tagging. The original word form and tag still contribute a frequency of one, but the inflected forms contribute frequency in proportion to their likelihood, which helps to minimize the risk of unlikely words.

We performed a pilot evaluation the method with in-domain testing on AAC Emails and Micase. The results are separated by whether the testing word was seen in training or not, to separately evaluate the contributions of vocabulary expansion and recency promotion. Table 5.4 shows the results on AAC Emails — the inflection/derivation model improves keystroke savings on not just unseen words but also has a slightly beneficial effect on seen words. Table 5.5 shows the results on Micase, which benefits much more from cache modeling. The model improves keystroke savings on unknown words for Micase at the cost of a slight decrease in seen word keystroke savings. However, the overall results show that the benefit to unseen words outweighs the effect on seen words.

AAC Email	**Keystroke savings (W=5)**		**Perplexity**	
	Seen	Unseen	Seen	Unseen
No cache	54.934%	0%	259.96	44123.76
Basic adaptive cache	55.551%	6.430%	256.02	16200.90
+ inflection ($t_1 = 0.01, t_2 = 0.2$)	55.563%	6.916%	256.02	16035.75

Table 5.4: Evaluation of the cache inflection model on AAC Email.

The results show that the model has successfully produced some novel words which occur in training. When experimenting with the thresholds, in general we found that keystroke savings improved slightly even when more words were postulated (i.e., lower t_1). However, the extra keystroke savings comes at the cost of more nonsensical words, which may be confusing to a user even if it doesn't reduce keystroke savings. For example, one of the incorrect inflections was $family \Rightarrow familie$.

Micase	**Keystroke savings (W=5)**		**Perplexity**	
	Seen	Unseen	Seen	Unseen
Basic adaptive cache	53.497%		289.27	
	55.345%	33.072%	230.07	22889.83
+ inflection ($t_1 = 0.01, t_2 = 0.2$)	53.576%		290.02	
	55.311%	34.403%	231.32	21748.74

Table 5.5: Evaluation of the cache inflection model on Micase.

We attempted to address some of the problems with the model by instead measuring tag-suffix pairs: Rather than measure the cooccurrance of $suffix_1$ and $suffix_2$, we instead measure the cooccurrance of $suffix_1, tag_1$ with $suffix_2, tag_2$. This can be helpful in cases where the reported suffix is an ambiguous indicator of part of speech. It allowed us to generate words from empty suffixes and also constrained the generations to be more sensible. However, we found that this approach produced less keystroke savings. Although it generally produced fewer unnatural words, it also reduced the number of natural words that were generated, thus lowering keystroke savings.

The approach could be improved in many ways. Firstly, a better morphological tool could be used (e.g., PC-KIMMO (Antworth, 1990)). PC-KIMMO could be used to generate legitimate inflections and derivations without the need for as much statistical learning. Secondly, common words (which tend to have more irregular morphology) might be addressed using different methods than uncommon words. For example, words of high frequency in the training data might only be allowed to generate other words from the training data, whereas the inflection model might be applied only for infrequent words.

Baroni et al. (2002) address a somewhat related problem in German — derivation of noun compounds, specifically two nouns which are compounded into a single token. The frequency of German noun compounds combined with the sheer number

of possible nouns puts the problem outside the realm of training data. Although they postulate new word forms like us, their methods are very different.

Carlberger (1998) mention inflection of word forms, but instead seek to compress the prediction vocabulary. They attempt to store only base forms along with a list of allowable morphological rules, thus reducing the storage space.

5.1.3 Conclusions

In summary, our cache adaptation provides an effective model of both novel vocabulary and lexical repetition. By leveraging the transition probabilities from our baseline model along with an emission probability model for the cache, we are able to predict novel words in context without substantial issues from data sparseness. A bigram or trigram word cache model would be able to predict novel words in context, but only in very limited contexts. In contrast, when a novel noun is added to our part of speech cache model, it can be predicted in all contexts appropriate for nouns.

We improved on the basic emission cache model by dynamically adjusting the relative weight of the cache and baseline models throughout a document. This allows the cache model weights to increase as the cache fills and becomes more useful. Implicitly, this accounts for the differences in cache model value between corpora with longer documents (e.g., Micase) and corpora with shorter documents (e.g., AAC Email). We found that weights specific to each part of speech tag were cumbersome and did not improve over unconditional weights.

In addition to the basic cache model, we developed a word derivation model to further expand the vocabulary of the cache. When a novel word (or any word) is encountered in testing, the derivation model not only adds the word to the cache but also adds alternate derivations and inflections of the word. Although this may not help on some corpora, the model produced a noticeable benefit on Micase. Furthermore, we analyzed the improvement on both known and unknown words,

demonstrating that the derivation model can indeed derive novel words and improve keystroke savings.

In the future, we would like to combine our part of speech cache model with a word bigram or trigram cache model. Although a word trigram model would rarely suggest predictions, the predictions it suggests are more appropriate for context than predictions from the part of speech cache. Therefore, we expect that we can leverage both the strong contextual model of word bigram/trigram caches as well as the large coverage of the part of speech cache.

We have presented novel methods for cache adaptation. The part of speech cache model follows Kuhn and de Mori (1990) with one major difference — our adaptive weights are determined dynamically and evolve along with the document. Additionally, we have presented a novel algorithm for expanding the prediction vocabulary — we can successfully derive new words using morphology in conjunction with statistical learning.

In the context of word prediction for AAC, we have developed an effective method of increasing keystroke savings, especially when no relevant training data is available. The overall benefit using in-domain evaluation is shown in Table 5.6 — our cache model improves keystroke savings from about 1%–5% depending on corpus features such as document length and OOVs. We expect more benefits using out-of-domain evaluation and in actual system usage. This model helps to complement topic and style adaptation in our system by improving the model even when there is no relevant training data.

5.2 Dictionary-based Methods

Dictionary-based methods process a large list of words and integrate the words into word prediction. Even very basic methods can yield a tangible increase in keystroke savings, especially when the training corpus is small. Our work with

	No cache	POS cache, adaptive
AAC Emails	48.217%	49.556% (+1.339)
Santa Barbara	44.586%	48.083% (+3.497)
Callhome	47.129%	49.396% (+2.267)
Charlotte	49.957%	52.129% (+2.172)
Micase	48.582%	53.574% (+4.992)
Switchboard	53.392%	54.538% (+1.146)
Slate	49.609%	51.319% (+1.710)

Table 5.6: Final gains from part-of-speech cache model using in-domain evaluation. The cache model uses basic decay, but the inflection model was not included.

dictionary backoff uses the yawl-0.3 word list, which contains about 250,000 words.[7]

For example, we previously included basic dictionary backoff in all evaluations. If the normal language models could not generate enough predictions to fill the window, words matching the prefix were generated from the dictionary in alphabetic order. This approach is unlikely to affect the predictions until 3 letters or so of the prefix have been entered, but like a unigram backoff cache, cannot decrease keystroke savings. Table 5.7 evaluates this basic type of dictionary approach. The benefit varies by corpus, from only minor improvements to almost 1%. Also, note that dictionary backoff works best at larger window sizes, because the dictionary predictions are only added after any words from the baseline mode.

The improvement is substantial compared to other methods, especially considering the ease of implementation. However, predictions offered by the dictionary model are not appropriate for the context; they are only appropriate for the prefix.

[7] YAWL is available from http://freshmeat.net/projects/yawl as of 11/2010. The list does not include proper nouns.

	Baseline	Basic dict.
AAC Emails	48.217%	49.602% (+1.385)
Santa Barbara	44.586%	45.439% (+0.853)
Callhome	47.129%	48.283% (+1.154)
Charlotte	49.957%	50.772% (+0.815)
Micase	48.582%	49.684% (+1.102)
Switchboard	53.392%	53.509% (+0.117)
Slate	49.609%	49.764% (+0.155)

Table 5.7: Basic dictionary backoff compared to baseline method using in-domain evaluation.

5.2.1 POS-tagged Dictionary using Suffix Tagger

The suffix tagger allows us to postulate reasonable taggings for the word list. Then given the most probable taggings, we can leverage the transition probabilities to ensure that any predictions from the dictionary are appropriate for the context. Note however, that the predictions from the dictionary are still generated only if the baseline model does not generate enough predictions.[8]

In tests using the cheat-lookahead setting for tagging (i.e., we look up the tag of the next word), this improved keystroke savings over the basic method by 1% or more. However, in a real test using Viterbi and transition probabilities, we found that this was sometimes better and sometimes worse than the basic dictionary method (see Table 5.8). In simple tests, we found that generating words from the most likely tag and then applying the basic model was better, indicating that the POS dictionary lacks some modeling of the basic dictionary.

The primary difference between the basic and POS models is that the basic model orders words alphabetically and the POS dictionary orders words based on

[8] It may be possible to improve on our methods by integrating the dictionary-based predictions somewhat with the predictions from the training corpus, however we did not test that yet. In the current setup, predictions from tags that are unlikely for the context are still generated before the dictionary is checked.

	Baseline	Basic dict.	POS dict.	POS dict. with exp weight
AAC Emails	48.217%	49.602%	49.544%	**49.850%**
Santa Barbara	44.586%	45.439%	45.445%	**45.574%**
Callhome	47.129%	48.283%	48.266%	**48.497%**
Charlotte	49.957%	50.772%	50.776%	**50.892%**
Micase	48.582%	49.684%	49.736%	**49.814%**
Switcboard	53.392%	53.509%	53.503%	**53.512%**
Slate	49.609%	49.764%	49.767%	**49.772%**

Table 5.8: POS dictionary backoff (with and without exponential length-weighting) compared to basic dictionary backoff and baseline. The maximum for each row is shown in bold.

transition probabilities, and then by tagging confidence within each tag. However, the basic model's alphabetical sort implicitly predicts shorter words before longer words, whereas the POS model doesn't account for word length. The POS dictionary is a full-fledged emission probability distribution but assumes all words are equally frequent. The following section will explore the possibility of estimating the relative frequencies of words based on word length, and thereby restoring some of the ordering from the basic dictionary model.

5.2.1.1 Frequency as a Function of Word Length

The dictionary model is effectively a normal emission probability model, but computed from impoverished data without frequencies. Although our method accounts for context, the effect of word length clouded any benefit. For example, our dictionary contains both "antibiotic" and "anticholinesterase". Clearly, the former is more likely in most contexts.

We decided to estimate the frequency of words as a function of word length. Then the emission frequencies are computed as:

$$f(w, tag) = weight(w) * P(tag \mid w) \tag{5.5}$$

where $weight(w)$ is an estimate of the relative frequency of w. The weight doesn't need to be a true frequency; only a value that is comparable to other values in the dictionary model so that normalization can produce a sensible probability distribution.

We explored three different methods of estimating frequencies — exponential weighting, linear weighting and modeled weighting. The following sections will describe each.

5.2.1.1.1 Exponential Weighting

The basis of exponential weighting is that words of length n are twice as likely as words of length $n+1$:

$$weight(w) = 2^{10-L(w)} \tag{5.6}$$

Although we use base 2 for example, we also tried base 1.5 and base 1.3. Some example weights (before and after normalization) are shown in Table 5.9.

Word length	**Exponential weight** ($base = 2$)	**Exponential weight** ($base = 1.5$)
1	512 (1.000)	38 (1.000)
2	256 (0.500)	25 (0.667)
3	128 (0.250)	17 (0.444)
4	64 (0.125)	11 (0.296)
5	32 (0.062)	7 (0.198)
6	16 (0.031)	5 (0.132)

Table 5.9: Example exponential weights.

We performed a basic evaluation and found that 1.5 as the base seemed best. In tests with cheat-lookahead, exponential weighting improves results by about 0.4% keystroke savings compared to an unweighted version. The effects in a full test are shown in Table 5.8 and Table 5.12.

5.2.1.1.2 Linear Weighting

Linear weighting decreases the relative frequency by a constant for each additional letter, and treats all words above length 9 as equal:

$$weight(w) = \frac{10 - min(L(w), 9)}{10} \tag{5.7}$$

We picked the constants under the assumption that 10 is a reasonable maximum length for normal words. Example weights are shown in Table 5.10. The relative

Word length	Linear weight	Normalized
1	0.9	1.00
2	0.8	0.89
3	0.7	0.78
4	0.6	0.67
5	0.5	0.56
6	0.4	0.44

Table 5.10: Example linear weights.

frequencies decay slower than exponential weighting at 1.5.

We evaluated linear weights but found that they were worse than the exponentially weighted dictionary words, although they still produced a substantial benefit over the unweighted baseline.

5.2.1.1.3 Modeled Weighting

Finally, we decided to base the weights on actual corpus data. We measured the mean and median frequencies on AAC Emails for words of each length on the training data for the system. The resulting weights (before and after normalization) are shown in Table 5.11. We had hoped to use median-based weighting for its reliance against outliers, but found that medians were unhelpful, especially on small corpora. In general, the mean-based weights mirror the exponential weights with base 2.

Word length	Mean-weighted	Median-weighted
1	83 (1.00)	3 (1.00)
2	40 (0.48)	2 (0.67)
3	19 (0.23)	2 (0.67)
4	9 (0.11)	2 (0.67)
5	5 (0.06)	2 (0.67)
6	3 (0.04)	1 (0.33)

Table 5.11: Example modeled weights.

We evaluated both and found that they were worse than the other weighting schemes. The deficiency of modeled weighting is likely due to sparseness in the data — training corpora are poor estimates of the relative frequency of words of length 10 and 11, for instance. In contrast, although the exponential weighting scheme is artificial, it matches the modeled data reasonably well for short words and provides clear distinctions between words of all lengths.

5.2.2 Conclusions

In summary, we developed methods to integrate a large word list to improve word prediction. Our algorithm relies on the suffix tagger to postulate likely part of speech tags for each word based on morphology, then leverages the baseline transition probabilities to predict dictionary words in appropriate contexts. We found that we needed a model of relative word frequency, and the best function was an exponential function of word length. Thus, the dictionary model has approximations of both context and frequency. The modifications led to a significant increase in keystroke savings for all corpora. The final gains with in-domain evaluation are shown in Table 5.12 — dictionary backoff leads to a significant improvement on all corpora, more so for smaller corpora.

The dictionary model complements our other language modeling improvements. Topic and style adaptation depend on the training corpora, which can be

	Baseline	POS dict. with exp. weight
AAC Emails	48.22%	49.85% (+1.63)
Santa Barbara	44.59%	45.57% (+0.98)
Callhome	47.13%	48.50% (+1.37)
Charlotte	49.96%	50.89% (+0.93)
Micase	48.58%	49.81% (+1.23)
Switchboard	53.39%	53.51% (+0.12)
Slate	49.61%	49.77% (+0.16)

Table 5.12: Final gains from dictionary backoff, using a the part-of-speech tagged dictionary with exponential length weighting.

sparse or simply not have the right words. In contrast, the dictionary model leverages the larger vocabulary of word lists to help ensure that the model can predict a sufficiently large vocabulary. The dictionary model also complements our cache model — the basic cache model can only improve the prediction of words already encountered in testing. In contrast, the dictionary model can help to predict the first occurrence of words that are not included in the training corpora.

In the context of the overall system, dictionary backoff adds more resilience against poor choices of training data. The research also has a side benefit for device maintenance and configuration — AAC practitioners or caretakers can add new words to the vocabulary of the model and the system will be able to predict these words in contextually appropriate situations.

The results with a perfect transition probability model (i.e., cheat-lookahead) suggest that improvements in the transition model may have a large effect on the quality of the dictionary-based predictions. Additionally, we would like to expand the work by integrating dictionary predictions more closely with the baseline model. One the of disadvantages of the backoff approach is that words in the training data will always be predicted higher in the prediction window than those in the dictionary. Although this is a reasonable effect in many situations, this also means that words encountered in training that are inappropriate for the context can displace words

in the dictionary that are appropriate for the context. We would like to investigate these issues in further research.

Chapter 6

CONCLUSIONS

Writing a book is an adventure. To begin with, it is a toy and an amusement. Then it becomes a mistress, then it becomes a master, then it becomes a tyrant. The last phase is that just as you are about to be reconciled to your servitude, you kill the monster, and fling him to the public.[1]

Winston Churchill

The problem of word prediction for augmentative and alternative communication often appears as a problem in obtaining relevant training data. We have addressed this problem in several ways. Chapter 2 describes our rigorous evaluation, which approximates the problem of AAC texts with evaluation methods. We evaluate using a mixture of corpora and analyze the results individually on each corpus. This allows us to analyze high-level trends in word prediction, such as whether techniques are more beneficial to corpora with very large vocabulary (e.g, Micase, Slate) or smaller vocabulary (e.g., Switchboard). Similarly, it allows us to separately estimate the benefits of language modeling improvements on different types of texts (e.g., written vs. spoken, formal vs. informal). Additionally, we emulate real-world usage by varying the similarity of the training data to actual usage. In contrast to most research, where the training data is held constant, we hold the testing data

[1] This may also apply to reading a book!

constant, varying the relationship of the training data to testing from in-domain to mixed-domain to out-of-domain. This gives a better sense of the effect on AAC users, where not all of the training data will be relevant or possibly even all of the training data may not be relevant. We have also developed detailed evaluations specific to our language modeling improvements (e.g., per-tag analysis for part of speech models, seen/unseen evaluation for cache inflection model).

Although we can generally increase language model performance with more training data, these efforts are unlikely to increase the average relevance of training texts. Instead, many documents will be dissimilar to testing data. In extreme cases, the entire training corpus will be of only partial benefit. Even if a few relevant texts are added to the training data, their contribution will be drowned out by the abundance of irrelevant texts. We have addressed this problem in two general ways.

We address the problem of varying relevance through adaptive language modeling. Specifically, we have developed topic adaptations (Chapter 3) and style adaptations (Chapter 4). Both adaptations function as a two-stage process: First, groups of texts (either topics or styles) must be compared to the current partial document for relevance. Then, we tune the language model to weight relevant training data more highly than irrelevant data. This approach reduces the problem of training data weighting to a problem of topical and stylistic relevance. We have demonstrated that topic and style similarity scores can be effective measures of relevance for adaptation.

We found that this adaptation increases keystroke savings for both topic and style adaptation individually, and also when topic and style modeling are combined. Our topic and style adaptations have two main benefits: Firstly, they improve keystroke savings and ensure that predictions are relevant both for the short-distance context (e.g., grammar) and the long-distance context (i.e., topic, style). Secondly, our adaptation methods allow developers of AAC devices to include more text —

even if the text is not useful to many AAC users, topic and style adaptation will weight the text according to its relevance to the current discourse.

We have addressed the problem of irrelevant training data in another way — we have developed cache models to learn new words and model lexical repetition, and also integrated a large word list using methods developed for the part of speech ngram model (Chapter 5). Our cache model balances the opposing goals of modeling context and data sparseness by using a part of speech ngram model. We additionally apply dynamic weighting of the cache model compared to the baseline, reflecting the increasing utility of the cache model as a document is typed. We have extended cache modeling in a novel way by automatically deriving new words using morphology. Overall, the cache modeling produced the largest improvement in keystroke savings, especially for corpora with long documents and smaller corpora (i.e., 1–5% with in-domain evaluation).

The cache model primarily predicts words after they occur once in the testing document, but dictionary-based methods instead imitate a large corpus which may contain many rare words. We found that we could improve over basic word list methods by tagging the words based on suffixes, then leverage the baseline part of speech model to predict dictionary words in appropriate contexts.

Our research has made significant contributions to the field of natural language processing:

- domain-varied evaluation

  Our evaluation methods are applicable to any training/testing problem and allow researchers to better estimate the expected performance of their methods in best-case situations (in-domain evaluation) compared to real-world situations (out-of-domain evaluation and mixed-domain evaluation).

- user adaptation as a topic modeling problem

  We have demonstrated how topic adaptation can be used in a simple manner

to address the problem of user adaptation (Section 3.3).

- topic adaptation for part of speech tagging

  Topic adaptation in a Markov model part of speech tagger increases tagging accuracy by optimizing the emission probabilities. We would like to evaluate this for pure part of speech tagging in the future.

- style adaptation

  The design of our style adaptation is a novel contribution to language modeling, and demonstrates the usefulness of a part of speech model for integrating linguistically-motivated information.

- reasonable gold standard for evaluation of language model combinations

  The reasonable gold standard described in Section 4.5.3 can be applied to language model combination in general. Even though the output is in terms of keystroke savings, the method can be applied for other tasks to evaluate the quality of a combination model and furthermore can analyze the overlap in the utility of different language models.

- dynamic adaptation for part of speech cache modeling

  We have extended the work of Kuhn and de Mori (1990) from static adaptation using EM to dynamic adaptation, which allows the weight of the cache model to evolve along with the document. Our dynamic weighting scheme also allows the model to work well for different types of documents (where the benefit of cache modeling may differ).

- better utilization of word lists

  We have demonstrated that large, freely available word lists can have a significant benefit in language modeling. Furthermore, we designed a method for better utilizing the information in a word list by suffix tagging the list.

Additionally, we have made significant contributions to the field of word prediction for augmentative and alternative communication:

- designed more rigorous evaluations

  Our evaluations better estimate the real-world benefit of word prediction methods, especially for different types of users (Section 2.3). Furthermore, we have concretely identified the upper bound for improvements in keystroke savings (Section 2.2.1.3). Also, we have identified many complications with keystroke savings and we have correlated keystroke savings with user evaluation.

- principled natural language processing

  We have integrated research in natural language processing with the field of augmentative and alternative communication and demonstrated that keystroke savings can be improved substantially over typical models in AAC, such as trigrams with basic models of recency.

- adaptive methods to help developers

  Not only do our adaptive methods benefit users of AAC devices, but adaptive methods also benefit device developers and practitioners in choosing relevant training data. Notably, developers can include more text and more diverse text in devices because our adaptive methods help to reduce the degradation in performance due to adding irrelevant text. Furthermore, topic and style adaptation allow practitioners to include data that may match actual usage only in topic or style (e.g., adding a textbook as training data to help word prediction in class conversations or homeworks).

- adaptive methods without additional user guidance

  In contrast to previous AAC methods for topic-specific vocabulary, our methods require no interaction on the part of the user. Additionally, previous

methods allow for only binary relevance of vocabulary (i.e., select a whole vocabulary set or not) whereas our methods model degrees of relevance — the system can still predict the entire vocabulary but relevant words are predicted sooner. Even further, our methods allow for multiple relevance — texts which encompass intersections of multiple topics or styles, in contrast to selecting a single vocabulary set. Finally, we have presented cache and dictionary methods which help a developer to ensure that the predictions have high quality even if the training data does not include much (or any) relevant data.

6.1 Future Work

We plan to **combine the different sources of information** for word prediction into a single model in the future. Fortunately, all of our methods have been implemented in a part-of-speech modeling framework. The transition probabilities are affected only by our style adaptations, whereas emission probabilities are affected by topic, cache, and dictionary methods. We will show our plan in the part of speech equation:

$$P(w \mid h) = \sum_{tag \in POS(w)} P(tag \mid tag_{-1}, tag_{-2}) * P(w \mid tag) \tag{6.1}$$

In the planned combination, $P(tag \mid tag_{-1}, tag_{-2})$ is the result of style adaptation:

$$P(tag \mid tag_{-1}, tag_{-2}) = \sum_{s \in styles} P(s \mid h) * P(tag \mid tag_{-1}, tag_{-2}, s) \tag{6.2}$$

where $P(s \mid h)$ is the normalized style relevance score and $P(tag \mid tag_{-1}, tag_{-2}, s)$ is a transition probability model trained on style s, discussed in Chapter 4. This fine-tunes the transition model to stylistically similar training data. The emission probability will be a combination of the cache model using the topic model as the baseline model of the training data:

$$P(w \mid tag) = (1 - \lambda) * P_{topic}(w \mid tag) + \lambda * P_{cache}(w \mid tag) \tag{6.3}$$

where $P_{cache}(w \mid tag)$ and λ are determined according to Chapter 5. The topic-adapted probabilities will use our topic linear combination:

$$P_{topic}(w \mid tag) = \sum_i P(topic_i \mid h) * P(w \mid tag, topic_i) \tag{6.4}$$

where $P(topic_i \mid h)$ is the normalized topical relevance score, discussed in Chapter 3 and briefly again in Chapter 4. $P(w \mid tag, topic_i)$ is the emission probability measured on only training data from topic i. The dictionary model can be integrated in the same way as Chapter 5 — adding predictions only if the list is not full.

However, there are two variations we would like to explore regarding the combination. Firstly, we suspect that better improvements are possible from the dictionary model by integrating into the baseline emission probabilities. This may be reasonable without adaptive methods, for example the maximum frequency of any unknown words in the dictionary would be set to one.[2] Although this may be possible in the topic-adapted POS model after the topic interpolation, it may also be possible to treat the dictionary as a general-purpose topic model and allow topic relevance to compute an appropriate weight.

Secondly, the weighting system of the cache model and the topic model might be combined. In the equations above, we weight the cache vs. topic model first, then separately weight the individual topic models. Also note that the weighting methods are different. The structure of our overall combination follows the intuition that the cache model is a different type of model than each of the topic-specific models. However, it may be possible to combine the weighting schemes and treat the cache like any other topics in the training data.

In addition to combining the various sources of adaptive information in our system, we would also like to **combine word ngram information and part of**

[2] If other methods cause frequencies to take real values, we would use the minimum frequency in the training data, rather than 1.

speech ngram information. Word bigrams and trigrams provide highly appropriate predictions for the context, but they generate a limited number of predictions due to data sparseness. In contrast, part of speech ngram models can generate the full vocabulary of unigram models, but with a more basic model of context than word bigrams and trigrams. For example, the part of speech model is much better at predicting words after a proper noun or number due to data sparseness. However, the POS model treats too many words equivalently (e.g., the transition model views "a door" and "an door" as equally likely). Therefore, we would like to replace the word unigram model with the part of speech ngram model in the backoff process, following Niesler and Woodland (1996). Although this may downplay some of the benefits of style adaptation, the contribution from cache and dictionary modeling should remain intact.

We would also like to extend the baseline models to account for **multi-word prediction** — the theoretical keystroke savings limits of Section 2.2.1.3 demonstrate that multi-word prediction may be one of the untapped resources available for improvement. We have explored a basic approach where we modified the tokenizer to group known multi-word expressions, but we found that this approach decreased keystroke savings. Instead, we would like to generate multi-word predictions probabilistically, incorporating both single-word and two-word predictions in the prediction window.

We would like to improve the baseline part of speech ngram model, perhaps by including **special tags for common words** following Copestake (1997). We performed pilot experiments of this method with in-domain evaluation and found a benefit on a small corpus (AAC Emails) — 0.75% when using special tags for the 200 most common words. On a larger corpus (Switchboard), the improvement is more drastic — 2.68% improvement when using special tags for the 200 most common words. The more strongly conditioned language model takes advantage of

the larger amount of training data in Switchboard. Like many trends in NLP, we found that there were diminishing returns in going from the baseline to 50, 100, and 200 special tags. Note however, that this addresses only some of the weaknesses of part of speech models (such as the a/an distinction) but does not address weaknesses involving less common words (e.g., colocations, selectional restrictions).

However, the part of speech model also enables more **linguistically interesting models**. Notably, the POS model provides basic information about linguistic structure which can be used to further condition predictions. For example, we can have a different emission probability model structure for each tag. We could additionally condition the relative probability of he/she in the pronoun emission model based on the gender of the most recent person name. The transition probabilities can also reflect more linguistic information — for example the idea of a verb tense cohesion could be applied to better predict the verb category (e.g., past vs present tense).

Additionally, basic improvements in the **dictionary model** may still be possible by more closely integrating the model into the baseline model. Specifically, contextually appropriate words from the dictionary should be predicted sooner than contextually inappropriate words from the baseline emission model. Also, the word list we used did not contain any proper nouns — a list of current named entities may improve our results with dictionary modeling.[3]

Basic improvements in the **cache model** may also still be possible, for example by tuning the initial weight of the cache using held-out data. Additionally, the dynamic weighting scheme we used was very focuses on the language model probabilities rather than the effects on word prediction. We may be able to design a dynamic weighting method that better adapts for word prediction rather than language modeling in general.

[3] Though we must be careful to match the time period to our corpora.

We would like to **improve topic and style** by selecting more appropriate units of adaptation, perhaps with better clustering. Additionally, we allowed topic modeling to affect non-topical words, which is responsible for some of the drawbacks of topic adaptation. A better method might only allow topic adaptation to affect the prediction of topical words, following Hsu and Glass (2006). Also, a more principled approach to style adaptation may be beneficial. For example, style might be better modeled as a generative process dependent on a set of lexical features, similar to linguistic analyses of style (Biber and Conrad, 2009).

Our **evaluation** could be improved in a few ways. Notably, when integrating significant linguistic information, the algorithm may predict synonyms of the intended word. In a real system, if a synonym is predicted sooner than the intended word, a user would likely pick the synonym instead. We might be able to evaluate the impact of this by allowing matching based on WordNet synsets or another thesaurus-like resource. Additionally, when typing, not all keys are equally difficult to press — the distance from the previous typing location impacts the speed of selection.[4] For example, selecting the letter L twice is typically faster than selecting the letter L then the letter T. Part of this is due to distance effects, but also some users of AAC devices have a raised overlay grid to more easily select a letter. When typing a double-letter with an overlay, the user may be able to not just take advantage of the reduced movement distance but also their finger is already in the "valley" for the appropriate letter, further reducing selection time. Additionally, evaluation metrics from other fields may help identify problems and benefits more easily, for example mean reciprocal rank (MRR) from information retrieval. However, MRR can be somewhat difficult to apply — if we sort the entire vocabulary, we can easily apply MMR without simulating any typing of words. However, sorting the entire

[4] Thanks to an ACL commenter for this suggestion!

vocabulary is normally impractical, so we would either need to find a smaller number of predictions that is useful for MRR (say 100) or else simulate typing at a given window size (say 10), and record MRR after every letter. Although there are difficulties with MRR, we would like to consider it in the future for developmental evaluations.

Our improvements over a basic HMM-style part of speech model may translate to the field of **part of speech tagging**. For example, we often saw modest improvements in tagging accuracy due to topic modeling because of the improved emission probability distribution. In particular, topic adaptation can help disambiguate in cases where different word senses have distinct part of speech distributions. For example, consider *to/TO study/VB* versus *the/DT user/NN study/NN* — the correct tag can be partially identified by the topic of discourse. Style adaptation may also have a benefit for POS tagging by improving the transition probabilities, though we typically saw smaller changes. Cache modeling might be applicable to part of speech tagging as well, although in our system we did not experiment with cache modeling as part of the model used for Viterbi tagging. Although the dictionary model doesn't benefit POS tagging as-is, it may be applicable for the methods of Cucerzan and Yarowsky (2000). We would like to evaluate our methods in the field of part of speech tagging in the future, especially for domain adaptation.

In contrast to traditional part of speech tagging, however, we would like to move the base system to evaluate in terms of **predictive tagging accuracy** rather than traditional tagging accuracy. The problem with traditional tagging accuracy is that the emission probability model is the most important source of information, so evaluation of tagging accuracy does not necessarily reflect the accuracy of using the model for word prediction. Instead, predictive tagging accuracy more clearly measures what we are trying to accomplish in word prediction. In pilot work, we found that predictive tagging accuracy was significantly lower than traditional

tagging accuracy, which is part of the reason the part of speech model fares poorly compared to the word ngram model.

In **actual AAC devices**, the concept of a document is more fuzzy than typical NLP. We may not have explicit document boundaries, which can cause problems for adaptive methods. Similarly, even within a conversation the topic or style might change over time. Although we designed decay methods to help address this, we can leverage additional information in a real device. Notably we can record the timing of text entry — we expect that topic and style shifts will be accompanied by larger pauses in the typing stream. Firstly, we might be able to automatically segment the text stream based on pause information (and thus reset our topic/style/cache distributions). Secondly, investigation of pauses may reveal additional information. Longer pauses may indicate a larger topical or stylistic shift. For example, it typically takes longer to go from school to home but much less time to go from one classroom to another.

In addition to information about pauses in an actual system, there are other potential sources of information. For example, many AAC devices function like normal computers — they have core software for typing but use different software for email vs web vs reports vs speaking. It may be possible to decide on an initial tuning of the language model based on the current application. Furthermore, stored user texts may be indexed by application, which would lead models such as iterative re-training to have better discrimination between different types of user texts. Also, actual devices may include GPS for location information. It may be possible to build location-specific language models based on real-world distance measures, which could automatically help tune a language model for shopping or the home. However, it may be complicated to spot the topical and stylistic differences between conversations in adjacent classrooms. One way to address the problem would use location information and time information together — the topic in a classroom may

be different at 9am vs 11am or different on Monday at 9am vs Tuesday at 9am.

Real-world AAC devices have a wealth of useful additional information that can help better predict language. Unfortunately, corpora with such information are scarce.

Practitioners are often faced with the task of customizing AAC devices for particular users. For example, **practitioners** may be responsible for **managing the vocabulary** of the AAC device and adding new words to the system. Our methods can help practitioners by learning from texts which match the intended topic but not style or style but not topic. Additionally, we would like to help reduce the workload of practitioners. For example, they may need to add a single word to the dictionary and ensure that it is reasonably predicted, which can be an involved task. If provided the word to add and potentially a sentence of context, our methods can identify the part of speech tag with reasonable accuracy. Then we can estimate the frequency of the word based on word length, following our dictionary methods. The combination of these techniques could allow a practitioner to add new words and have them integrate correctly with the existing predictions.

Appendix A

KEYSTROKE SAVINGS AND WINDOW SIZE

We previously evaluated word prediction at multiple window sizes. The following graphs show a trigram baseline model on Switchboard with word completion (minimum one keystroke to see completions) compared to word prediction (words are predicted without any typed prefix). The results are shown in reference to the theoretical limits.

The evaluation using word *completion* is shown in Figure A.1. Keystroke savings of the trigram model is graphed by window size in reference to the theoretical limit. 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 achieved 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 A.2 alongside word completion. Word prediction achieves much higher keystroke savings, in part due to the higher theoretical limit. We focus on word prediction due to the larger potential for improvement. However, it is important to consider the theoretical keystroke savings limit when analyzing lower keystroke savings than

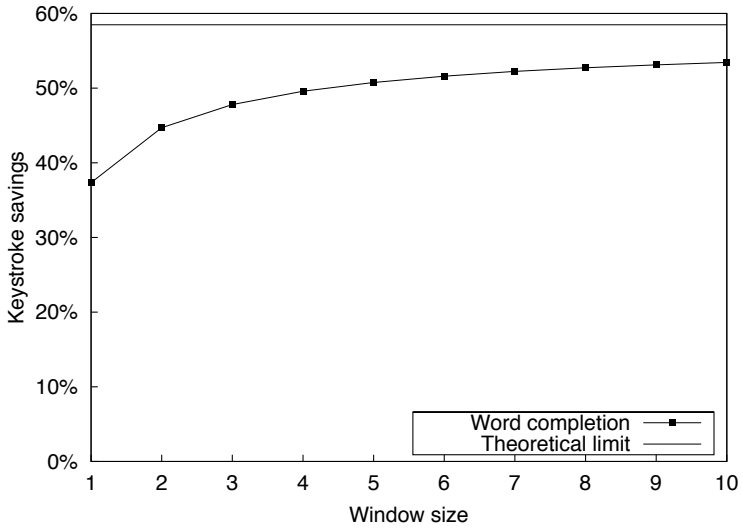


Figure A.1: Keystroke savings vs. window size with word completion. The theoretical limit is shown above the curve.

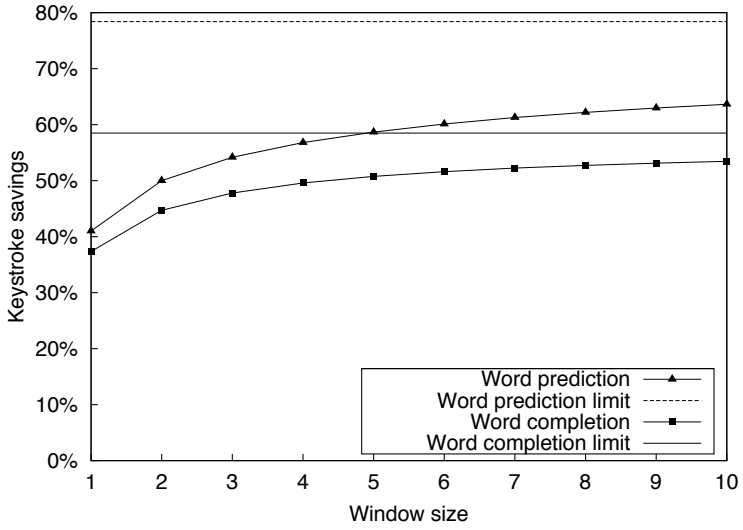


Figure A.2: Keystroke savings vs. window size with word completion and word prediction.

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.

Appendix B

PREVIOUS WORK IN TOPIC MODELING

This appendix presents a more detailed treatment of our previous work in topic modeling compared to Section 3.1. However, due to the necessary background knowledge, some of the information is duplicated below.

B.1 Pure and Hybrid Topic Modeling

One of the problems in topic modeling is deciding how to compute each topic-specific ngram model. 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*,[1] we investigated the use of a trigram model for each topic in (Trnka et al., 2006b) — $P(w \mid w_{-1}, w_{-2}, t)$. Note 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 (1997), Florian and Yarowsky (1999), and Mahajan et al. (1999).

In the second approach, which we call *hybrid topic modeling*,[2] a simplistic language model is computed for each topic. In previous work (Trnka et al., 2006b), we measured a unigram model for each topic. The topic modeling approach in

[1] In earlier work (Trnka et al., 2006b), this was called Method A.

[2] In earlier work (Trnka et al., 2006b), this model was called Method B.

this case creates a topic-adapted unigram model, similar to the LSA-adapted model of Bellegarda (2000, 1998a, 1998b). Because unigrams are generally poor models of language, we combine the topic-adapted unigram model with a baseline ngram model trained on all data.

$$P_{hybrid}(w \mid h) = P_{baseline}(w \mid h) * \left(\sum_{t \in topics} P(t \mid h) * P(w \mid t) \right)^{\alpha} \qquad \text{(B.1)}$$

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 (2000), 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.

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 (Trnka et al., 2006b). 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

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 B.1: In-domain evaluation of pure and hybrid topic modeling on Switchboard. The improvement over the trigram baseline is shown in parentheses for both methods. This evaluation only studies fringe words, not all words.

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 B.1). We conclude that hybrid topic modeling might be more appropriate for devices with limited computational power, whereas pure topic modeling is more appropriate for maximizing keystroke savings.

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. The trigram model rarely produces a full 10 predictions due to sparseness, so the model will typically generate many predictions using the bigram model due to backoff (which will be explained 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 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. Thus 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.

B.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 framework to balance context and data sparseness, following Katz (1987).

Backoff models are a method for combining ngram models of different orders (e.g., trigrams, bigrams, unigrams). 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:

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

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

$$\alpha = 1 - \sum_{w} P_{smooth}(w \mid h) \tag{B.3}$$

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 B.3.3.1.

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 better able to produce predictions, 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. Although an ngram model using only discounting may be applied to speech recognition, such a model is equivalent to an MLE model for word prediction[3]; word prediction requires knowledge of the vocabulary, whereas discounting by itself only fixes issues of zero probabilities for unknown words. Word prediction can only predict words in the vocabulary and is therefore unaffected by the probability estimate of a novel word. 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 (such as the baseline of Section 5.2).

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

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

model to work well) and selecting a smoothing method that is appropriate for both a non-adaptive model as well as a topic model.

B.3 When to Apply Backoff and Smoothing?

Although 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.

B.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:

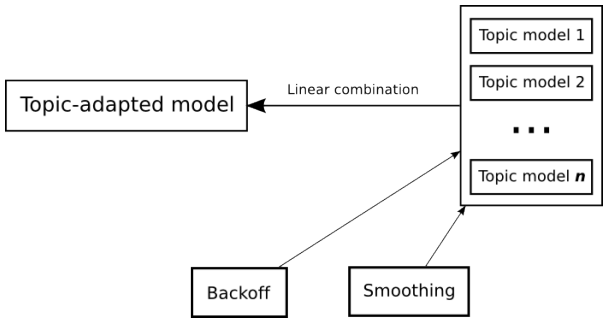


Figure B.1: Illustration of backoff and smoothing in the first implementation option.

Additionally, we can represent this decision mathematically:

$$P(w \mid h) = \sum_{t \in topics} P(t \mid h) * P_{\text{backoff}}(w \mid h, t) \tag{B.4}$$

where $P_{\text{backoff}}(w \mid h, t)$ is Equation B.2 with the additional conditioning information t:

$$P'(w \mid w_{-1}, w_{-2}, t) = \begin{cases} P_{smooth}(w \mid w_{-1}, w_{-2}, t) & if P(w \mid w_{-1}, w_{-2}, t) > 0 \\ \alpha_{w_{-1}, w_{-2}, t} * P_{smooth}(w \mid w_{-1}, t) & if P(w \mid w_{-1}, t) > 0 \\ \alpha_{w_{-1}, t} * \alpha_{w_{-1}, w_{-2}, t} * P_{smooth}(w \mid t) & otherwise \end{cases} \tag{B.5}$$

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. 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 of 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.

B.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.

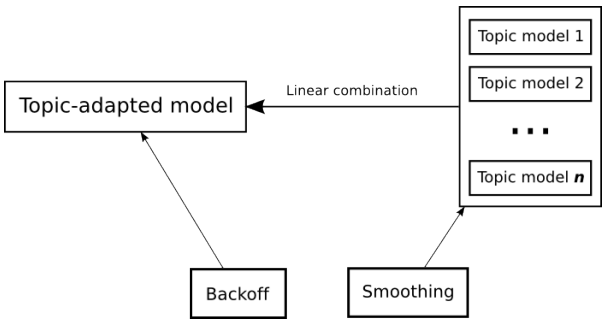


Figure B.2: Illustration of backoff and smoothing in the second implementation option.

$$P_{\text{smooth}}(w \mid w_{-1}, w_{-2}) = \sum_{t \in topics} P(t \mid h) * P_{\text{smooth}}(w \mid w_{-1}, w_{-2}, t) \tag{B.6}$$

$$P_{\text{smooth}}(w \mid w_{-1}) = \sum_{t \in topics} P(t \mid h) * P_{\text{smooth}}(w \mid w_{-1}, t) \tag{B.7}$$

$$P_{\text{smooth}}(w) = \sum_{t \in topics} P(t \mid h) * P_{\text{smooth}}(w \mid t) \tag{B.8}$$

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:

$$P'(w \mid w_{-1}, w_{-2}) = \begin{cases} P_{smooth}(w \mid w_{-1}, w_{-2}) & if P(w \mid w_{-1}, w_{-2}) > 0 \\ \alpha_{w_{-1}, w_{-2}} * P_{smooth}(w \mid w_{-1}) & if P(w \mid w_{-1}) > 0 \\ \alpha_{w_{-1}} * \alpha_{w_{-1}, w_{-2}} * P_{smooth}(w) & otherwise \end{cases} \tag{B.9}$$

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).

B.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.

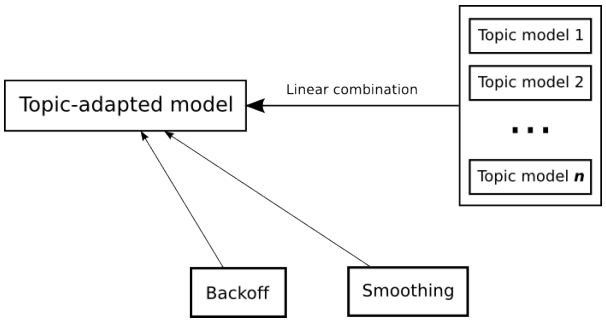


Figure B.3: Illustration of backoff and smoothing in the third implementation option.

$$f(w \mid w_{-1}, w_{-2}) = \sum_{t \in topics} P(t \mid h) * f(w \mid w_{-1}, w_{-2}, t) \tag{B.10}$$

$$f(w \mid w_{-1}) = \sum_{t \in topics} P(t \mid h) * f(w \mid w_{-1}, t) \tag{B.11}$$

$$f(w) = \sum_{t \in topics} P(t \mid h) * f(w \mid t) \tag{B.12}$$

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

$$P'(w \mid w_{-1}, w_{-2}) = \begin{cases} P_{smooth}(w \mid w_{-1}, w_{-2}) & if P(w \mid w_{-1}, w_{-2}) > 0 \\ \alpha_{w_{-1},w_{-2}} * P_{smooth}(w \mid w_{-1}) & if P(w \mid w_{-1}) > 0 \\ \alpha_{w_{-1}} * \alpha_{w_{-1},w_{-2}} * P_{smooth}(w) & otherwise \end{cases} \tag{B.13}$$

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 B.6). In addition, because large corpora are generally unavailable for AAC, 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. (1999) 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.

B.3.3.1 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.

B.3.3.2 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:

$$\sum_w f'_{topic}(w \mid h) = \alpha * \sum_w f_{topic}(w \mid h) = \sum_{t \in topics} \sum_w f(w \mid h, t) \tag{B.14}$$

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.

B.4 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 (Witten, 1991) for simplicity, following these equations:

$$P(w \mid h) = (1 - P(unseen \mid h)) * \frac{f(w \mid h)}{f(h)} \tag{B.15}$$

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:

$$P(unseen \mid h) = \frac{\mid \{w \mid f(w \mid h) > 0\} \mid}{f(h)} \tag{B.16}$$

We later switched to the smoothing used in Katz' backoff[4] for high performance on the baseline trigram model (Katz, 1987). The smoothing in Katz' backoff requires the entire trigram distribution to be smoothed at once.[5] 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 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 (Good, 1953), but without the requirement of a

[4] 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 component.

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

somewhat non-sparse distribution[6].

$$P(w \mid h) = \frac{f(w \mid h)}{f(w \mid h) + \lambda} * \frac{f(w \mid h)}{f(h)} \tag{B.17}$$

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:

$$\alpha_h = 1 - \sum_w P(w \mid h) \tag{B.18}$$

$$P'(w \mid h) = \begin{cases} P(w \mid w_{-1}, w_{-2}) & if P(w \mid w_{-1}, w_{-2}) > 0 \\ \alpha_{w_{-1},w_{-2}} * P(w \mid w_{-1}) & if P(w \mid w_{-1}) > 0 \\ \alpha_{w_{-1}} * \alpha_{w_{-1},w_{-2}} * P(w) & otherwise \end{cases} \tag{B.19}$$

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 B.2 for early experiments on the Switchboard corpus without topic modeling.

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

[6] 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 (Gale and Sampson, 1995) 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.

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 B.2: 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.

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.

B.5 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, which we will refer to as a *(topical) relevance score*.

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.

B.5.1 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. This model, although simplistic, will weight the importance of words in proportion to how often the speaker or author uses them. However, the document cache can be improved by taking recency and topical salience into account.

B.5.2 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. (1997), who found that the probability of lexical repetition could be modeled using an exponential function of the recency of the word. We follow Bellegarda (2000) in using an exponentially decayed cache to model this effect of recency on importance at the current position in the document. In an exponential decay model, λ is a tunable decay weight. We chose $\lambda = 0.95$ following Bellegarda (2000). 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 (Bellegarda, 2000).

B.5.3 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 B.6 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.

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.

B.5.4 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 different functions to model $P(t \mid h)$ and then describe manipulations of the relevance functions to improve topic modeling.

B.5.4.1 Relevance Functions

Our initial work in approximating $P(t \mid h)$ used distributional similarity measures along with normalization (see (Lee, 1999) 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:

$$P(t \mid h) \approx \frac{sim(t, c)}{\sum_{t'} sim(t', c)} \tag{B.20}$$

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.

$$sim_{cosine}(t,c) = \frac{\sum_{w \in t \cap c} f_t(w) * f_c(w)}{\sqrt{\sum_{w \in t} f_t(w)^2} * \sqrt{\sum_{w \in c} f_c(w)^2}} \tag{B.21}$$

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 (Seymore and Rosenfeld, 1997, Lee, 1999), 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 (1999) and Naïve Bayes, which gave better results than cosine for Seymore and Rosenfeld (1997) for topic modeling.

The Jacquard coefficient is shown below. This method, although simplistic,

was found to be the best previously existing technique by Lee (1999) for differentiating between a artificial and natural verb-object pairs.

$$sim_{Jacquard}(t, c) = \frac{\mid t \cap c \mid}{\mid t \cup c \mid} \tag{B.22}$$

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).

$$P(t \mid h) = P(t) * \prod P(w \mid t)^{f_h(w)} \tag{B.23}$$

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:

$$\log(P(t \mid h)) = \frac{\log(P(t)) + \sum f(w) * \log(P(w \mid t))}{\sum f(w)} \tag{B.24}$$

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 B.5.4.2 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 B.3: 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 and was found to reduce perplexity slightly over TF-IDF for topic modeling for Seymore and Rosenfeld (1997). 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 cosine 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.

B.5.4.2 Score Scaling

One of the variations explored in Florian and Yarowsky (1999) 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.

$$sim'(t,h) = \left(\frac{sim(t,h) - min_{t'}(sim(t',h))}{max_{t'}(sim(t',h)) - min_{t'}(sim(t',h))} \right)^2 \tag{B.25}$$

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, Florian and Yarowsky (1999) found that squaring the score only reduced the perplexity slightly beyond the scaling, so we only perform the scaling and not the squaring:

$$sim'(t,h) = \frac{sim(t,h) - min_{t'}(sim(t',h))}{max_{t'}(sim(t',h)) - min_{t'}(sim(t',h))} \tag{B.26}$$

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 B.3), 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.

B.5.4.3 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:

$$sim'(t,h) = \frac{sim(t,h) + \gamma * min_{t'|sim(t',h)>0}(sim(t',h))}{max_{t'}(sim(t',h)) + \gamma * min_{t'|sim(t',h)>0}(sim(t',h))} \tag{B.27}$$

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 human-labeled topics (Switchboard). 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 B.6), 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 that similarity smoothing is generally beneficial

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 B.4: 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.

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.

B.5.4.4 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 B.6), 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 verbs may be associated with one topic while present tense 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.

B.6 What's in a Topic?

Our approach to topic modeling expects training data that is segmented by topic. Documents in some corpora are manually assigned to topic labels, such as "air pollution" or "fishing" in Switchboard (Godfrey et al., 1992), which can be used for language modeling (Trnka et al., 2006b, Lesher and Rinkus, 2001). Other researchers use topics of a similar type, but rely on automatic document clustering rather than manual clustering of topics (Florian and Yarowsky, 1999). However, other researchers have used each individual document as a sort of topic (Mahajan et al., 1999, Seymore and Rosenfeld, 1997). 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 2.3. 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).

B.6.1 Medium-grained Topics (Human-annotated from Switchboard)

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 B.5.

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

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 B.5: 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

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.

B.6.1.1 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 B.6 and out-of-domain shown in Table B.7.

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 B.6: 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 B.7: 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 B.6) shows improvement in some cases and a loss of keystroke savings in other cases. However, for out-of-domain evaluation (Table B.7), 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.

B.6.1.2 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 B.8, out-of-domain evaluation in Table B.9, and mixed-domain evaluation in Table B.10.

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 B.8: 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

In-domain evaluation (Table B.8) shows that fine-grained topic modeling significantly improves keystroke savings for all corpora except Callhome. Out-of-domain evaluation (Table B.9) 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 B.10) 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

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 B.9: 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

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 B.10: 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

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 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.

B.6.2 Nearest-neighbors Approaches

Fine-grained topic modeling can be used in conjunction with a nearest-neighbors approach. Mahajan et al. (1999) 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 B.4.

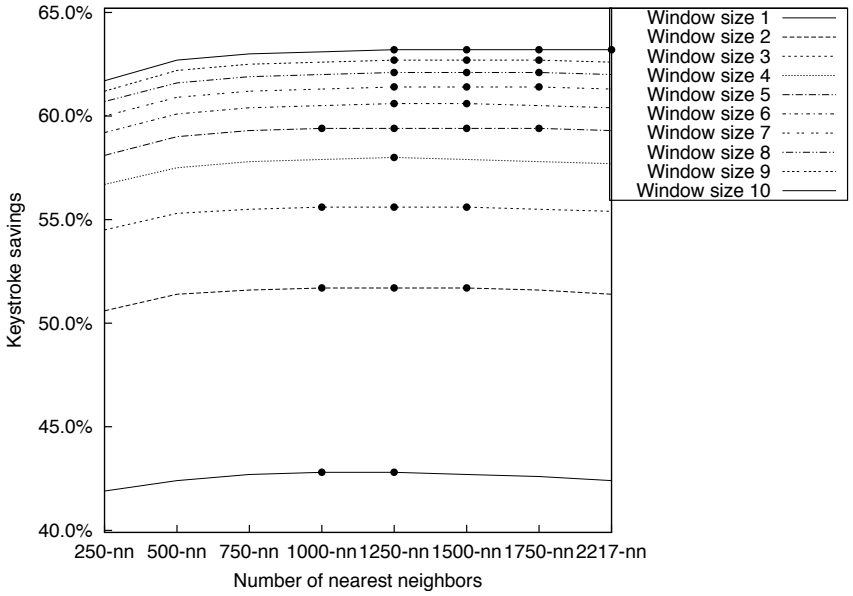


Figure B.4: Keystroke savings across window sizes at different numbers of nearest neighbors. The maximum value for each window size is shown with a dot. When multiple window sizes have the same value to one decimal place, there are multiple dots per line.

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 B.5 for more details). Not only did stemming improve fine-grained

topic modeling, but it changed the nearest neighbors trend. Figure B.5 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

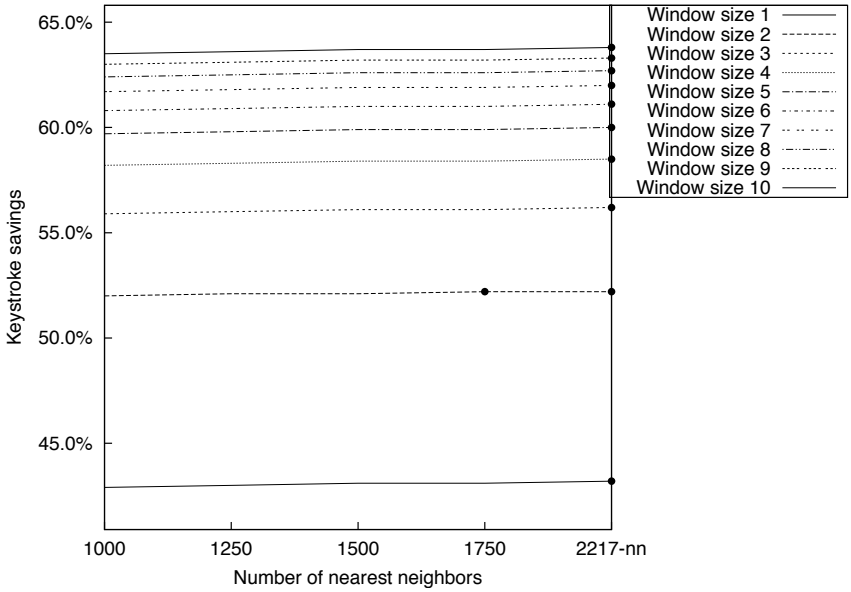


Figure B.5: Keystroke savings at across window sizes at different numbers of nearest neighbors using stemming. The maximum value for each window size is shown with a dot. When multiple window sizes have the same value to one decimal place, there are multiple dots per line.

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 similarity metric is

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

Appendix C

CASE-STUDY STYLE ANALYSIS

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
- Penn Treebank part-of-speech (POS) unigrams
- Penn Treebank part-of-speech (POS) bigrams
- classed Penn Treebank[1] part-of-speech (POS) unigrams
- classed Penn Treebank part-of-speech (POS) bigrams

C.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

[1] The Penn Treebank tags were placed into more general classes, shown in Table C.1. For example, all variations of pronouns and different nouns were grouped together under the general noun tag and likewise for verbs, adjectives, and adverbs.

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 C.1: Penn Treebank part of speech tags are shown in the left columns and the reduced POS tags are shown in the right columns.

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[2]. For more coarse-grained POS classes, we used the mapping of POS tags shown in Table C.1.

C.2 Methods

We tagged the corpora and measured the different distributions (word unigram, POS unigram, POS bigram, etc.). Then we performed comparative analysis of the distributions. For example, a small portion of the word unigram comparison is shown in Figure C.1. 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 ($d < 0.5$)

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

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 only present analysis of frequent words to avoid over-fitting. Words that were both frequent and different in a corpus were considered to be 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.

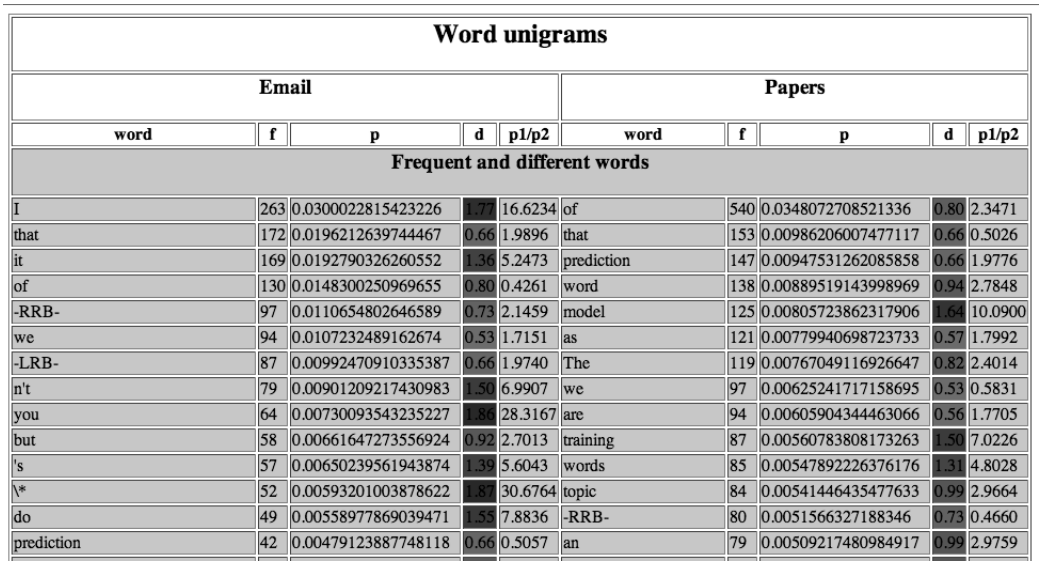
We also performed this analysis for POS unigrams and bigrams in addition to words, treating a POS tag as a word for POS unigrams and a POS tag pair as a word for POS bigrams. The two POS analyses were run using both the standard Treebank tags as well as more general tags. In the total four POS analyses, 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/.

C.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[3] of emails and papers are shown in Table C.2. Each token is presented with the ratio of the in-domain probability (i.e., email for email, papers for papers) 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.

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



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 C.1: Screenshot of comparison between email word unigrams (left) and paper word unigrams (right)

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 (e.g., "however", "so") and pronouns, would comprise the majority of the differences. Indeed, the analysis shows a difference in closed-class words, such as the higher likelihood of pronouns (typical of informal text) and contractions in papers.[4]

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 any

[4] 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.

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 C.2: Word unigrams

To see whether more structural differences existed between the two text types, we also investigated differences between unigram distributions of part-of-speech (POS) tags. Table C.3 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 are more likely in emails whereas past participles are more likely in papers. Bracketing is much more likely in emails, most likely because frequent parentheses are frowned upon in formal writing. Symbols are similar — it's common in email to have things like "svn add *.tex", but those symbols are less likely in formal text. 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 participles in papers. In addition, foreign words (e.g., Latin) are more likely in papers, which is typical of formal academic writing. Grouping the parts of speech into more general classes 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 C.3: 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 C.4. Papers seem to have more modified nouns than email —

both noun-noun[5] and adjective-noun[6] pairs are more likely in papers. There were also more strings of multiple adjectives in papers[7]. 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 C.4: Coarse-grained part of speech bigrams

[5] "word prediction" and "keystroke savings" were the two most frequent in papers.

[6] "basic prediction", "in-domain training" and "out-of-domain training" were the 3 most frequent in papers

[7] Examples include "several conversational", "significant cognitive", "much larger", and "most common"

Appendix D

PREVIOUS WORK IN CACHE MODELING

D.1 Named Entity Caching

We implemented and evaluated the named entity cache used in Li and Hirst (2005) — 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. The named entity cache 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 D.1 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 (2005), 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 (1990) model that is used for the NNP-based POS tags, if the weight on the cache component were very high. 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 helped motivate our cache model in Chapter 5.

	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 D.1: 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.

D.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 (Carlberger, 1998, Väyrynen, 2005). 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 (1998) and Väyrynen's (2005) 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 D.2. 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 prediction model at all, and because the predictions are always below those of the trigram model, the addition of this model

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 D.2: 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.

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.

BIBLIOGRAPHY

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

E.L. Antworth. 1990. PC-KIMMO: a two-level processor for morphological analysis. *Occasional Publications in Academic Computing*, 16.

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

Marco Baroni, Johannes Matiasek, and Harald Trost. 2002. Predicting the Components of German Nominal Compounds. In *European Conference on Artificial Intelligence (ECAI)*, pages 470–474.

C. Beck, G. Seisenbacher, G. Edelmayer, and WL Zagler. 2004. First user test results with the predictive typing system FASTY. *ICCHP*, pages 813–819.

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

J. R. Bellegarda. 1998a. 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.

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

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

Jerome R. Bellegarda. 2004. Statistical language model adaptation: review and perspectives. *Speech Communication*, 42(1):93–108, January.

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

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

Douglas Biber and Susan Conrad. 2009. *Register, Genre, and Style*. Cambridge University Press.

Douglas Biber. 1988. *Variation Across Speech and Writing*. Cambridge University Press.

D.M. Blei, A.Y. Ng, and M.I. Jordan. 2003. Latent dirichlet allocation. *The Journal of Machine Learning Research*, 3:993–1022.

J. Blitzer, R. McDonald, and F. Pereira. 2006. Domain Adaptation with Structural Correspondence Learning. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Thorsten Brants and Alex Franz. 2007. *Web 1T 5-gram Version 1*. Linguistic Data Consortium, Philadelphia.

T. Brants. 2000. TnT: a statistical part-of-speech tagger. In *Proceedings of the sixth conference on Applied natural language processing*, pages 224–231.

Eric Brill, Radu Florian, John C. Henderson, and Lidia Mangu. 1998. Beyond n-grams: Can linguistic sophistication improve language modeling? In *Annual Meeting of the Association for Computational Linguistics*, pages 186–190, Montreal, August.

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

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

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

Stanley Chen, Kristie Seymore, and Ronald Rosenfeld. 1998a. 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.

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

P. R. Clarkson and A. J. Robinson. 1997. 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.

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

Ann Copestake. 1997. 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.

Silviu Cucerzan and David Yarowsky. 2000. Language independent, minimally supervised induction of lexical probabilities. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, ACL '00, pages 270–277, Morristown, NJ, USA. Association for Computational Linguistics.

Hal Daumé III. 2007. Frustratingly easy domain adaptation. In *Conference of the Association for Computational Linguistics (ACL)*, Prague, Czech Republic.

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

Nigel Dewdney, Carol VanEss-Dykema, and Richard MacMillan. 2001. 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.

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

C. DiMarco, G. Hirst, and M. Stede. 1993. 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.

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

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

Afsaneh Fazly and Graeme Hirst. 2003. 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.

A. Fazly. 2002. The Use of Syntax in Word Completion Utilities. Master's thesis, University of Toronto.

Ol'ga Feiguina and Graeme Hirst. 2007. Authorship attribution for small texts: Literary and forensic experiments. In *Proceedings, International Workshop on Plagiarism Analysis, Authorship Identification, and Near-Duplicate Detection, 30th Annual International ACM SIGIR Conference (SIGIR '07)*, Amsterdam, July.

Radu Florian and David Yarowsky. 1999. 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.

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

Nestor Garay-Vitoria and Julio Abascal. 2004. A comparison of prediction techniques to enhance the communication rate. *User-Centered Interaction Paradigms for Universal Access in the Information Society*, pages 400–417.

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

Nestor Garay-Vitoria and Julio González-Abascal. 1997. Intelligent word-prediction to enhance text input rate. In *International Conference on Intelligent User Interfaces (IUI)*, pages 241–244, Orlando, Florida.

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

J.J. Godfrey, E.C. Holliman, and J. McDaniel. 1992. Switchboard: telephone speech corpus for research and development. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 1, pages 517–520, March.

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

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

J.T. Goodman. 2001a. A bit of progress in language modeling. *Computer Speech & Language*, 15(4):403–434.

J.T. Goodman. 2001b. A Bit of Progress in Language Modeling Extended Version. Technical report, Microsoft Research Technical Report MSR-TR-2001-72.

Neil Graham, Graeme Hirst, and Bhaskara Marthi. 2005. Segmenting documents by stylistic character. *Natural Language Engineering*, 11(4):397–415, December.

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

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

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

Graeme Hirst and Ol'ga Feiguina. 2007. Bigrams of syntactic labels for authorship discrimination of short texts. *Literary and Linguistic Computing*, 22(4):405–417.

Thomas Hofmann. 1999. Probabilistic Latent Semantic Analysis. In *Uncertainty in Artificial Intelligence (UAI)*, pages 289–296. Citeseer.

Eduard Hendrik Hovy. 1987. *Generating Natural Language Under Pragmatic Constraints*. Ph.D. thesis, Yale University, March.

B.J.P. Hsu and J. Glass. 2006. Style & topic language model adaptation using HMM-LDA. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 373–381. Association for Computational Linguistics.

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

F. Jelinek, B. Merialdo, S. Roukos, and M. Strauss. 1991. 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.

Frederick Jelinek. 1991. Up from trigrams! - the struggle for improved language models. In *Eurospeech*, pages 1037–1040, Genova, Italy, September.

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

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

Slava M. Katz. 1987. 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.

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

R. Kneser and V. Steinbiss. 1993. 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.

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

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

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

R. Lau, R. Rosenfeld, and S. Roukos. 1993. 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.

Yong-Bae Lee and Sung Hyon Myaeng. 2002. 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.

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

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

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

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

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

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

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

Milind Mahajan, Doug Beeferman, and X. D. Huang. 1999. 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.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: the penn treebank. *Computational Linguistics*, 19(2):313–330.

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

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

David McClosky. 2010. *Any Domain Parsing: Automatic Domain Adaptation for Parsing*. Ph.D. thesis, Brown University, May.

S. E. Michos, E. Stamatatos, N. Fakotakis, and G. Kokkinakis, 1996a. *Categorising Texts by Using a Three-Level Functional Style Description*, pages 191–198. IOS Press.

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

Andrei Mikheev. 2000. Document centered approach to text normalization. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '00, pages 136–143, New York, NY, USA. ACM.

J.E. Miller, M. Torii, and K. Vijay-Shanker. 2007. Building Domain-Specific Taggers Without Annotated (Domain) Data. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 1103–1111.

Alan F. Newell, John L. Arnott, Lynda Booth, William Beatie, Bernadette Brophy, and Ian W. Ricketts. 1992. Effect of the "PAL" Word Prediction System on the Quality and Quantity of Text Generation. *Augmentative and Alternative Communication*, 8(4):304–311, December.

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

T.R. Niesler and P.C. Woodland. 1996. Combination of word-based and category-based language models. In *International Conference on Spoken Language Processing*, volume 1, pages 220–223, oct.

Fuchun Peng, Dale Schuurmans, and Shaojun Wang. 2003. 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.

Randi Reppen, Nancy Ide, and Keith Suderman. 2005. ANC Second Release. Accessed from http://americannationalcorpus.org/SecondRelease/ around 11/2010.

Ronald Rosenfeld. 1994. 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.

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

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

H. Schmid. 1994. Probabilistic part-of-speech tagging using decision trees. In *Proceedings of International Conference on New Methods in Language Processing*, volume 12.

S. Schwarm and M. Ostendorf. 2002. Text normalization with varied data sources for conversational speech language modeling. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 1.

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

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

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

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

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

Michael Stubbs, 1986. *Language Development, Lexical Competence and Nuclear Vocabulary.* Croom Helm.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *NAACL '03: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 173–180, Edmonton, Canada. Association for Computational Linguistics.

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

Keith Trnka and Kathleen F. McCoy. 2008. Evaluating Word Prediction: Framing Keystroke Savings. In *ACL-08: HLT, Short Papers (Companion Volume)*, pages 261–264, June.

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

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

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

Keith Trnka, John McCaw, Debra Yarrington, Kathleen F. McCoy, and Christopher Pennington. 2008. Word Prediction and Communication Rate in AAC. In *Telehealth and Assistive Technologies (Telehealth/AT)*, pages 19–24, April.

Keith Trnka, John McCaw, Debra Yarrington, Kathleen F. McCoy, and Christopher Pennington. 2009. User Interaction with Word Prediction: The Effects of Prediction Quality. *ACM Transactions on Accessible Computing (TACCESS)*, 1(3):1–34.

Keith Trnka. 2008a. Adapting Word Prediction to Subject Matter without Topic-labeled Data. In *ACM SIGACCESS Conference on Computers and Accessibility (ASSETS)*, pages 315–316, Halifax, Nova Scotia, Canada, October 13–15.

Keith Trnka. 2008b. Adaptive language modeling for word prediction. In *ACL-08: HLT Student Research Workshop (Companion Volume)*, pages 61–66, June.

Keith Trnka. 2008c. Word Prediction Techniques for User Adaptation and Sparse Data Mitigation. In *University of Delaware PhD Thesis Proposal*, May.

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

Pertti Väyrynen. 2005. *Perspectives on the utility of linguistic knowledge in English word prediction*. Ph.D. thesis, University of Oulu.

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

Tonio Wandmacher and Jean-Yves Antoine. 2006. 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.

T. Wandmacher and J.Y. Antoine. 2007. 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.

Tonio Wandmacher, Jean-Yves Antoine, and Franck Poirier. 2007. 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.

Tonio Wandmacher. 2009. *Adaptive word prediction and its application in an assistive communication system.* Ph.D. thesis, Université François – Rabelais de Tours.

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

J.O. Wobbrock and B.A. Myers. 2006. From letters to words: efficient stroke-based word completion for trackball text entry. *ACM SIGACCESS Conference on Computers and Accessibility (ASSETS)*, pages 2–9.

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