NORMALIZATION OF INFORMAL TEXT FOR TEXT-TO-SPEECH

by

Deana L. Pennell

Dedicated to the memory of

Larry King

September 17, 1945 – October 1, 2010

"Good teaching is more

a giving of right questions

than a giving of right answers."

–*Josef Albers*

NORMALIZATION OF INFORMAL TEXT FOR TEXT-TO-SPEECH

by

DEANA L. PENNELL, B.S., M.S.

DISSERTATION

Presented to the Faculty of

The University of Texas at Dallas

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY IN

COMPUTER SCIENCE

THE UNIVERSITY OF TEXAS AT DALLAS

December 2011

PREFACE

ACKNOWLEDGMENTS

I would like to take this opportunity to thank the many people without whom this dissertation would not have been possible.

First and foremost, my utmost thanks goes to my adviser, Dr. Yang Liu. Without her, I would have been lost in the world of research. I can safely say that none of my prior schooling experiences prepared me in any way for the realities of pursuing a doctoral degree. With her guidance, I was able to grow as a scholar and complete a dissertation that I can be proud of. I know that I was not always the most diligent student, but she graciously accepted my faults as well as my strengths and pushed me to reach higher and live up to my capabilities. Seeing her successes, her motivation, and the obvious enjoyment she draws from her work kept me going at times when I felt tempted to give up and move on. I feel honored and privileged to have had the opportunity to work hand-in-hand with such an inspiring woman.

I also need to extend my thanks to my committee members: Dr. Vincent Ng, Dr. John Hansen and Dr. Haim Schweitzer. Thank you all for taking the time out of your very busy schedules to provide guidance and valuable feedback on my work.

My lab mates, past and present, are also very deserving of my thanks for their constructive criticism of my work, suggestions, and general back and forth intellectual discussion. My gratitude is extended to Je Hun Jeon, Melissa Sherman, Fei Liu, Shasha Xie, Dong Wang, Rui Xia, Zhonghua Qu, Bin Li, Thamar Solorio, Feifan Liu, Keyur Gabani and Nisa Hassanali. Special thanks are extended to Justin Schneider and Duc Le for their work on the message selection for annotation described in Section 3.3 in this dissertation, and for various scripts written along the way.

Thanks as well to the various others who have played parts both small and large throughout my days at UTD. Mark Hittenger and Brian Nelson are especially deserving of my thanks for putting up with the myriad questions I asked to which I should have already known the answers. To Eric Moden and Judy Patterson for dealing with my lack of knowledge about deadlines and requirements for so many years. To all the staff in the Computer Science Head office, for completing the various small tasks I presented you with a smile and for keeping track of my paychecks for me.

A big thank you goes to all of my co-workers at SCALE 2010 for providing a sounding board for my ideas and providing me with valuable feedback. Special thanks are extended to Spence Green, Paul McNamee, Stephanie Poisson and Chris Callison-Burch for extended discussions that led directly to certain aspects of the work in this dissertation.

To my family, who have always been extremely supportive of my decision to pursue a doctoral degree even though it meant leaving Maine. My parents in particular encouraged me to pursue any and all interests and to dream big and my brother Trevor played devil's advocate to my early attempts at research (a.k.a. my "brilliant attempts" to solve all of the world's problems before finishing middle school). And to my friends, without whom my sanity would have been lost long ago; who helped me take my mind off work when it was needed most; who graciously listened to me both when I was overly excited about results and wouldn't stop spouting new ideas and when I was frustrated and unmotivated and wouldn't stop complaining: Lara Graham, Jennifer Bates, Barbara Crawford, Melissa Masucci, Ehsan Nourbakhsh, Vitali and Sahar (Salari) Loseu and Paul Johnson. Thanks also to the CHAMPS fellows with whom I had the opportunity to get middle school children excited about education and technology, and to help each other grow as teachers. Robert Anderson, Anastasia Kurdia and Steven Kirtzic, who I collaborated with on lesson plans and who helped me refine my abilities. Special mention should be made for my new friends and acquaintances at the Arthur Murray Dance Studio in Plano for providing me with a fun and healthy way to unwind from the stresses of graduate school; in particular, our teacher Lindsay Jordan for understanding that sometimes working on my dissertation was more important than practicing the last week's lesson!

Finally, my amazing husband, Ryan – without you, formatting this dissertation would have taken me many more years! Words cannot begin to express my heartfelt gratitude, appreciation and thanks for all you have done to support and encourage me on this journey. Throughout our years at UTD, together we have grown as students and as people. Thinking back on our time together, from those first meetings in Venky's class where you were his protogé and I was that girl who always showed up halfway through class (and eventually convinced the professor to start class half hour later, just for her!), to the nights spent frantically finalizing paper drafts and generating final rounds of dissertation results at Flying Saucer in between rounds of Quelf, it seems like there isn't anything we haven't done together. And now that we've both written dissertations, finished graduate school and gotten excellent jobs lined up, it seems like there is nothing we can't accomplish. Together. The last five years have been amazing. Let's make the next five even better.

September 2011

NORMALIZATION OF INFORMAL TEXT FOR TEXT-TO-SPEECH

Publication No. ____________________

Deana L. Pennell, Ph.D.
The University of Texas at Dallas, 2011

Supervising Professors: Dr. Yang Liu, Co-Chair
Dr. Vincent Ng, Co-Chair

A large amount of information is found in noisy contexts as texting and chat lingo have become increasingly prolific in the past decade. To take advantage of this vast knowledge resource, natural language processing techniques need to be adapted to work accurately on this unconventional data. This dissertation describes various noisy-channel approaches for the normalization of informal text, such as that found in emails, chat rooms, and SMS messages. In particular, two methods for the abbreviation modeling aspect of the noisy channel model are introduced: a statistical classifier using language-based features to decide whether a character is likely to be removed from a word, and a character-level machine translation model. A two-phase approach is used; in the first stage the possible candidates are generated using the selected abbreviation model and in the second stage the best candidate is chosen through decoding using a language model. Various combinations of the two methods are explored. The model is finally extended to decrease the number of false positives by using dictionary heuristics. Normalization accuracy for all systems is presented as well as both quantitative and qualitative results showing the effects of normalization on text-to-speech output. A significant decrease in word and phoneme error rates is achieved as well as an improvement shown using human perceptual tests. These contributions reflect an important step toward the development of improved text for automatic speech synthesis applications driven by informal text domain sources.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

Texting and chat lingo has become increasingly prolific in the past decade. According to the CTIA's semi-annual report, over 1.5 trillion text messages were sent during the 2009 calendar year (approximately 150% increase from 2008), with almost 5 billion messages per day sent by the end of the year; an increase of 33% saw 1.8 trillion SMS messages sent between June 2009 and June 2010 (CTIA, 2009, 2010a,b). Unfortunately, this popularity has lead to a large increase in automobile accidents caused by people who are texting while driving, which is gaining extensive media attention. The National Security Council estimates that over 200,000 accidents in 2008 were caused by texting or emailing while driving (NSC, 2010), and while statistics are not yet available, it is expected that those numbers have risen with the growth of texting popularity. The website `http://txtresponsibly.org/` was designed to educate drivers about the dangers of reading and writing text messages while driving. In April 2010, Oprah Winfrey promoted making vehicles a "No Phone Zone" (Winfrey, 2010), with a country-wide publicity campaign on April 30. Although we realize the safest alternative is to prevent phone usage altogether while driving, it is probably unrealistic to believe people will completely ignore their phones. We hope that by developing a text-to-speech (TTS) system for text messages, we can provide a somewhat safer, though admittedly imperfect, solution.

When using a standard TTS system to read text messages, many problems arise due to phenomena in text messages, *e.g.*, use of abbreviations, emoticons and informal capitalization and punctuation. These problems also exist in other domains, such as blogs, forums, social network websites, chat rooms, message boards, and communication between players in online video game chat systems. For lack of a formal definition for the domain of our data, we will use the term *"informal text"* to refer to the type of writing with which we work. There is an increasing volume of information contained in informal text due to a rapid increase in popularity of the writing style. This has lead to a rapidly growing need for efficient processing and applications such as summarization, topic classification, sentiment analysis and user behavior modeling, just to name a few. Traditional natural language processing (NLP) techniques perform poorly on informal text because they are typically trained using formal written text whereas text in this domain is written in a casual, speech-like style and contains many non-standard words.

Industry is becoming interested in this domain, especially with regard to micro-blogging sites such as Twitter (`http://www.twitter.com`). The Library of Congress has archived all public status messages from Twitter from the site's inception in 2006 until the present, and will continue to do so (Gross, 2010). Google has recently released many applications for their "real-time search" results, which return current posts on Twitter and other micro-blogging sites along with typical search results (Singhal, 2009; Wise, 2010; Casey, 2010). Twitter itself provides a list of "trending topics" and promotes many user-designed applications to enhance other user's experiences providing search, recommendations for users to follow, among many other features. By addressing the challenges presented by the domain of informal text, we are paving the way not only for improved TTS output for text messages, but also for improved applications for many other sources of information.

1.1 Domain-specific Challenges for NLP

The informal text studied in this work contains features that make it very different from either formal written text or text annotated from speech data. These differences mean that current techniques for many NLP tasks perform quite poorly on data from this domain. Not all differences are detrimental, though. Some features of this domain present useful information that is often difficult to obtain from text alone, such as stress and emotion. This is because writers in this domain often phrase messages to reflect the way they speak, which is generally not true of formal text. A description of the main differences making this domain difficult follows. The descriptions focus on the effect of these differences on text-to-speech (TTS) applications because this is the application where most of our proposed work is concentrated; however, brief descriptions of problems associated with other applications are also listed.

1. **Texting abbreviations.** The greatest challenge when working with informal text is the large number of out-of-vocabulary words arising from the prolific use of abbreviations. These abbreviations have their roots in the first instant messaging (IM) systems and multi-player video games where it was important to communicate quickly over the internet despite a lack of touch-typing skill. As text messaging via the short message service (SMS) on cell phones became increasingly common, people began using abbreviations more frequently to keep their messages below the 160 character limit to save money. Abbreviations are used inconsistently, even by an individual user, and new abbreviations are constantly being invented, preventing us from simply creating a dictionary of all abbreviations. These abbreviations pose serious problems for many useful NLP applications.

   Chat abbreviations have a detrimental effect on text-to-speech (TTS) systems because it would be impossible to store pronunciations in the lexicon for all possible abbreviations. Although it is possible to ignore the unknown word and just leave a silent gap in the utterance, it is generally better to at least guess a pronunciation. TTS systems may either try to pronounce them using letter-to-phoneme (L2P) rules or spell them out character by character. For some classes of abbreviations, one or the other of these approaches produces an understandable utterance, e.g. “nite” is easily pronounced as the correct word “night” using L2P rules and “ne” is understandable as “any” when the letters are spelled out. The abbreviation “abt” for “about”, however, does not lend itself well to either of these approaches. Abbreviations are constantly changing and a single abbreviation can stand for multiple words or phrases, so building a mapping between abbreviations and words is not a feasible solution.

   Search applications are also hindered by abbreviated text because they may not return relevant results when the search term has been abbreviated in the conversation, status message, blog or forum in which it occurs. Users cannot possibly search for all possible abbreviations or misspellings of a word in order to find the post they are looking for.

Recently, detection of topics, keywords and sentiment is being performed on blogs and Twitter status messages in an attempt to get real-time information about how people feel about a given person, subject, current event, political candidate, or even a television show that is currently being broadcast (Bollen et al., 2009; Diakopoulos and Shamma, 2010; Tumasjan et al., 2010; Pak and Paroubek, 2010). Because these applications still rely heavily on the words used in the text, they are negatively affected by abbreviations, which do not appear in the lexicons used.

2. **Emoticons and fillers.** An emoticon is a textual representation of a facial expression. Informal text is usually stream of consciousness and written in one pass without the multiple drafts and editing sequences that are common for formal domains. In this way, informal text is much like the spoken word, where words are uttered without prior planning and refinement. In a spoken conversation, humans use the speaker's tone of voice and visual cues such as facial expressions or eye movement serve to indicate when someone is being sarcastic, telling a joke, tired, confused, or genuinely thinks something is funny. In written text, these clues are absent, which often leads to misunderstandings. Emoticons help provide a "voice" to the text, leading to a better understanding of the writers intentions.

   There are styles of emoticons for most character sets, including adding diacritics or using writing systems that do not use the Roman alphabet, such as Korean or Cyrillic characters. The most common emoticon styles that we encounter in English writing are typically called the "Western Style" and the "Asian Style". Western emoticons are viewed by tilting the head to the side, e.g. `:-)` indicating smiling or `<3` representing a heart. Asian emoticons, on the other hand, can be viewed right side up, e.g. `(^_~)` can indicate winking or playfulness, while `(o_O)` indicates surprise or confusion. English users tend to leave off the parenthesis representing the side of the face when using

Asian emoticons, for instance writing `^_~` or `o_O` for the previous examples. Although there is a core set of emoticons, new expressions are being constantly invented.

Although not technically considered emoticons, users often explicitly mark pauses with fillers such as “ummm” and “uhh”, a specially marked word indicating they are performing an action (such as **cries**) or simply punctuation to indicate that they are thinking. A message consisting of only an ellipsis often indicates that the writer believes the previous statement was unintelligent. Question marks and exclamation points with no words can indicate a lack of understanding, confusion, or surprise.

When normalizing a message, one may need to distinguish between emoticons and punctuation, or in the case of emoticons such as “XD” (extremely happy: big grin with squinty eyes), between emoticons and words or abbreviations, especially as they are often attached to the previous (or following) word with no space in between. Because new emoticons are constantly being created, it is very difficult to create a comprehensive list for filtering. Emoticons present a difficulty in TTS due to a lack of understanding of how (or whether) they should be pronounced. Much like texting abbreviations, new emoticons are constantly being created and multiple emoticons can serve the same purpose. They may, however, be useful for TTS when used as clues to the type of prosody and intonation that an utterance should have in order to sound natural to a human listener. Often, emoticons also indicate the beginning or ending of a sentence in lieu of typical sentence-final punctuation. This may be a helpful clue for sentence boundary detection in a domain where punctuation marks are often used inconsistently or are altogether absent.

3. **Inconsistent capitalization.** In formal English, proper nouns and sentence initial words are always capitalized. This is not necessarily true of writing in this domain. Capitalization is optional and often used inconsistently even within a single message.

   Although may users capitalize names and sentences inconsistently, it is quite common to use capital letters for emphasis, emotion, just to to LoOk InTeReStInG, or not at all. Cases of abnormal capitalization indicating emotion or emphasis may actually improve a TTS system. A sentence containing entirely capital letters is generally accepted to mean that the writer is yelling, while a few words in all capitalized letters in an otherwise lowercase message means the writer intends to emphasize these words. Therefore, we may see improvements in emotion detection or detection of stress and prominence information that are generally difficult to obtain from text alone. This information can then be used to provide prosodic information to the utterance, making it seem more natural to a human listener.

On the other hand, inconsistent capitalization can greatly increase the difficulty of named-entity recognition (NER), a task that is relatively easy on formal written English where proper nouns are always capitalized. NER in this domain is similar to that of NER on transcripts from an ASR system, which usually lack capitalization information, or on languages such as Arabic or Chinese that have no capital characters. However, other features unique to this domain may prevent us from using techniques from those tasks without modification. The lack of NER may cause inaccuracies when normalizing abbreviations; without the capital letter, it may be difficult to detect whether a given OOV word is an abbreviation that should be expanded or a name, which should not. Lack of capitalization can also have a negative effect on sentence boundary detection (SBD) leading to strange phrasing from the TTS system, especially when combined with a lack of punctuation which is also very common. One common feature for SBD in English is whether the word following the possible sentence boundary is capitalized or not. This task also resembles the equivalent task on ASR transcripts.

1.2 Contributions of the Proposed Work

This thesis describes our advancements toward building an improved TTS system for reading informal text. Our main contributions to the field are summarized as follows:

- The development of an annotated corpus of informal sentences. The corpus will be made publicly available to aid further research.
- Use information gathered from our corpus to develop a state-of-the-art normalization system for this domain, transforming informal text to look more like the formal standard English typically read using TTS
- Development of two methods to create abbreviation models for use during normalization: the first using a statistical classifier to recognize when characters are likely to be deleted from a word, and the second using a character-level machine translation model.
- Exploration of various methods of combining the two abbreviation models.
- Preliminary tests of a heuristic using a dictionary to reduce the false positive rate during normalization.
- Verification that our enhancements improve pronunciation using the Festival TTS system (Taylor et al., 1998) using both quantitative and qualitative evaluations.
- Our work can be generalized to other NLP applications and will improve results for these tasks as well.

CHAPTER 2
LITERATURE REVIEW

This chapter briefly describes research related to processing informal text in general, as well as research related to the specific tasks relevant to this dissertation. Interest in informal text is a relatively new development, so evidence of its increasing importance is provided as well as a description of informal corpora in current use. The final two sections describe work in fields directly related to the dissertation research, though not necessarily directly applied to informal text.

2.1 Informal Text

Informal text has become an increasingly popular research topic in recent years. Many academic conferences are encouraging research on informal domains and noisy text. Both NAACL and ACL have recently had multiple papers and workshops working with text from blogs, micro-blogs and SMS messages. The International Journal on Document Analysis and Recognition has had three special issues on noisy text analytics, in 2007, 2009 an 2011 containing a variety of work, from correcting false online friends to topic mining and understanding the informal language used. TREC has introduced a blog specific track for their workshops, resulting in papers for summarization, topic detection, sentiment analysis, and a variety of other tasks on blog data. Although it would be impossible to give an exhaustive list of examples, we describe a variety of recent work on processing informal text.

One important vein of research on informal text lies in the prevention of spam and trolling messages. Spam messages add no useful content to a blog thread, conversation, or online community and often contain nothing but a hyperlink to the poster's own blog, website, or pornographic webcam. Trolls are users who purposely post derogatory or negative comments in order to incite anger in other users or the original poster. Of the two, spam has the most related work because it began as a problem in e-mail before blogging and other social media became popular. It was shown by Sculley and Wachman (2007) that SVMs can provide efficient spam filtering on blogs in addition to their typical use on email spam. Veloso et al. (2007) uses data mining techniques to provide automatic moderation of spam and trolling comments on blogs that outperforms the traditional SVM and decision trees in terms of speed and accuracy.

A relatively new line of research looks at analyzing the content of micro-blogging websites, such as the popular site, *Twitter* (`http://www.twitter.com`). Researchers are performing search (Teevan et al., 2011), topic modelling of an individual user's stream (Ramage et al., 2010), as well as the more popular research on summarizing"trending topics" on the site as a whole (OConnor et al., 2010; Sharifi et al., 2010a,b; Chakrabarti and Punera, 2011; Liu et al., 2011) and detecting and analyzing users' sentiment regarding a variety of topics (Bollen et al., 2009; Davidov et al., 2010; Barbosa and Feng, 2010; Pak and Paroubek, 2010; Diakopoulos and Shamma, 2010; Tumasjan et al., 2010).

Much work has been done on SMS messages to improve the speed of input through language models (Hasselgren et al., 2003; Klarlund and Riley, 2003; Bento and Gil, 2005; Tanaka-ishii, 2007; Pennell, 2007; Gong, 2008; van den Bosch and Bogers, 2008) and keyboard optimizations (Kober et al., 2001; MacKenzie et al., 2001; How and Kan, 2005; Gong, 2008). Microsoft Research has recently produced many articles about using speech to reply to SMS messages while driving (Ju and Paek, 2009; Wu et al., 2010; Ju and Paek, 2010).

Three small SMS corpora are currently publicly available for current research. The NUS SMS corpus contains approximately 10,000 SMS messages gathered from college students at the National University of Singapore (How and Kan, 2005). The language used is a hybrid of English and many other languages spoken in Singapore commonly known as "Singlish". The work of Choudhury et al. (2007) provides a small parallel corpus of 1,000 messages collected from `http://www.treasuremytext.com`. Finally, Fairon and Paumier (2006) provide a parallel corpus of 30,000 French SMS messages paired with what they call "Standardized French" (as opposed to "Standard", which is controversial).

In addition, recent years have seen the development of corpora based on Twitter status messages, which can be very similar to SMS messages and are used extensively in this dissertation. The Edinburgh Twitter Corpus is a collection of over 97 million Twitter status messages in a standardized format (Petrovic et al., 2010). This corpus is for general use and its size prevents annotating it in its entirety, however it is a valuable resource for many research topics in this domain. In addition, the annotated data used by Han and Baldwin (2011) has been released publicly. Although their dataset is small (549 status messages containing 1184 ill-formed words), is is fully annotated for OOV words and the tokens are left in sentence order so context is available.

2.2 General Abbreviation Modeling, Disambiguation and Expansion

Abbreviation disambiguation and expansion is a common problem when processing text from any domain. Willis et al. (2002) studied abbreviation generation in the hopes of lowering the effort of text entry for people with motor disabilities; the user could enter abbreviated text that would be expanded by a system to be read by his or her conversation partner. They asked a group of young people to abbreviate a text of 500 characters to progressively smaller lengths, assuming they are charged per letter but there is a hefty fee for every error in decoding by another person. Although they do not attempt to expand the abbreviations automatically, they produce a set of rules by which participants produced abbreviations, using both deletion and substitution.

Early work (Pakhomov, 2002; Yu et al., 2003; Gaudan et al., 2005; Pakhomov et al., 2005) showed that medical abbreviations can be modeled and expanded using various machine learning (ML) techniques alongside the principle of "one sense per location" introduced by Yarowsky (1993). These studies used contextual information to help disambiguate medical terms under the assumption that an abbreviation and its correct expansion will be found in similar contexts. This assumption was also used in combination with various ML techniques by HaCohen-Kerner et al. (2008) for Jewish law texts.

Wong et al. (2006) introduced their ISSAC system that works on top of the spell checking program Aspell to perform spelling correction, abbreviation expansion and capitalization restoration simultaneously. Their model gives weight to the original weight given to a suggested correction by Aspell, normalized edit distance, domain significance, number of hits for a word on Google, and appearance of the word/abbreviation pair in an online abbreviation dictionary. In addition, a word is given more weight if it has been seen paired with the current abbreviation earlier in the document. Their re-ranking scheme provides a significant increase in accuracy over Aspell alone.

Yang et al. (2009) work with abbreviations for spoken Chinese rather than for English text messages, but their process is quite similar to ours. They first perform an abbreviation generation task for words and then reverse the mapping in a look-up table. They use conditional random fields as a binary classifier to determine the probability of removing a Chinese character to form an abbreviation. They rerank the resulting abbreviations by using a length prior modeled from their training data and co-occurrence of the original word and generated abbreviation using web search.

Cook and Stevenson (2010) focus on lexical blends, an interesting type of neologism that is common in informal text. While these are not technically abbreviations, the ability to break a lexical blend into its component words may be helpful for pronunciation by a TTS system as well as for understandability in downstream tasks. They generate candidate sets of words for the blend based on substring overlap between the blend and standard English words. They then use a variety of features to generate a ranked order of the candidate sets and choose the highest ranked set at the blend's source words.

2.3 Text Normalization

Abbreviation expansion is just one of many techniques needed for the task of text normalization. Text normalization is an important first step for any text-to-speech (TTS) system. Regardless of the size of the training corpus, there will always be tokens that do not appear and have unknown pronunciations. Text normalization has been widely studied in many formal domains. Sproat et al. (2001) provides a good resource for text normalization and its associated problems.

Table 2.1. Methods for processing unseen tokens during normalization.

Method	Formal Example	Informal Example
as chars	RSVP	“cu” (see you)
as word	NATO	“l8r” (later)
expand	Corp.	“prof” (professor)
combine	WinNT	“neway” (anyway)

In Table 2.1, we show some generally accepted processing methods for unknown words with examples from text messages and more formal text domain. Text message normalization presents many difficulties that are not encountered in other domains. This domain is constantly evolving; new abbreviations appear frequently and inconsistently. One user may abbreviate a single word in multiple ways. Abbreviations containing numbers and symbols are very uncommon in formal text but often seen in text messages. Spell-checking algorithms are mostly ineffective on this data, perhaps because they generally do not account for the characteristics of text messages. They instead focus on single typographic errors using edit distance, such as Kukich (1992), or a combination of this approach and pronunciation modeling, such as Toutanova and Moore (2002).

One line of research views normalization as a noisy channel problem. The work of Choudhury et al. (2007) describes a supervised noisy channel model using HMMs for SMS normalization. Cook and Stevenson (2009) modified this work to create an unsupervised noisy channel approach. They divided abbreviations into eleven categories and created probabilistic models for the most common types. The English word with the highest probability after combining the models is chosen as the standard form of the texting word. Deletion-based abbreviations were addressed in our work (described in detail in Chapter 5) using statistical models (maximum entropy and conditional random fields) combined with an in-domain language model (LM) (Pennell and Liu, 2010, 2011b). Liu et al. (2011) extend the statistical model to be independent of abbreviation type with good results.

Whitelaw et al. (2009) used a noisy channel model based on orthographic edit distance using the web to generate a large set of automatically generated (noisy) pairs to be used for training and for spelling suggestions. Although they use the web for collection, they do not focus on informal text but rather on unintentional spelling mistakes. Acharyya et al. (2009) used an HMM to decode the abbreviations, but implemented an approximate maximum likelihood inference algorithm using clustering over substring vectors rather than the typical Baum-Welsch. Each vector contains a list of subsequences of the term with the normalized weighted count of appearance for each subsequence, where each subsequence has decaying exponential weights depending on the distance between the first and last character of the subsequence in the original string. Unfortunately they do not perform any tests to determine the accuracy of their system; however, anecdotal evidence in the form of their clusters shows reasonable lists of abbreviations for each word. Beaufort et al. (2010) combine a noisy channel model with a rule-based finite-state transducer and got reasonable results on French SMS, but have not tried their method on English text. Han and Baldwin (2011) first determine whether a given OOV word needs to be expanded or is some other type of properly-formed OOV. For those predicted to be ill-formed, a confusion set of possible candidate words is generated based on a combination of lexical and phonetic edit distance and the top word is chosen in context using LM and dependency parse information.

The work of Pini et al. (2010) was originally designed to make text entry easier by allowing users to type free-form abbreviated text that their system would then expand into a formally written message, rather than the typical normalization scenario; however, their methods are necessarily similar. They train a discriminative rather than generative model using an SVM using n-gram features, a language model probability feature, the length of the string and the consonant to vowel ratio. To estimate match probabilities they use logistic regression with similarity features; positive examples of abbreviation pairs are taken from their data, and they generate negative examples by pairing an abbreviation with a randomly chosen English word. Their system allows substitution and deletion of characters in a word as well as deletion of entire function words. Their accuracy on tests using common phrases of up to five words found in an email corpus is quite promising.

Machine translation (MT) techniques trained at the word- or phrase-level are also common. The research of Bangalore et al. (2002) uses consensus translations to bootstrap a translation system for instant messages and chat rooms where text messaging abbreviations are commonly used. Aw et al. (2006) viewed text messaging lingo as if it were another language with its own words and grammar to produce grammatically correct English sentences using MT. Kobus et al. (2008) incorporate a second phase in the translation model that maps characters in the texting abbreviation to the corresponding phonemes, which are viewed as the output of an automatic speech recognition system. They use a non-deterministic phonemic transducer to decode the phonemes into English words. Q. and H. (2009) trained an MT system using three on-line SMS dictionaries for normalizing chat-like messages on Twitter. The technical paper of Raghunathan and Krawczyk (2009) details an explicit study varying language model orders, distortion limit and maximum phrase length allowed by the MT system during decoding. Contractor et al. (2010) also uses an SMT model; however, in an attempt to get around the problem of collecting and annotating a large parallel corpus, they automatically create a noisy list of word-abbreviation pairs for training using some heuristics. As far as we know, our recent work is the first to use an MT system at the character-level for this task (Pennell and Liu, 2011a). The details of that MT system are described in Chapter 6.

CHAPTER 3
CORPUS COLLECTION AND ANNOTATION

NLP research in most domains requires a large, domain-specific, annotated corpus. In particular, the desire to use statistical methods requires a large number of accurate annotations for model training. Unfortunately, such a corpus is not currently available to the research community. Three SMS corpora are available but none met the needs of this study. The use of "Singlish" in the NUS SMS corpus (How and Kan, 2005) and French in the corpus developed by Fairon and Paumier (2006) means that they are not representative of the target English speaking demographic, although it may be interesting to test the methods described in this dissertation on these corpora in the future to see how well the method generalizes. The corpus provided by Choudhury et al. (2007) is very small, with only 1000 messages. Moreover, the test set provided with this corpus is merely an alphabetical listing of pairs of words without contextual information needed for the methods described herein.

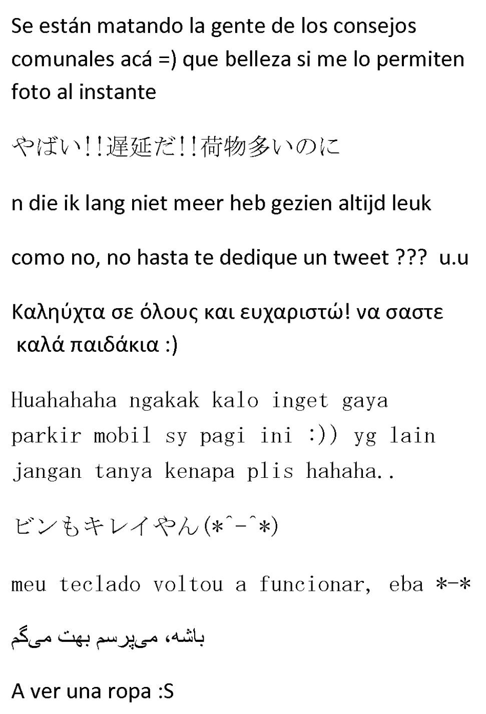
At the time this research began, the Edinburgh Twitter Corpus (Petrovic et al., 2010) and the Twitter corpus developed by Han and Baldwin (2011) were not yet available. Had they existed, neither would have been ideal. These corpora are built from Twitter status messages in general while, in an effort to be more representative of the domain, the Twitter status messages collected for this study were sent to Twitter using an SMS message. While the meta-data is available for the very large Edinburgh corpus, no parallel corpus exists that is specific to messages sent from SMS so the annotation performed for this dissertation would have still been required.

Due to the lack of a suitable corpus, one was built and annotated for this study.

3.1 Collection

The first task in creating a corpus is to gather data. A corpus of actual text messages would contribute best to our end goal. Friends, family, and people from online communities volunteered to anonymously donate their sent messages for the project. Volunteers were asked only to donate their sent messages, since we felt that received messages would require permission from the message writer. Messages were collected using two methods. Some either typed up their messages by hand or used a third party program to download their messages to a text file. These messages were donated via email; to maintain anonymity, the address from which the email originated was not saved. To make the process of donating easier on the participants, a more automatic method of contribution was developed using a Telit GM862-GPS cell phone modem (Telit Wireless Solutions, 2006). The modem was attached to a development board with a RS-232 serial communication bus, which is used to interface with a computer. Commands were issued to the modem over the serial port using a C program; the modem notified the computer when a new text message was received. The computer gathered only the content of the message and saved it to a text file. All other data attached to the message was ignored by the program, including the sender's phone number, making this method of message gathering completely anonymous. Using these two methods combined, approximately 20,000 sentences were collected. Although both costly and slow, these messages give an accurate view of the domain.

To more rapidly enlarge the corpus, public data from the social networking site, Twitter [http://twitter.com] was gathered. Twitter allows users to share "status messages" with their "followers" (friends). These messages may contain up to 140 characters, which is 20 fewer than allowed in a text message. In fact, users may send a text message containing their desired new status to a special number to update their status on the site without having to access the internet. Over one million Twitter updates were collected during two days in early August, 2009. Information about the sender of the update is removed to again maintain anonymity during our collection process.



Se están matando la gente de los consejos comunales acá =) que belleza si me lo permiten foto al instante

やばい!!遅延だ!!荷物多いのに

n die ik lang niet meer heb gezien altijd leuk

como no, no hasta te dedique un tweet ??? u.u

Καληύχτα σε όλους και ευχαριστώ! να σαστε καλά παιδάκια :)

Huahahaha ngakak kalo inget gaya parkir mobil sy pagi ini :)) yg lain jangan tanya kenapa plis hahaha..

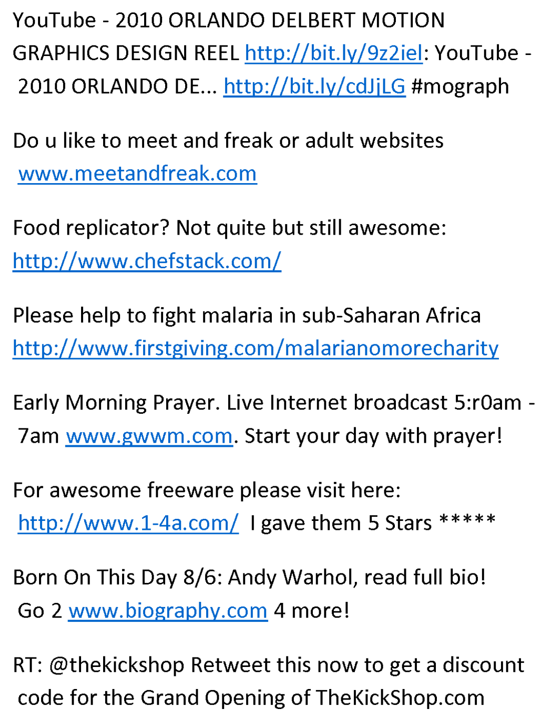
ビンもキレイやん(*ˆ-ˆ*)

meu teclado voltou a funcionar, eba *-*

باشه، می‌پرسم بهت می‌گم

A ver una ropa :S

(a) Non-English messages from *Twitter*.



YouTube - 2010 ORLANDO DELBERT MOTION GRAPHICS DESIGN REEL http://bit.ly/9z2iel: YouTube - 2010 ORLANDO DE... http://bit.ly/cdJjLG #mograph

Do u like to meet and freak or adult websites www.meetandfreak.com

Food replicator? Not quite but still awesome: http://www.chefstack.com/

Please help to fight malaria in sub-Saharan Africa http://www.firstgiving.com/malarianomorecharity

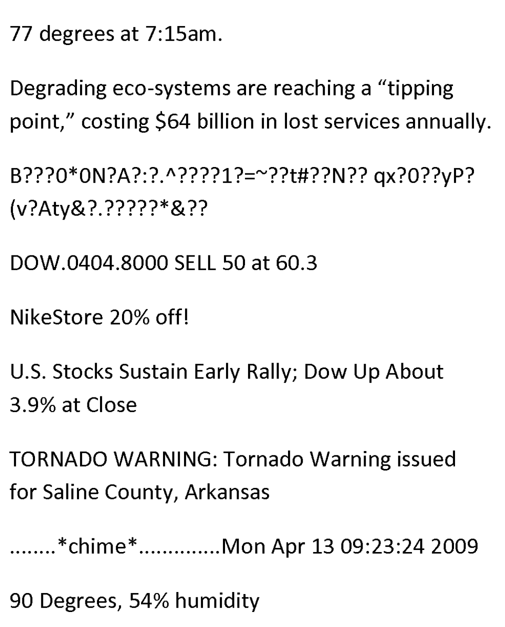
Early Morning Prayer. Live Internet broadcast 5:r0am - 7am www.gwwm.com. Start your day with prayer!

For awesome freeware please visit here: http://www.1-4a.com/ I gave them 5 Stars *****

Born On This Day 8/6: Andy Warhol, read full bio! Go 2 www.biography.com 4 more!

RT: @thekickshop Retweet this now to get a discount code for the Grand Opening of TheKickShop.com

(b) Spam messages detected with URL filter.



77 degrees at 7:15am.

Degrading eco-systems are reaching a “tipping point,” costing $64 billion in lost services annually.

B???0*0N?A?:?.^????1?=~??t#??N?? qx?0??yP? (v?Aty&?.?????*&??

DOW.0404.8000 SELL 50 at 60.3

NikeStore 20% off!

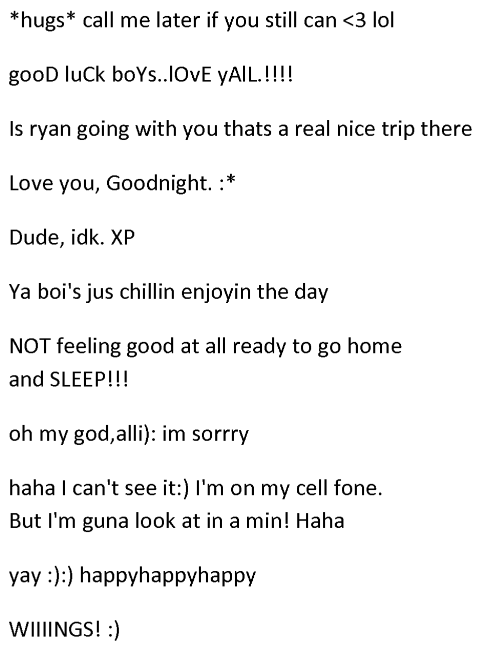
U.S. Stocks Sustain Early Rally; Dow Up About 3.9% at Close

TORNADO WARNING: Tornado Warning issued for Saline County, Arkansas

........*chime*..............Mon Apr 13 09:23:24 2009

90 Degrees, 54% humidity

(c) Other types of unrepresentative messages.



hugs call me later if you still can <3 lol

gooD luCk boYs..lOvE yAlL.!!!!

Is ryan going with you thats a real nice trip there

Love you, Goodnight. :*

Dude, idk. XP

Ya boi's jus chillin enjoyin the day

NOT feeling good at all ready to go home and SLEEP!!!

oh my god,alli): im sorrry

haha I can't see it:) I'm on my cell fone. But I'm guna look at in a min! Haha

yay :):) happyhappyhappy

WIIIINGS! :)

(d) Messages representative of the informal domain.

Figure 3.1. An illustration of the problems faced when adapting *Twitter* status messages to our domain.

The meta-data provided by Twitter enabled collection of only status messages updated by text message, which may more strongly resemble the target domain. Unfortunately, many of these sentences are still not very representative of text messages. A large number of public status messages are contributed by "bots" and contain spam or information like stock quotes, weather reports, or the current time. In addition, since Twitter is utilized by people all around the world, many messages are not in English. Furthermore, many messages contain well-formed text and discuss topics unlikely to be mentioned in SMS messages.

First, a basic filtering of the collected data was performed for some easy-to-find unrepresentative message types. Foreign language messages are shown in Figure 3.1(a). Notice that many contain characters that are not used in standard English, providing a simple method to filter a large portion of foreign messages. However, some messages are more difficult to detect because they only contain the 26 characters (plus digits and punctuation) used in English. In fact, in languages that use the Roman character set plus diacritics (accent markers), users often leave off the diacritic marks when writing informal text. Examples of spam messages containing URLs are shown in Figure 3.1(b). The presence of a URL flags the message for removal. While it would be interesting to attempt using characteristics messages containing URLs to detect spam that does not contain a URL, this is not directly addressed at the present time. The messages shown in 3.1(c), demonstrate other ways to be unrepresentative of the domain without containing a URL or foreign text. These messages are difficult to detect so human input is currently relied upon, although many of these messages are eventually discarded by the filtering process described in Section 3.3.

Finally, messages that are representative of the domain are shown in Figure 3.1(d). Native English speakers can decipher such messages with little difficulty. Conversely, a TTS system lacks the background knowledge needed to guess what these abbreviations might mean or how they might be pronounced. The inconsistency in writing poses problems for NLP tasks.

3.2 Annotation

A set of abbreviations paired with the standard English word(s) from which they were derived is needed to train the normalization methods. In the process of finding the best method both for annotating message and for choosing appropriate messages to annotate, various methods were tried before settling on a message selection algorithm and annotation interface.

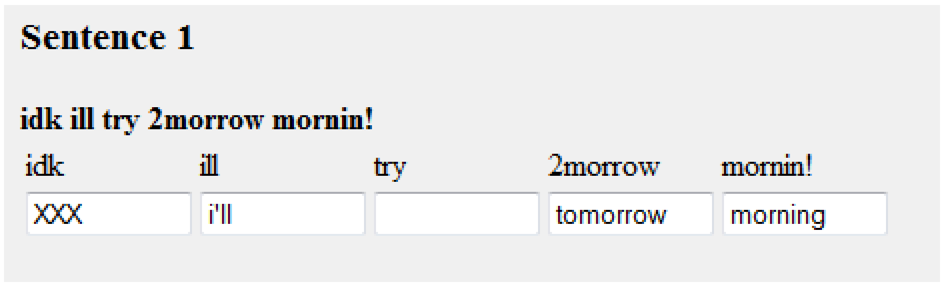
3.2.1 Amazon's "Mechanical Turk"

As a cost and time saving measure, Amazon's "Mechanical Turk"[1] was tried initially. The annotation template submitted to Mechanical Turk is shown in Figure 3.2. Sentences from the SMS portion of the corpus containing at least one OOV word with respect to the CMU dictionary were chosen for annotation. Each annotator was shown the entire sentence since context plays an important role in deciphering abbreviations. Each word in the sentence (using spaces as word boundaries) is then shown with a corresponding text box. Annotators were asked to write the standard English form of all abbreviations in the sentence in their corresponding text boxes. If the annotator recognized that a word was an abbreviation, but unable to guess the standard form, he was instructed to type "XXX". Otherwise, the annotator was asked to leave the text box blank. Three unique workers per message annotated 2,500 total messages in this fashion.

Unfortunately, the results were not very useful. This can partly be attributed to the poor heuristic used to choose messages for annotation. Of the 2,500 sentences submitted, fewer than expected were marked as containing an abbreviations, and many of these were actually names of people or places. We have since implemented an improved method for selecting messages for annotation, which is described in Section 3.3.

It also appears as if participants on the Mechanical Turk do not represent our target demographic. A high percentage of the abbreviations found were marked with "XXX",

[1]This tool allows researchers to upload human intelligence tasks such as annotation, called HITs. Turk users complete the tasks for a small price. This tool can be found at `https://www.mturk.com/mturk/welcome`


Sentence 1

idk ill try 2morrow mornin!

idk	ill	try	2morrow	mornin!
XXX	i'll		tomorrow	morning

Figure 3.2. Our template for annotating abbreviations with their standard English forms using Amazon's Mechanical Turk. The text boxes show responses given by an actual annotator.

indicating that the annotator was unsure of the term's meaning. Not only that, but we could have completed much of this task ourselves in the time taken to check the annotations by hand due to low inter-annotator agreement. For this reason, an annotation GUI was developed and annotators were hired and trained for the final corpus.

Reverse Annotations

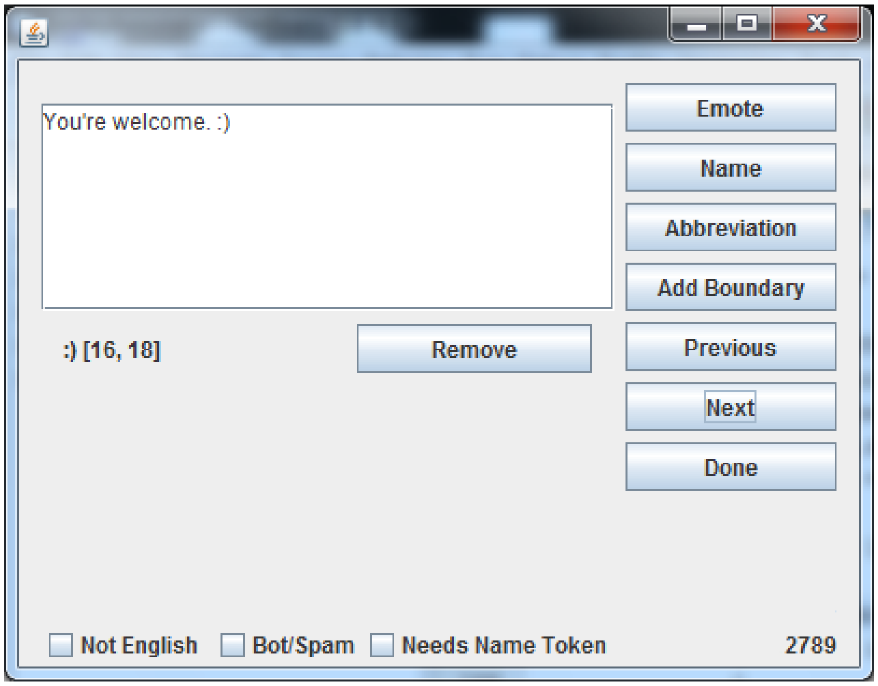
While the GUI development and corpus annotation was in progress, abbreviation to standard English mappings were still needed in order to continue making progress. Reverse annotations of the type shown in Table 3.1 were gathered for this purpose, which are relatively fast and easy to collect. Annotators were asked to perform the reverse translation task, listing as many reasonable abbreviations for a given English word as they could think of. Each annotator could use his or her own definition of "reasonable". To choose English words to be annotated, sentences from our corpus consisting of five or more words were selected as candidates. Those sentences containing words of three or more characters that do not appear in a dictionary were eliminated. The annotator was given an English word from the remaining sentences and context from the containing sentence. Eight annotators translated 1000 total words in this fashion, with each word marked by three distinct annotators. For consistency, each participant was required to complete a minimum of 250 words; most completed more, and one annotated completed all 1,000 words. The inter-annotator agreement is quite low overall. Of the 2917 unique pairs generated by the annotators, 67.08% were listed by only one annotator, 19.78% were listed by exactly two annotators, and only 13.10% were listed by all three. This confirms that a system for this task needs to be robust to new abbreviations in order to be effective. These reverse annotations were only used during development of the abbreviation expansion methods.

Table 3.1. Possible abbreviations for the word “fortunately” given by three annotators.

Word:	fortunately
Context:	fortunately, he’s 10 min away what time?
Ann. 1:	fortuntly, 4tunatly, 4tntly, 42ntly
Ann. 2:	fortnately, forch
Ann. 3:	fortnly, frtly

3.2.2 Java Annotation GUI

Finally, a Java GUI was developed for performing many types of annotations in one pass over the corpus, shown in Figure 3.3. Each message is displayed in the text box in the upper left of the GUI. The annotator can click the check boxes at the bottom or buttons on the right to add annotations; individual uses for each type are explained below. Each annotation is displayed in the area below the message as it is added, and can be removed from the list if the annotator notices a mistake. If a message requires no annotations, the user can simply press the “Next” button to move on to the next message. The “Previous” button can be used to redo the last message if the annotator realizes a mistake was made. The number of completed sentences is displayed in the lower right corner. The annotator can stop at any time by pressing the “Done” button, which writes the output at any time. When the program is run next, it will pick up where they left off. Explanations of each type of annotation we hope to collect are given below.



You're welcome. :)

:) [16, 18]

2789

Figure 3.3. The GUI used for annotating our corpus.

1. **Translation of abbreviations to their standard English forms.** This is the most important type of annotation for the work described in this dissertation. On the GUI, the user highlights an abbreviation with the mouse and clicks the "Abbreviation" button. A pop-up text box appears where the user can write the standard English word or words corresponding to the selected abbreviation. If the user is unsure of the meaning, he can type "XXX".

2. **Location of emoticons in messages.** Currently, this information is only used for tokenization of sentences since users often do not insert spaces between a word and an emoticon. Typical tokenization methods often view these emoticons as punctuation and insert spaces between each character, rather than viewing them as a single entity. Detection of emoticons helps to eliminate this problem. This information should also be useful for sentence boundary detection and improving the naturalness of text-to-speech output, although these are not currently implemented. To mark an emoticon, the user merely highlights it with the mouse and presses the "Emote" button.

3. **Location of named entities in messages.** Detection of named entities is important to distinguish OOVs that are abbreviations needing to be expanded from OOVs that are named entities and should be read using L2P rules. By detecting that an OOV word is a named entity, the accuracy of our system improves since we will not perform expansion. Annotating a named entity is also done by highlighting the entity and pressing the corresponding button. The "Needs Name Token" checkbox is used to fix a potential problem caused by a pre-processing step. On Twitter, users can direct a message at another user by prefacing the message with an @ symbol and the username of their friend, for instance "`@username I saw that too!!`" Because this is not done in text messages, message initial tokens beginning with the @ symbol were removed. This notation is also used when talking about another *Twitter* user; if this happens at the beginning of the message its removal results in an incomplete sentence, such as "`is coming over later`". Named entity annotations are not used in current work, but have many potential uses for future research.

4. **Flagging of unrepresentative messages.** The GUI includes check boxes that an annotator can mark if a foreign or otherwise non-representative message is accidentally selected for annotation. This can provide feedback for tuning the prioritizing process as well as giving cause to eliminate the message from the corpus.

3.3 Choosing Messages for Annotation

After settling on the GUI annotation method, messages must be selected for annotation. With over a million Twitter messages, it is infeasible to have a human look through all of the messages and choose a set by hand. In addition, an annotator's time is wasted if he is presented with the unrepresentative messages described above. Annotating many messages containing no abbreviations, or repeatedly annotate the same, very common abbreviations would also be detrimental to the training process. A scoring system to determine which messages to annotate and in what order was thus devised using the following metrics:

1. **Word Count Index.** A low word count index indicates that a message is close to the mean message length. Messages with fewer than five words are removed from consideration. The index is $|N - E(N)|/\sigma(N)$, where N is the number of words in the message and mean and standard deviation are calculated over the entire corpus.

2. **Perplexity Scores.** Two perplexity scores are utilized calculated against character-level language models. First, the perplexity of the message compared to standard English text is calculated. A lower score indicates that the message is less likely to be in a foreign language or jibberish. The perplexity of the message compared to the rest of the corpus is also calculated. A low score here indicates that the message is more representative of the domain. The sentence is removed completely if in either case the perplexity value is greater than a threshold (1000 in this study).

3. **OOV Count.** This is a simple count of the number of out of vocabulary (OOV) words in the message compared to an English dictionary, which is denoted N_{OOV}. This metric helps guarantee selection of messages containing many OOV words. The sentence is removed completely when $N_{OOV} = 0$.

4. **OOV Percentages.** This metric consists of two scores: the first is N_{OOV}/N; the second is a non-duplicate OOV percentage, where all repeated words are removed and then the percentage is calculated again. If the first score is greater than 0.5 but the second is not, the message is removed from consideration.

5. **OOV Frequency Scores.** For each OOV token (including emoticons) the frequency of the token across the entire corpus is found. This ensures annotation of those abbreviations that are commonly used.

One sorted list is generated for each metric. A final score is generated for each sentence by a weighted average of its position in each list, where more weight is given to the non-duplicate OOV percentage list and less weight is given to the OOV frequency scores. The sentences are ranked in a penultimate list based on this final score. Finally, there is a post processing step consisting of iterating through the list and removing sentences introducing no new OOV words when compared to higher-ranked sentences. Messages were then annotated in the order they appeared in the final score list.

3.4 Annotation Statistics

Five undergraduate students were hired and trained for annotation using the GUI. The students were asked to use all available resources (including web searches or asking friends) to help when they were unsure of a term's meaning. As of the writing of this dissertation, 4661 twitter status messages have been annotated. Table 3.2 shows number of annotations made per student[2]. **Tokens** refers to the total number of annotations made, including duplicates (*i.e.*, the same abbreviation was used in multiple sentences). **Types** refers to the number of annotations when duplicates are removed. **Unique** refers to those pairs that only appeared in this student's annotations. **Unknown** refers to the number of tokens for which this annotator could not find the standard English form.

Table 3.2. Division of annotations in our data.

Annotator	**1**	**2**	**3**	**4**	**5**
Sentences	600	547	892	977	1925
Tokens	866	784	1251	1676	3192
Types	593	570	894	1125	2017
Unique	344	311	542	704	1439
Unknown	51	36	45	68	92

Seventy-four messages were used for inter-annotator agreement. These messages were also annotated by the author as a standard set for comparison. Unfortunately no single hired annotator completed all 74 messages. Agreement is first computed at the boolean level; that is, whether a token was marked as an abbreviation or not regardless of the provided translation. Table 3.3 shows the pairwise agreement between annotators on tokens given to

[2]Originally, the messages were to be divided equally amongst the annotators, but some annotators left the university before the task was completed.

both people, including those that both agreed were not abbreviations. The Fleiss' Kappa of $\kappa = 0.891$ is quite high so the number of annotators can probably be reduced.

Table 3.3. Pairwise boolean agreement (%) between annotators. *A* indicates the author.

Annotator	**1**	**2**	**3**	**4**	**5**
A	98.6	96.8	96.3	97.2	97.7
1		97.0	97.0	96.7	98.1
2			96.4	95.7	97.0
3				97.0	96.4
4					97.1

Non-boolean agreement is also calculated, where in the case that two annotators marked a token as an abbreviation but provided different translations they are considered to be in disagreement. For this task, the maximum number of annotators (per token) who are in agreement is used as a metric. As an example, if one annotator says a token is not an abbreviation at all, two say it is an annotation and translate it as *A* and the remaining three also believe it is an abbreviation but translate it as *B*, the agreement is 3.

Table 3.4. Distribution of maximum annotators in agreement per token.

1	**2**	**3**	**4**	**5**	**6**
0	3	5	15	30	425

Table 3.4 shows the distribution across only those tokens seen by all five annotators and the author, but this is slightly misleading. Of the 478 tokens included in the table, only 82 tokens were marked as an abbreviation by at least one person. Clearly, the 53 with a maximum agreement less than 6 all come from this set, showing this task is more difficult.

Table 3.5. Abbreviation types. Subcategories are italicized below their parent category.

Category	Definition	Example
Deletion	Deletions only.	tmro (tomorrow)
Clipping	*Entire syllable(s) deleted*	*prof (professor)*
Location	*Silent 'e', Initial 'h' or -ing 'g'*	*talkin (talking)*
General	*Single word, not covered above*	*abt (about)*
Initialization	*First letter of each word*	*jk (just kidding)*
Substitution	Replacing characters with others	2nite (tonight)
As Sound	*Reading the token as a word*	*l8r (later)*[3]
As Character	*Pronouncing character's names*	*ne(any)*[3]
Symbolic	*Symbol(s) that look similar*	*$top (stop)*
Insertion	Word length is increased	lightining (lightning)
Repetition	*Letters repeated for emphasis*	*yesssss (yes)*
Stylistic	*Intentional changes*	*confrused (confused)*
Error	*Generally caused by a typo*	*jusyt (just)*
Swap	Correct letters in wrong positions	freind (friend)
Combination	Use of two or more of the above.	2sdy (Tuesday)

[1]As opposed to "ell-ate-arr" and "neh"

3.5 Types of Abbreviations

Abbreviations can be categorized by formation method. Cook and Stevenson (2009) proposed eleven categories that are somewhat subjective due to overlap. Three broad categories are proposed for use in this work (two with subcategories) loosely based on edit distance. The categories are listed with examples in Table 3.5, and are referred to throughout the dissertation. Abbreviations can mostly be categorized by looking at the characters alone using these divisions, though some ambiguity stems from silent 'e' removal (does "mayb" remove a silent 'e', or replace "be" with 'b'?). Insertions and swaps are often the result of typing errors. The count of each type of abbreviation found in the annotated data is shown in Table 3.6. The vast majority of insertions are of the repetition type. For this reason, they are categorized and treated separately in this work. Insertions of the phonetic type (such as *dawg* for *dog*) are considered substitutions for the purposes of categorization.

Table 3.6. Division of abbreviation types in the annotated data.

	Deletion	**Substitution**	**Repetition**	**Swap**	**Insertion**	**Combin.**
Tokens	2546	828	558	106	59	197
Types	1406	620	510	106	55	140

3.5.1 Analysis

Table 3.7 shows a comparison of statistics gathered from collected SMS messages and the annotated Twitter data. Note that there are approximately four times as many SMS messages as there are annotated Twitter messages. The number of words and the number of out-of-vocabulary (OOV) words compared to the CMU Lexicon[4], which contains over 125k words were counted first. Although the number of words is far higher for the SMS data (as expected, since there are many more messages) the number of unique words is lower. This can be attributed to the fact that we had a small number of contributers, and a single person tends to use a somewhat stable set of words. The Twitter data could have theoretically been taken from as many users as there are messages. Twitter users are also more wordy than the SMS contributers; SMS users have an average of just over 6 words per message, while Twitter users have just over 14 words per message. Many of the SMS messages appear to be replies to another SMS user, such as “no thanks”, “that’s fine” or “just got home”. Twitter messages, on the other hand, are often written to no one in particular and usually contain explanatory sentences so that followers will understand the context. The OOV rate of the SMS data is lower than expected, which we believe is because most of our contributers are in graduate school and probably do not represent the stereotypical SMS user. The SMS OOV rate is quite high compared to formal text and considering the size of the reference dictionary but is very low compared to the Twitter data, which has a high OOV rate due to the large number of proper nouns in the data.

[4]Available from `http://www.speech.cs.cmu.edu/cgi-bin/cmudict`

Examination of the OOV words showed that the OOV words generally fell into five categories: intentional misspellings, abbreviations and slang (*e.g.,* "wen" for "when", "brb" for "be right back", "friended"); proper nouns (*e.g.,* "leanne"); onamotapoea (*e.g.,* "aaaaargh"); alphanumeric words (*e.g.,* "2moro", "gr8", "10pm"); and emotes (*e.g.,* ":-)", "<3"). Regular expressions were used to estimate the number of words falling into the alphanumeric and emote categories for the SMS data. For the twitter data, a regular expression was used to find the alphanumeric words, but the emote numbers are taken directly from the annotated data.

Table 3.7. Analysis of tokens in the corpus.

	Category	Total (i.e., tokens)	Unique (i.e., types)
SMS	words	121,572	8,185
	OOV words	3,969	1,498 (18.3% of word types)
	alphanumeric (regex)	150	95 (6.34% of OOV word types)
	emotes (regex)	1,015	45 (3% of OOV word types)
Twitter	words	65,315	12,243
	OOV words	16,666	7,553 (61.6% of word types)
	annotated abbrs.	6300	3096 (25.28% of word types)
	alphanumeric (regex)	597	495 (6.5% of OOV word types)
	emotes	1,170	199 (2.6% of OOV word types)

3.6 Chapter Summary

Supervised methods in natural language processing require the use of a large annotated corpus. To this end, a collection of SMS-like status messages from Twitter was gathered, as well as a set of actual SMS messages from graduate student volunteers. Statistics gathered on both data sets show that they are similar in terms of the types and amounts of non-standard words present. Five student annotators used a Java GUI designed specifically for this project to mark the location and meaning of abbreviations found in the Twitter data, with excellent agreement. This annotated data will be used throughout the following chapters.

CHAPTER 4
NORMALIZATION MODEL AND SETUP

This chapter describes the setup for the experiments performed in Chapters 5, 6 and 7. First, the noisy channel model used in this work for normalization of messages is explained and broken down into its constituent parts in Section 4.1. The four experiments performed for each model presented in the following chapters are then described in Section 4.2. Finally, some baseline experiments and results with which system performance can be compared are described in Section 4.3.

4.1 The Noisy Channel Model for Normalization

This work uses the noisy channel model for normalization, similar to automatic speech recognition (ASR) and statistical machine translation (SMT) problems. For a given text message sentence, $A = a_1a_2...a_n$, the problem of determining the sentence of standard English words, $W = w_1w_2...w_n$, can be formally described as:

$$\begin{aligned}
\widehat{W} &= \arg\max P(W|A) \\
&= \arg\max P(W)P(A|W) \\
&\approx \arg\max \prod P(w_i|w_{i-n+1}...w_{i-1}) \times \prod P(a_i|w_i) \\
&= \arg\max(\sum \log P(w_i|w_{i-n+1}...w_{i-1}) + \sum \log P(a_i|w_i)) \qquad (4.1)
\end{aligned}$$

where the approximation is based on the assumption that each abbreviation depends only on the corresponding word (note that we are not considering one-to-many mappings in this study), and a word is dependent on its previous $(n-1)$ words. In other words, the probability $P(w_i|w_{i-n+1}...w_{i-1})$ is represented by a traditional n-gram language model (LM), which we briefly describe in Section 4.1.1. The abbreviation score in Equation 4.1, $P(a_i|w_i)$, typically corresponds to the acoustic model during ASR or the translation model during SMT. In this work, these models are replaced by an abbreviation model (AM) representing the likelihood that an abbreviation a_i is derived from word w_i. There could be many ways to generate the AM; two possible methods are described in the following chapters.

Equation 4.1 assumes that the AM and LM should be weighted equally, but in actuality one model may prove to be more helpful than the other. For this reason, the terms from Equation 4.1 may be given weights, yielding the final equation

$$\widehat{W} = \arg\max(\alpha \sum \log P(w_i|w_{i-n+1}...w_{i-1}) + \beta \sum \log P(a_i|w_i)) \tag{4.2}$$

where α and β are determined empirically. The final system is thus a two-stage process: In the first stage the $P(a|w)$ scores are generated and in the second stage they are combined with LM scores using Equation 4.2.

4.1.1 Language Modeling

Language modeling, and n-gram modeling in particular, is an important part of natural language processing. Modeling the informal domain will help achieve better disambiguation results when normalizing abbreviations in the text.

The AM provides possible standard English words with associated scores corresponding to a given abbreviation. Assuming that the context is entirely made of words in our LM, the probability that each standard word in the list appears in that context is given by the model. Of course, there is no such guarantee for this task. In fact, it is quite likely that the surrounding words are also abbreviations that do not appear in the LM. A word-level LM is used in Equation 4.2 to help disambiguate multiple word hypotheses for an abbreviation when translating an entire text message.

The language model (LM) used during decoding is a trigram language model generated using the SRILM toolkit (Stolcke, 2002) with Kneser-Ney discounting. Those messages from the Edinburgh Twitter corpus (Petrovic et al., 2010) containing no out-of-vocabulary (OOV) words compared to a dictionary are used to train the model. SRILM is constrained to use the appropriate order LM for the amount of context given during testing.

4.1.2 Abbreviation Modeling

There are many ways to create an AM for the $P(a_i|w_i)$ scores. Two methods are described in this dissertation: a model specific to those abbreviations formed by only deletions (Chapter 5) and a more general model using a character-level MT system (Chapter 6). Some approaches for combining these two models are discussed and evaluated in Chapter 7.

4.2 Experimental Setup

Cross validation was used for the experiments on each AM presented in the dissertation. The data from four annotators is used as training data, while the data from the fifth annotator is divided in half for development and testing. Two tests are performed for each fold; initially, the first half is used for development and testing is done on the second half, then the development and test portions were swapped. The results are averaged over all ten tests.

Throughout the experiments there are two sets of abbreviations on which experiments are performed. The first is made up of all single-word abbreviations (SW) in the annotated data. This data does not include those annotations where a single token was mapped to multiple words or when multiple tokens (or parts of multiple tokens) are mapped to one or more words. As previously mentioned, due to pre-processing it also does not include those tokens annotated as sound effects or those tokens where the annotator was unable to guess a translation even though he or she recognized that it was an abbreviation. There are 4409 abbreviations in this test set. The second test set is made up of only those abbreviations that can be formed from their annotated standard form by deletions only (DEL), resulting in 2546 abbreviations for testing. This is necessary because the abbreviation model described in Chapter 5 is designed to address this and only this type of abbreviation. For fair comparison, all other AMs and the baseline systems are also tested on this deletion-only set. When performing tests, abbreviations not in the current test set (SW or DEL) are replace with their standard forms. This yields original word error rates (WERs) of 9.49% for DEL and 11.6% for SW. Note that the original error rate of the data set is even higher. These WERs are much lower than the OOV rate shown in Table 3.7 because of the high occurrence of proper nouns in the Twitter data set, which are OOV compared to the dictionary used but are already considered to be in standard form.

For each abbreviation model, the three tests listed below are performed.

1. **Abbreviation Model Alone.** The first question is how well the scores generated by each abbreviation model perform on their own, without any information from a language model. Without a language model, a system cannot make use of context, so we submit only an abbreviation for each test case.

2. **Incorporation of an LM for Decoding.** Next, the language model information is incorporated using Equation 4.2. Hypotheses are generated for an abbreviation using the AM under test and then decoded using the language model to find the final score for each hypothesis. The hypotheses are ranked by score and the highest scoring word is suggested by the system as the translation. The language model scores are automatically retrieved using the SRILM toolkit (Stolcke, 2002).

3. **AM for Pruning Only.** Finally, it may be interesting to see what contribution each abbreviation model has toward re-ranking the candidates to find the best choice of translation. In this setup, the AM under test is used to generate the possible hypotheses for translation; however, only the language model is used for decoding rather than combining the LM score with the AM score (*e.g.*, setting $\beta = 0$ in Equation 4.2). In this way, the AM is only used to prune the number of words the LM must consider but does not contribute to the score used to rank these hypotheses. Once again, the language model scores are automatically retrieved using SRILM.

Top-N accuracy is calculated for $N = 1$, 3 and 10 as well as the total number of correct standard forms found regardless of position. Each of the three tests is executed using no context (*unigram*). Tests 2 and 3 are also run with one or two words of context on either side (*bigram* and *trigram*, respectively) in order to see how the order of the LM affects results. These tests are first run using only trigram context on the development set to optimize top-1 accuracy. During optimization, α is set to 1 and β is varied from 0.01 to 100. In these context tests, abbreviated context words are replaced with their annotated standard form to yield the best chance of decoding.

Each test is also performed at the message level, where the entire message is decoded using the model under test and the SRILM lattice-tool. It is common for abbreviations to appear in the context of other abbreviations, which poses a more difficult decoding problem when context is used. Thus, in the message level tests the only words replaced with their standard forms are those not in the current test set. Sentences containing no abbreviations for the current test set were removed. For some setups, thirteen sentences caused the SRILM lattice-tool to run out of memory, even when run on a machine with 8G of memory and using options designed to help with memory usage. These messages were therefore discarded for all message level tests across all setups described in this dissertation. This results in 2525 messages containing 4387 abbreviations for the SW test set and 1687 messages containing 2511 abbreviations for DEL.

A word lattice is generated using the AM under test in order to decode the message. An except from an example lattice is shown in Figure 4.2. At the top, the number of nodes N and transitions (links) L are specified. The eight nodes for the example sentence are defined by the lines “I=0” through “I=7”. The words label the links between nodes. The first link shown in the figure, “J=6”, starts at node 2 (“S=2”) and ends at node 3 (“E=3”). There are 14 possible words that can link nodes 2 and 3 shown in the figure. Each W represents a candidate word, and the associated score a represents $P(a_i|w_i)$. When the system under test is unable to generate any candidate hypotheses, as is the case for the tokens “fraps” and “:)” in the example sentence, the original token is used as the candidate with the score -0.1. If the original token is not one of the hypotheses from the system under test, it is added

```
N=8     L=41
I=0
I=1
I=2
I=3
I=4
I=5
I=6
I=7
J=0     S=0     E=1     W=<s>    a=-0.1
J=1     S=1     E=2     W=greening        a=-3.37862
J=2     S=1     E=2     W=greeno          a=-5.19076
J=3     S=1     E=2     W=gren   a=-5.46853
J=4     S=1     E=2     W=green  a=-0.952995
J=5     S=1     E=2     W=greene          a=-4.37003
J=6     S=2     E=3     W=eat    a=-4.66563
J=7     S=2     E=3     W=ate    a=-5.6587
J=8     S=2     E=3     W=teat   a=-6.58284
J=9     S=2     E=3     W=tate   a=-7.1831
J=10    S=2     E=3     W=te     a=-5.39761
J=11    S=2     E=3     W=tee    a=-6.49165
J=12    S=2     E=3     W=tae    a=-6.5319
J=13    S=2     E=3     W=teer   a=-7.38816
J=14    S=2     E=3     W=heat   a=-7.16311
J=15    S=2     E=3     W=teal   a=-7.23479
J=16    S=2     E=3     W=tear   a=-6.14705
J=17    S=2     E=3     W=ta     a=-6.55731
J=18    S=2     E=3     W=ter    a=-6.32367
J=19    S=2     E=3     W=tea    a=-2.77546
J=20    S=3     E=4     W=fraps  a=-0.1
...
J=34    S=5     E=6     W=god    a=-7.16848
J=35    S=5     E=6     W=guide  a=-7.21387
J=36    S=5     E=6     W=guard  a=-5.92715
J=37    S=5     E=6     W=good   a=-7.77716
J=38    S=5     E=6     W=gude   a=-7.32483
J=39    S=5     E=6     W=guuud   a=-7.77716
J=40    S=6     E=7     W=:)     a=-0.1
```

Figure 4.1. An excerpt from the lattice for the input sentence “green tea fraps r guuud :)”

as the final link between the two nodes with a score equal to the minimum score across all other hypotheses for that token (see link “J=39” in the figure).

The SRILM lattice-tool is used to generate the 20 best sentences. The lattice-tool is typically used to combine a language model with an acoustic model; the abbreviation model replaces the acoustic model for this test. The decoding is easier than the typical case because there is a strict one-to-one alignment between words and abbreviations for all of the tests in this dissertation. The LM is constrained to use order 1, 2 or 3 to test the impact of LM order. For message level tests, the following metrics are used:

1. **Abbreviation top-N accuracy.** This metric is used to see what degradation of performance occurs due to the added difficulty of decoding abbreviations when there are other abbreviated words nearby. An abbreviation is correct in top-N if the word is correctly translated in at least one of the top N sentences. Note that when using this metric a score of 100% in top-N would not guarantee that any single top-N sentence is 100% correct. One abbreviation may be correct in sentence $i < N$ but incorrect in sentence $j < N$, where the opposite is true of a second abbreviation. In addition, it is possible that a system may falsely “correct” a word that was already correct in the original sentence, changing it to another English word.

2. **Sentence-level top-N accuracy.** A system is marked as correct in top-N at the sentence-level if one of the top-N decoded sentences exactly matches the reference sentence. The system therefore corrected all abbreviations in the original sentence but did not mistakenly change any words that were already in their proper standard form.

3. **Top-1 false positive (FP) rate.** This metric is only calculated for the top-1 hypothesis for each message since it is difficult to define the metric over multiple sentences. It describes the percentage of words already in standard form in the original message that are changed to an incorrect word by the system under test.

4. **Top-1 word error rate (WER).** Similar to the last metric, this is also calculated for only the top-1 hypothesis. It describes the final WER of the top-1 hypothesis for

the system under test. This includes both correctly normalized abbreviations as well as the FP changes.

4.3 Baseline Experiments

Before the methods presented in this dissertation can be tested, baselines must be established with which a comparison can be made. Two baselines are provided: the first uses a language model (LM) for decoding with no information about the abbreviations themselves, while the second uses a state-of-the-art spelling checker, Jazzy (Idzelis, 2005).

4.3.1 Language Model Baseline

A natural question is how well the LM performs on its own without the use of an AM. It is impractical conduct experiments with no context for this baseline because the LM is unable to differentiate between unigrams without context and will always choose the same word (the one with the highest unigram probability).

Thus the LM baseline is tested using one or two words of context on either side of the abbreviation. For a test case w_1Aw_2 ($w_1w_2Aw_3w_4$) the LM is used to extract all bigrams (trigrams) that begin with the left context word(s), as well as their scores. The second (third) word in the bigram (trigram) is considered a candidate *A'* for the translation of A. Then the LM the score for the bigram (trigram) beginning with *A'* followed by the right context is extracted. For trigram testing, the score for the trigram $w_2A'w_3$ is also used. As the scores produces by SRILM are log-probabilities, the scores are added to find a final score for the word *A'*.

In the exceptional case where no bigram (trigram) beginning with the left context exists, the system backs off to the unigram (bigram) model and considers all unigrams as possible candidates for A. This model also backs off when there are no bigrams (trigrams) beginning with a candidate *A'*, or in the case of trigram testing, when the trigram $w_2A'w_3$ does not appear in the model.

Finally the candidates are ordered by their scores and the highest ranking candidate *A'* is chosen as the translation of the abbreviation A.

Table 4.1. Top-N accuracy (%) using only a language model for decoding.

	Top-1	**Top-3**	**Top-10**	**Found**
Bigrams (SW)	13.95	24.86	38.98	91.69
Trigrams (SW)	18.32	28.51	39.22	75.54
Bigrams (DEL)	13.15	24.82	39.31	90.7
Trigrams (DEL)	17.32	26.82	37.50	74.51

The results of these tests are shown in Table 4.1. The top-N accuracy is calculated for values $N = 1$, 3 and 10, where a system is considered correct in top-N if the correct translation appears in the first N hypotheses given by that system. The number of correct answers eventually **found** by the system regardless of how far down the list it appears is shown in the final column of the table.

Using only the LM performs very poorly, as expected. The results are similar for the two test sets, which is to be expected since the LM has no knowledge of the abbreviation form, or even what abbreviation originally appeared in the context. Note that while the trigram context outperforms the bigram context in general, the bigram context has a higher **found** percentage. This is because the trigram context eliminates the correct word C from consideration when the trigram W_1W_2C does not appear in the language model, which happens somewhat frequently.

4.3.2 Jazzy Spell Checker

Similar to Liu et al. (2011), the state-of-the art spell checking algorithm, Jazzy (Idzelis, 2005) is also used as a baseline system. Jazzy is based on the Aspell algorithm and integrates a phonetic matching algorithm (DoubleMetaphone) and Ispells near miss strategy that enables the interchanging of two adjacent letters, as well as the changing/deleting/adding of letters.

Rather than using the small dictionary included with the Jazzy source, an Aspell dictionary[1] is incorporated. Initial tests using both dictionaries indicated better performance using the larger Aspell dictionary. An attempt was made to use the Aspell phonetic dictionary[1]as well, however it degraded performance significantly compared to Jazzy's default setup. Jazzy's configuration file was modified so that corrections for words containing digits would be suggested; by default Jazzy ignores these tokens, which are common in this domain.

Jazzy's predictions do not change based on context, so only unigram-based tests are performed. When Jazzy believes a word to be misspelled, it returns a ranked list of words. This is treated as the N-best list and top-N accuracy is calculated. If Jazzy does not provide any suggestion we mark it as correct if the token was already in standard form, and incorrect otherwise. The results for Jazzy on our two data sets are shown in Table 4.2. The results on this data are lower than those obtained by Liu et al. (2011) (on their own test set) by a fair margin.

Table 4.2. Top-N accuracy (%) using the Jazzy spell checker.

	Top-1	**Top-3**	**Top-10**	**Top-20**	**Found**
SW	37.91	39.47	44.25	44.53	44.60
DEL	34.68	36.72	43.40	43.51	43.60

4.4 Chapter Summary

Throughout this dissertation, the noisy channel model is used for normalizing the abbreviations found in informal text. This model combines the use of a language model (LM)–here derived from a portion of the Edinburgh Twitter corpus–with an abbreviation model (AM) that gives likelihoods for representing an English word with various abbreviations. The abbreviation models used will be described in the following chapters. For each abbreviation

[1]en_US dictionary and en_phonet.dat from Ubuntu, package version 6.0-0-6ubuntu1

model, evaluations are performed to determine the effect of having varying amounts of context available for decoding, the effect of increasing the order of the language model used, and the usefulness of the AM when it is used with an LM, without an LM and when the AM is used only to generate hypotheses but not in calculating the final hypothesis score. Two baselines are also presented: the first uses only a language model to choose a likely word for the context in which the abbreviation appears, and the second uses a state-of-the-art spell checking algorithm.

CHAPTER 5
STATISTICAL DELETION MODELING[1]

The first abbreviation model created focused only on deletion-based abbreviations of single words, which the annotated data shows to be the most common abbreviation type. This is a reasonable starting point because the end goal of this work is a TTS system. The *As Sound* class is the most frequent substitution method, leading to reasonable pronunciations using letter to sound rules. Although a deletion-based system cannot guess the standard form, the end user can understand the speech. This chapter describes an approach for creating an abbreviation model for the *Deletion* types.

5.1 Model Description

Concentrating only on the *Deletion* class allows the task to be viewed as a binary classification problem. For each word/abbreviation (W/A) pair, tagging is performed at the character level as shown in Figure 5.1. A tag of 'N' means that the character should be removed to form the abbreviation, while a tag of 'Y' means that the character should remain in the word. A comparison of two classifiers is presented. The first is maximum entropy modeling (ME) using the LeZhang implementation of the algorithm[2]. The second is conditional random fields (CRFs) using the open source software CRF++[3].

[1]©2010,2011 IEEE. Portions reprinted, with permission, from:
Pennell, Deana L. and Yang Liu, Normalization of Text Messages for Text-to-Speech, in *Proceedings of ICASSP*, April 2010.
Pennell, Deana L. and Yang Liu, Toward Text Message Normalization: Modeling Abbreviation Generation, in *Proceedings of ICASSP*, May 2011.

[2]http://homepages.inf.ed.ac.uk/lzhang10/maxent_toolkit.html

[3]http://crfpp.sourceforge.net/

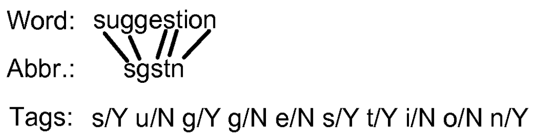


Figure 5.1. Character-level tags for one abbreviation of ‘suggestion’.

5.1.1 Features

The following list of features was compiled in an attempt to capture the abbreviation methods discussed in Section 3.5. The reasons for choosing these features follow the feature list.

1. Contextual Features
   - (a) The character itself, c_i.
   - (b,c) The two previous characters, c_{i-1} and c_{i-2}.
   - (d,e) The following two characters, c_{i+1} and c_{i+2}.
   - (f,g) The two bigrams containing current character, $c_{i-1}c_i$ and c_ic_{i+1}.
   - (h,i,j) The trigram containing the two previous and the current character, $c_{i-2}c_{i-1}c_i$, $c_{i-1}c_ic_{i+1}$ and $c_ic_{i+1}c_{i+2}$.
2. Function Features
   - (a) Whether it is identical to the previous character.
   - (b) Whether the character serves as a vowel.
   - (c) Concatenation of features 2a and 2b.
   - (d) Concatenation of features 1a and 2b.
3. Syllabic Features
   - (a) The characters making up the current syllable.
   - (b) Whether the character is in the first syllable of the word.
   - (c) Whether the character is in the last syllable of the word.

(d) Whether the character is in neither the first nor last syllable.

(e) Concatenation of features 3b, 3c and 3d.

(f) The character's position in its syllable.

(g) Concatenation of features 3a and 3f.

(h,i) Two features concatenating 3f with both 3b and 3c.

(j,k,l,m,n) Features resulting from concatenation of 1a with each of 3a, 3b, 3c, 3d and 3f.

4. CRFs enable us to use the classification of the previous character as a final feature.

The contextual features were designed to help with the *Location* class of abbreviations as well as some *General* deletions. <s> and </s> represent the beginning and ending of a word.

The function features were also intended to help with the *General* deletion abbreviations. Feature 2d was included to help locate the silent 'e' character, since no pronunciation features are included. Any syllable-final 'e' character is considered to be silent when preceded by a consonant. A silent 'e' determined in this fashion is marked as a consonant in Feature 2b.

The free online dictionary found at `http://www.dictionary.com` is utilized for finding syllable boundaries. This website does not list words containing prefixes or suffixes as separate entries from the base word, which causes problems in some cases. To address this issue, an automatic syllabification method (Bartlett et al., 2008) is planned for future work. Once the syllable information is known, features 3a–3d are easy to extract. For single syllable words, features 3b and 3c fire, while 3d does not. These features were intended to help generate the *Clipping* subclass of abbreviations.

For feature 3f, a character's position (**B**eginning, **M**iddle or **E**nd) within a syllable is determined as follows. When a consonant falls to the left of the sonorant vowel(s) in the syllable, it is labeled **B**, while those on the right are labeled **E**. Vowels with consonants on both sides are labeled **M**. Those with no consonants to the left are assigned **B** and those with none on the right are assigned **E**. If a syllable consists of only vowels, they are labeled

B. Again, a silent 'e' is considered to be a consonant, since it does not function as a vowel in the syllable. An example showing positions of characters in the word "symmetry" is shown in Figure 5.2.

Yang et al. (2009) found that for their Chinese abbreviation CRF model, the best features were the current character, the word in which the character appeared, the character's position within the word, and a combination of the final two features. This is analogous to using Feature 3a (syllable) and Feature 3f (position in syllable). Thus, Features 3g, 3h and 3i are included. Feature set 3j-n learns whether certain letters are more likely to be removed when in certain positions in a word or syllable.

5.1.2 Generating Word Candidates

To ensure that the abbreviation model is robust to the numerous and inconsistent variations in abbreviating, multiple abbreviations for each word must be generated. The posterior probability of the tag for each character c generated by the classifier can be used to compute an abbreviation score. For each abbreviation a of an N-character word w, its abbreviation score is:

$$S_a = \prod_{i=1}^{N} sc(c_i), \tag{5.1}$$

where

$$sc(c_i) = \begin{cases} p(c_i) & c_i \in a \\ 1 - p(c_i) & \text{otherwise} \end{cases} \tag{5.2}$$

c_i is the i^{th} character of word w, and $p(c_i)$ is the posterior probability of the 'Y' class.

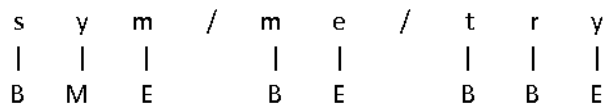


Figure 5.2. An example of syllable positions assigned to characters. The beginning, middle and end positions of a syllable are noted by 'B', 'M' and 'E', respectively.

5.1.3 Reranking and Decoding

The normalization performance is bounded by the number of W/A pairs from the test set that appear in the look-up table (described below). For this reason, the list of abbreviations generated for a word is re-ranked to maximize coverage. Based on the finding in Yang et al. (2009) that there is a strong correlation between the length of the formal text and the length of the abbreviation for Chinese organizations, the candidates are rescored by a length model. The original abbreviation score is combined linearly with the length based information, i.e., the probability of using an M-character abbreviation for an N-character word, $P(M|N)$, which is gathered from the training set. The new score S'_a is thus:

$$S'_a \approx \alpha \log P(M|N) + \beta \log S_a, \tag{5.3}$$

where α and β determine the weighting for each model, while S_a refers to Equation 5.1. Note that S'_a is the representation of $P(a_i|w_i)$ from Equation 4.2 for this model. The top N most probable abbreviations for each word in our dictionary are stored in a look-up table.

In decoding, it is desirable to find all possible w_i for a given a_i as quickly as possible. Therefore the look-up table is reversed from the generation process – for an abbreviation, the look-up table contains a list of possible words and their corresponding scores (S'_a from Equation 5.3).

5.2 System Setup

The look-up table is created by generating abbreviations for all words in the CMU lexicon that have syllables available through `dictionary.com`. For an n-character word, initially $2n$ abbreviations are generated using a particular setup. These $2n$ abbreviations were then re-ranked using the length model and then pruned to use the top $n-1$ abbreviations for building the look-up table. The values $2n$ and $n-1$ were found empirically to be a good balance between the size of the look-up table and accuracy. These generated pairs are then reversed and stored in the look-up table used to find the hypotheses and scores during testing. The results shown here were derived by using all of the features during training and weighting the length model and abbreviation scores equally, though we do attempt to optimize these values there was no significant gain from doing so (see Section 5.4). Finally, the development set is used to tune the weights of the language model and the resultant score model.

5.3 Experimental Results

The results from the experiments described in Section 4.2 are shown below. A full comparison of these results with those presented in Chapters 4, 6 and 7 can be found in the Appendix.

5.3.1 Model Performance

The results for these deletion-based methods (shown in Table 5.1) are somewhat disappointing on their own, with only just over half of the deletion-abbreviations appearing in the look-up table. CRFs outperform ME by a slight margin at all points in the top-N list, probably due to its use of the classification given to the previous character. For the CRF classifier, 72.8% of the correct standard forms are found in the top-1 even without context, which is promising. If this method is to be effective in future work, it will be important to find a way to raise the upper bound by improving the features and the re-ranking method.

Table 5.1. Accuracy (%) of deletion-based systems on abbreviations only without LM information.

	ME (DEL)	**CRF (DEL)**
Top-1	36.76	37.27
Top-3	41.39	43.75
Top-10	46.11	48.86
Found	49.13	51.17

5.3.2 Full System Performance

The results of the full system on the context tests using each of the deletion-based AMs are shown in Table 5.2. During development, it was found that setting β on the low end (while holding α at 1) produces higher results. The final values chosen for β are 0.1 and 0.05 for CRF and ME respectively. CRFs continue to outperform ME by a small margin. The addition of language model information greatly improves the performance of these models during the context tests over the using only the abbreviation model in Section 5.3.1. Even without context, the addition of the unigram LM score is able to increase the CRF performance to 89% (over 94% for ME) of the upper bound in the top-1. In general, a slight performance gain is obtained when the LM order is increased.

Table 5.2. Accuracy (%) of deletion-based systems in n-gram context tests when using both AM and LM scores.

		ME (DEL)	**CRF (DEL)**
Top-1	**Unigram**	43.00	45.48
	Bigram	45.12	46.81
	Trigram	45.87	47.44
Top-3	**Unigram**	48.03	49.84
	Bigram	48.35	50.39
	Trigram	48.35	50.39
Top-10	**Unigram**	49.13	51.09
	Bigram	49.13	51.09
	Trigram	49.13	51.09
Found		49.13	51.17

Table 5.3 shows the sentence-level results for abbreviation accuracy for both systems, while the sentence-level N-best results are shown in Table 5.4. These results at first seem surprisingly poor considering the good performance on abbreviations alone. However, when one examines the WER and FP rates shown in Table 5.5 the reason is evident. Recall that the original WER of the DEL test set is 9.49%. The deletion models perform quite poorly with regard to false positives. In many cases, words already in standard form are transformed

Table 5.3. Percentage of abbreviations correctly normalized in top-n sentence hypotheses for the deletion-based systems.

		ME (DEL)	**CRF (DEL)**
Top-1	**Unigram**	41.89	50.37
	Bigram	44.00	51.69
	Trigram	44.60	52.48
Top-3	**Unigram**	43.40	51.85
	Bigram	45.71	53.08
	Trigram	46.43	53.92
Top-10	**Unigram**	44.48	52.84
	Bigram	47.15	54.67
	Trigram	47.43	55.35
Found	**Unigram**	45.32	53.20
	Bigram	47.78	55.47
	Trigram	47.86	55.67

Table 5.4. Sentence-level top-N accuracy (%) using lattice decoding with deletion-based systems.

		ME (DEL)	**CRF (DEL)**
Top-1	**Unigram**	2.90	0.05
	Bigram	15.58	1.54
	Trigram	21.51	4.68
Top-3	**Unigram**	5.21	0.11
	Bigram	22.11	2.25
	Trigram	27.56	6.81
Top-10	**Unigram**	8.71	0.41
	Bigram	27.91	3.49
	Trigram	32.06	9.78
Found	**Unigram**	11.08	0.53
	Bigram	31.29	4.09
	Trigram	36.81	11.32

Table 5.5. False positive (FP) rate (%) and final WER in top-1 sentences when using deletion-based systems.

	ME (DEL)		**CRF (DEL)**	
	FP	WER	FP	WER
Unigram	23.33	26.6	52.33	52.1
Bigram	7.95	12.5	38.64	39.6
Trigram	4.49	9.3	27.82	29.7

to other English words bringing the overall sentence-level accuracy down drastically. Interestingly, while the CRF classifier outperforms ME in terms of accuracy on abbreviations, the sentence level accuracy is slightly worse and it performs significantly worse in terms of false positives and final WER. This may be worth investigating in future work.

Table 5.6. Accuracy (%) in n-gram context tests when using the deletion-based AMs for pruning only.

		ME (DEL)	**CRF (DEL)**
Top-1	**Unigram**	43.91	45.56
	Bigram	44.73	46.15
	Trigram	45.87	47.52
Top-3	**Unigram**	47.95	49.84
	Bigram	48.50	50.39
	Trigram	48.46	50.39
Top-10	**Unigram**	49.13	51.17
	Bigram	49.13	51.09
	Trigram	49.13	50.39
Found		49.13	51.17

5.3.3 Model Pruning Performance

Finally, results are presented when using the AM to generate the candidate standard forms for each abbreviation, but the AM scores are not combined with the LM scores for decoding. The results for the n-gram context tests are shown in Table 5.6. As expected, this method

Table 5.7. Percentage of abbreviations correctly normalized in top-N sentence hypotheses when using the deletion-based systems for pruning only.

		ME (DEL)	**CRF (DEL)**
Top-1	**Unigram**	42.57	38.63
	Bigram	43.09	46.95
	Trigram	43.76	48.78
Top-3	**Unigram**	45.55	45.04
	Bigram	46.35	51.45
	Trigram	46.75	52.72
Top-10	**Unigram**	47.15	50.45
	Bigram	47.86	54.67
	Trigram	47.98	55.67
Found	**Unigram**	47.82	52.25
	Bigram	48.14	56.07
	Trigram	48.22	56.31

Table 5.8. Sentence-level top-N accuracy (%) using lattice decoding when using deletion-based systems for pruning only.

		ME (DEL)	**CRF (DEL)**
Top-1	**Unigram**	19.73	5.98
	Bigram	23.94	18.25
	Trigram	24.71	23.53
Top-3	**Unigram**	26.49	11.79
	Bigram	29.75	27.20
	Trigram	30.82	29.93
Top-10	**Unigram**	30.46	17.72
	Bigram	33.43	32.89
	Trigram	34.73	37.10
Found	**Unigram**	31.89	21.99
	Bigram	36.51	35.74
	Trigram	40.13	39.30

gives a vast improvement over the LM only or AM only methods, but does not perform quite as well as the full system. However, on devices that have ample memory but low computational power, this may be a good compromise.

The sentence-level results show a vast improvement for the deletion-based models. The sentence-level abbreviation accuracy, sentence-level N-best accuracy and WER/FP rates are given in Tables 5.7, 5.8 and 5.9, respectively. Though the abbreviation accuracy at the sentence level is similar to that seen with the full system, the false positive rate is greatly reduced. This indicates that the AM scores are probably inaccurate and too high even though the mappings produced are reasonable. This decreased FP rate leads to a vast improvement in both sentence-level accuracy and WER.

Table 5.9. False positive (FP) rate (%) and final WER in top-1 sentences when using deletion-based systems for pruning only.

	ME (DEL)		**CRF (DEL)**	
	FP	WER	FP	WER
Unigram	4.28	9.3	13.97	18.6
Bigram	2.74	7.8	6.35	10.8
Trigram	2.29	7.4	4.92	9.3

5.4 Feature Selection

Although the CRF classifier consistently outperformed ME in terms of abbreviation accuracy, it is possible that its accuracy could be increased further by use of feature selection or optimization of the length model weight. For this reason, the weight for the length model and the set of features used are tuned simultaneously as follows. Both forward and backward feature selection are used to try to find the most optimal setup. For each feature set tested during feature selection, the generated list is re-ranked using the various weights for the length model described in Section 5.1.3 and then prune the list to length $n-1$. The look-up table's coverage of the development set is calculated and the feature/weight combination

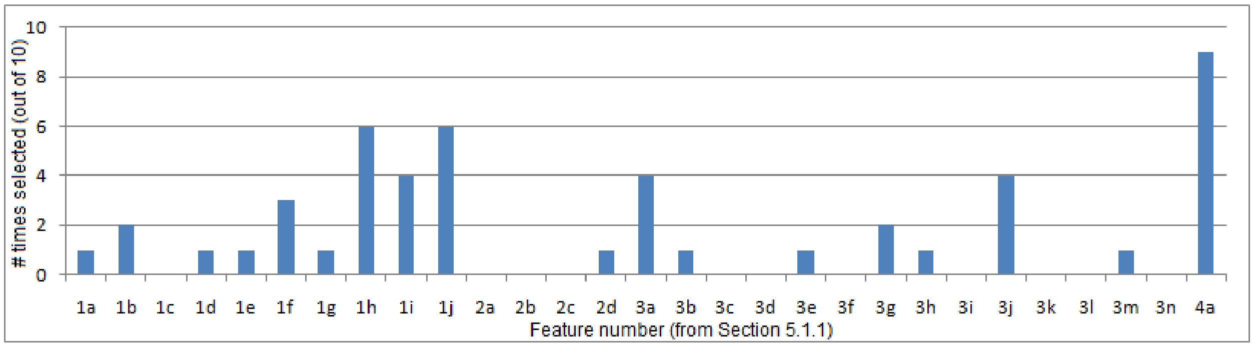


Figure 5.3. Distribution of chosen features (from Section 5.1.1) across the 10 development experiments.

that yields the maximum coverage is used during the testing phase. In general, the forward selection setup yielded slightly higher coverage on the development data than the backward selection setup. During backward decoding, it was often found that removing even one feature decreased performance. The set of all features outperformed the forward selection set by a fair margin (over 10 percentage points in some tests).

Because it is interesting to note which features were found to be useful to the model, a histogram showing the number of times each feature was selected over the 10 development experiments using forward selection is shown in Figure 5.3. Knowledge of the previous character's classification is very useful, and trigram context information is also important. Of the syllable related features, the actual characters in the syllable are selected most often.

To select a weight for the length model, the score model weight was held steady at 1 and the length model weight was tested on values from 0.01 to 100. The best performing weight varied so much during the development tests that it is difficult to suggest a good weight to choose for future tests. Histograms of the chosen weights are shown in Figure 5.4. After the forward selection setup, the length weights are well-centered around 1. The backward selection weights appear more spread out, but are still somewhat centered on a weight of 1.

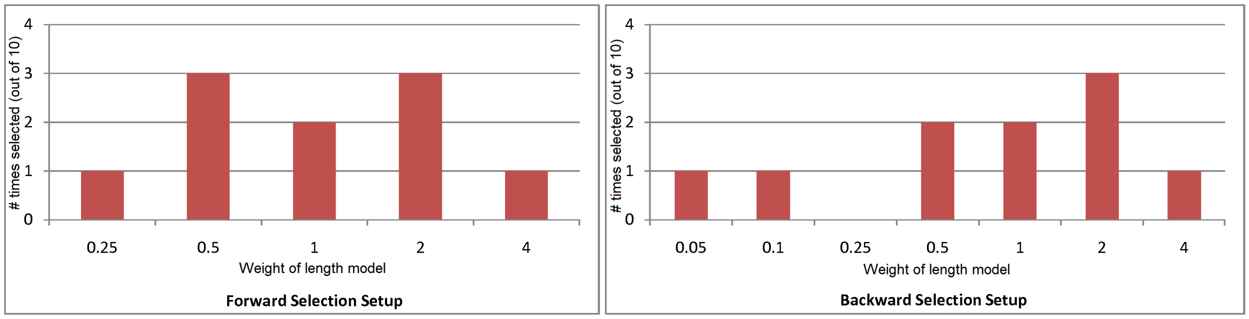


Figure 5.4. Distribution of chosen weight for the length model across the 10 development tests.

5.5 Chapter Summary

The first abbreviation model presented uses statistical classifiers (maximum entropy and conditional random fields) to learn the likelihood of removing a given character from a word in order to form an abbreviation. The features for the classifiers were designed by examining abbreviations found in the corpus and fall into three feature categories: Context Features, which use the characters surrounding the character in question; Function Features, which indicate whether the character is a vowel, double letter, or silent “e”; and Syllable Features, which look at the characters in the syllable, the syllable’s position in the word, and the character’s position in the syllable. The posterior probabilities from the classifier can be decoded using the Viterbi algorithm to determine for a given word the most likely abbreviations that can be formed using deletions only. Hypotheses are then re-ranked using a length model and stored in a look-up table for easy access. While this method is able to accurately translate approximately half of the deletion-based abbreviations in the data, it suffers from a high false positive rate that brings down its accuracy when decoding a full message.

CHAPTER 6
ABBREVIATION MODELING THROUGH CHARACTER-LEVEL MT[1]

Typically, an SMT system translates a sentence from one language to another. An alignment step learns a mapping of words and phrases between the two languages using a training corpus of parallel sentences. During testing, this mapping is used along with LMs to translate a sentence from one language to another. While researchers have used this method to normalize abbreviations (Bangalore et al., 2002; Aw et al., 2006; Kobus et al., 2008; Q. and H., 2009; Contractor et al., 2010), it is not robust to new words and leads to poor accuracy in this domain where new terms are used frequently and inconsistently.

The method described in this chapter differs in that the MT model is trained at the character-level to generate an AM; that is, rather than learning mappings between words and phrases, characters are mapped to other characters in what can be a many-to-many mapping. In this way, the model is able to learn common character-level changes that occur during abbreviation, regardless of whether the word in which it appears was seen during training. For example, the ending "-er" (as in "teacher" or "brother") is often abbreviated with the single character 'a'. Characters may also be mapped to symbols ("@" for "at"), numbers ("8" for "ate") or nothing at all (deletions).

6.1 Model Description

Formally, for an abbreviation $a : c_1(a), c_2(a), ..., c_m(a)$ (where $c_i(a)$ is the i^{th} character in the abbreviation), an MT system is used to find the proper word hypothesis:

[1]©2011 AFNLP. Portions reprinted, with permission, from: Pennell, Deana L. and Yang Liu, A Character-Level Machine Translation Approach for Normalization of SMS Abbreviations, in *Proceedings of IJCNLP*, November 2011.

$$\begin{aligned} \hat{w} &= \arg\max p(w|a) \\ &= \arg\max p(w)p(a|w) \\ &= \arg\max p(c_1(w), ...c_n(w)) \\ &\quad \times p(c_1(a), ..., c_m(a)|c_1(w), ...c_n(w)) \end{aligned} \tag{6.1}$$

where $c_i(w)$ is a character in the English word, $p(c_1(w), ...c_n(w))$ is obtained using a character language model, and $p(c_1(a), ..., c_m(a)|c_1(w), ...c_n(w))$ is based on the learned phrase translation table. Note that this is very similar to Equation 4.1, except that this works at the character level rather than at the word level.

To train the translation model, pairs are taken from the annotated data: an abbreviation and its corresponding English word. Spaces are replaced with the underscore character and then spaces are inserted between each character in order to facilitate character-level training.

```
a b ||| _ a b ||| 0.714286 0.880834 0.018315 0.164514 2.718
a b ||| a _ b ||| 0.5 0.880834 0.010989 0.164514 2.718
a b ||| a b b ||| 0.181818 0.880834 0.00732601 0.905748 2.718
a b ||| a b o u ||| 0.00714286 0.880834 0.003663 0.00346958 2.718
a b ||| a b o ||| 0.00662252 0.880834 0.003663 0.0690538 2.718
a b ||| a b u l a r y ||| 1 0.440819 0.003663 4.55505e-08 2.718
a b ||| a b ||| 0.894737 0.880834 0.934066 0.934993 2.718
a b ||| a p ||| 0.00956938 0.000874125 0.00732601 0.000887968 2.718
a b ||| a v e ||| 0.00249377 0.000970303 0.003663 3.35128e-05 2.718
a b ||| b ||| 0.000325839 0.308415 0.003663 0.968721 2.718
a b ||| i b ||| 0.0238095 0.308415 0.003663 0.077216 2.718
```

Figure 6.1. An excerpt from a phrase table showing possible translations when the character sequence "ab" is found in a message.

Because the translation system is trained at the character-level, each hypothesis (w) for an abbreviation (a) is a sequence of characters, which may or may not produce a valid word. To see why this is possible, examine the partial phrase-table[2] shown in Figure 6.1; using this

[2]Aside from the possible translations, the phrase table also shows five values for each word. They correspond to the inverse phrase translation probability $\phi(f|e)$, the inverse

table, “hab” could be translated to “hib”, “habulary” or “habou” as well as the correct word “have”. It is also possible, and in fact very likely, for a hypothesis to appear many times in the N-best hypothesis list due to different segmentation (two characters may be generated by a single phrase mapping or by two different mappings, one for each character). Thus, the MT system is used to generate the top 20 *distinct* hypotheses for each abbreviation and then those hypotheses that do not occur in the CMU Lexicon are eliminated.

The abbreviation score $P(a_i|w_i)$ from Equation 4.1 represents the likelihood that abbreviation a_i is derived from word w_i, and can be obtained from:

$$p(a_i|w_i) \propto \frac{p(w_i|a_i)}{p(w_i)} \tag{6.2}$$

where $p(w_i|a_i)$ is the score from the character-level MT system, and $p(w_i)$ is from the character LM used in MT decoding. In this work, the score from the character MT system is used as the likelihood score without dividing by the character-level LM contribution. This is equivalent to using both a character-level and a word-level LM during decoding.

6.2 MT System Setup

The popular open source SMT implementation, Moses (Koehn et al., 2007), is used for all experiments. The probability of swaps was lowered in Moses' parameters; while swaps are common in translating between different languages due to different grammatical word orders, they are relatively uncommon when forming abbreviations. Early in system development, character language models of orders 3, 5 and 7 were built using OOV words from the Edinburgh Twitter corpus Petrovic et al. (2010) and tested on the partially annotated data to determine the ideal order for the purposes of this work. A large increase in performance was seen when moving from order 3 to order 5, but when moving to order 7 there was only a very slight increase and occasionally a small decrease in performance. For this reason, an order 5 character language model was used in all experiments discussed below.

lexical weighting $lex(f|e)$, the direct phrase translation probability $\phi(e|f)$, the direct lexical weighting $lex(e|f)$ and the phrase penalty (always $exp(1) = 2.718$), where e and f are the English and foreign phrases, respectively. We do not currently make use of these values.

The MT system works by generating words from the abbreviations. Therefore it requires much less setup and tuning as we give the abbreviations in the test set to Moses directly after adding spaces between characters. As mentioned above, we generate the top 20 *distinct* hypotheses for each abbreviation and eliminate those hypotheses that do not occur in the CMU Lexicon. Once again, we use the development set to tune the weights for the abbreviation model (AM) and the language model (LM) and use the top performing weights for evaluation on the test set. Note that Moses may generate positive score (impossible for a real log-probability). When a negative value is expected, we use -0.1 instead, indicating that this pair is very likely.

Aside from evaluating the MT system on both the DEL and SW test sets, a third evaluation is performed that reuses the SW test set with a focus on improving performance on the *repetition* type. Similar to some past work (Han and Baldwin, 2011; Liu et al., 2011), these tokens are pre-processed to improve performance. The intuition behind the processing is that there is no standard number of times that a character is repeated when creating this type of token, and the MT system may not have enough examples with a given number of repeated characters to be able to generalize when new terms are seen. For this reason, all instances of repetition with length greater than two are reduced to have length two before training (*e.g.*, “yeeeeeesssssss” is reduced to “yeess”). The repetitions are also reduced as seen during testing before submitting the token to the MT system for hypothesis generation. This repetition focused system is denoted MT+R.

6.3 Experiments

The results from the experiments described in Section 4.2 are shown below. For comparison, the best results from Chapter 5 are shown. A full comparison of the results presented below with those presented in Chapters 4, 5 and 7 can be found in the Appendix.

6.3.1 Model Performance

The results for the AM without use of an LM are shown in Table 6.1. By examining the "found" row in the table, one can see that the MT model produces a higher coverage of the test set than the CRF model. This implies that (with proper re-ranking) it has the potential to obtain higher results in a final system. We also see that it performs significantly better at each step in the N-best list. However, if we take the number of abbreviations "found" as 100%, we see that the CRF model has a higher percentage of abbreviations in the top-1 (72.8) than the MT system for deletions (67.3). CRFs may therefore produce a slightly better initial model in terms of the original ranking of hypotheses. In addition, we see that pre-processing the repetitions does, in fact, lead to a slight improvement in performance for all values of n.

Table 6.1. Accuracy (%) of MT systems without LM information on abbreviations only.

	CRF (DEL)	**MT (DEL)**	**MT (SW)**	**MT+R (SW)**
Top-1	37.27	54.71	55.90	56.64
Top-3	43.75	74.15	72.81	74.38
Top-10	48.86	79.69	77.07	78.61
Found	51.17	80.72	77.74	79.26

Tests were also conducted to see how well the MT system performed on each type of abbreviation. Each abbreviation in the SW test set has been annotated with the formation methods corresponding to those listed in Table 3.6 and accuracy was computed for each type. Table 6.2 shows the top-1 accuracy for each abbreviation type for the two MT methods. Adding the pre-processing for repetition types does increase the repetition accuracy for the model alone, as well as giving a slight increase in the combination types (because some contain repetitions). There is a slight decrease in performance for most of the other types, though it is expected that with incorporation of the LM in decoding that difference can be mitigated.

Table 6.2. Accuracy (%) of the MT systems on each abbreviation type.

	Deletion	**Substitution**	**Repetition**	**Swap**	**Insertion**	**Combin.**
MT	54.71	43.96	74.55	87.73	45.76	50.24
MT+R	54.51	41.90	79.92	87.73	44.06	51.72

6.3.2 Full System Performance

The results for the n-gram context tests are shown in Table 6.3. During development, the best value of β was found to be the very low value 0.05, giving the LM score preference over the AM score.

Table 6.3. Accuracy of MT systems using both AM and LM scores on n-gram context tests.

		CRF (DEL)	**MT (DEL)**	**MT (SW)**	**MT+R (SW)**
Top-1	**Unigram**	45.48	62.09	64.19	65.91
	Bigram	46.81	69.59	69.69	71.32
	Trigram	47.44	70.58	70.44	71.93
Top-3	**Unigram**	49.84	79.45	76.91	78.50
	Bigram	50.39	76.39	75.17	76.69
	Trigram	50.39	77.57	75.87	77.39
Top-10	**Unigram**	51.09	80.71	77.73	79.25
	Bigram	51.09	80.71	77.73	79.25
	Trigram	51.09	80.71	77.73	79.25
Found		51.17	80.71	77.73	79.25

The sentence-level abbreviation accuracy, sentence-level N-best accuracy and WER/FP rates are given in Tables 6.4, 6.5 and 6.6, respectively. The sentence-level accuracy for the MT systems are a large improvement over the CRF system, especially as we look farther down the top-N list. Although the MT setup performs somewhat better than the CRF setup for abbreviation accuracy, the much lower false positive rate for MT is the major cause of this improvement. The MT system is able to achieve an overall decrease in WER for both

Table 6.4. Percentage of abbreviations correctly normalized in top-N sentence hypotheses when using the MT setup.

		CRF (DEL)	**MT (DEL)**	**MT (SW)**	**MT+R (SW)**
Top-1	**Unigram**	50.37	59.81	62.63	64.00
	Bigram	51.69	64.87	66.56	68.22
	Trigram	52.48	66.34	67.90	69.43
Top-3	**Unigram**	51.85	71.44	70.77	72.28
	Bigram	53.08	75.18	73.67	75.31
	Trigram	53.92	75.34	73.80	75.40
Top-10	**Unigram**	52.84	77.45	75.03	76.74
	Bigram	54.67	79.56	76.81	78.32
	Trigram	55.35	79.37	76.81	78.27
Found	**Unigram**	53.20	79.05	76.24	77.79
	Bigram	55.47	80.28	77.38	78.91
	Trigram	55.67	79.88	77.25	78.75

Table 6.5. Sentence-level top-N accuracy (%) using lattice decoding for MT.

		CRF (DEL)	**MT (DEL)**	**MT (SW)**	**MT+R (SW)**
Top-1	**Unigram**	0.05	31.47	31.36	32.07
	Bigram	1.54	37.16	36.55	37.82
	Trigram	4.68	38.64	37.98	39.08
Top-3	**Unigram**	0.11	43.09	40.83	41.94
	Bigram	2.25	48.60	45.54	46.85
	Trigram	6.81	49.49	46.49	47.88
Top-10	**Unigram**	0.41	50.50	46.45	47.60
	Bigram	3.49	54.83	50.93	52.87
	Trigram	9.78	58.56	54.37	55.72
Found	**Unigram**	0.53	53.23	49.18	50.77
	Bigram	4.09	59.45	55.92	56.91
	Trigram	11.32	66.56	62.49	63.48

datasets (recall the original WER was 9.49 and 11.6 for DEL and MT, respectively), but we still feel that it generates too many false positives. This problem is addressed in Chapter 7.

Once again, the MT system outperforms the CRF system by a large margin at all places in the top-N. However, the MT system (on DEL using a trigram context) reaches only 87.4% of its maximal performance, whereas the CRF system is able to reach 92.7%. This shows that there is more work that can be done to re-rank the MT system to obtain better results in the future. Increasing the order of the LM leads to an increase in performance across all setups. There is a consistent jump in accuracy when increasing the order from unigram to the bigram, with a smaller increase from bigram to trigram. All three MT setups are able to reach their maximum performance by the top-10 hypothesis. The slight improvement gained by using the MT+R setup continues to hold once the LM is applied.

Table 6.6. False positive (FP) rate (%) and final WER in top-1 sentences for MT.

	CRF (DEL)		**MT (DEL)**		**MT (SW)**		**MT+R (SW)**	
	FP	WER	FP	WER	FP	WER	FP	WER
Unigram	52.33	52.1	4.14	7.6	4.07	7.9	4.19	7.9
Bigram	38.64	39.6	3.61	6.6	3.50	7.0	3.58	6.8
Trigram	27.82	29.7	3.56	6.4	3.44	6.8	3.52	6.6

6.3.3 Model Pruning Performance

Finally, results are presented when using the AM to generate the candidate standard forms for each abbreviation, but the AM scores are not combined with the LM scores for decoding. The results for the n-gram context tests are shown in Table 6.7. Interestingly, we see an increase in performance for the MT system, indicating that the scores associated with the pairs in the MT setup may not be appropriate. This slight improvement also holds for abbreviation accuracy at the sentence level as well (shown in Table 6.8.

The sentence-level N-best accuracy is given in Tables 6.9 While there was an increase in performance for the deletion systems, instead there is a sharp decline in performance for all of the MT setups, bringing them down nearly to the level of the CRF-for-pruning setup. False positives are the culprit once more. Examination of Table 6.10 shows a sharp increase in false positives over the full system results for the MT setups, with a worse FP rate and final WER than the CRF setup. The context information helps significantly during decoding; a large jump in performance is evident between the unigram and bigram language models.

Table 6.7. Accuracy (%) of MT systems on abbreviations only using AM score for pruning only.

		CRF (DEL)	**MT (DEL)**	**MT (SW)**	**MT+R (SW)**
Top-1	**Unigram**	45.56	60.36	62.76	64.53
	Bigram	46.15	69.04	69.42	71.00
	Trigram	47.52	70.34	70.35	71.82
Top-3	**Unigram**	49.84	79.40	76.82	78.41
	Bigram	50.39	76.39	75.22	76.73
	Trigram	50.39	77.29	75.76	77.28
Top-10	**Unigram**	51.17	80.71	77.73	79.25
	Bigram	51.09	80.71	77.73	79.25
	Trigram	50.39	80.71	77.73	79.25
Found		51.17	80.71	77.73	79.25

6.4 Chapter Summary

The second abbreviation model presented uses a machine translation (MT) system trained at the character level to find the words that are most likely to generate a given abbreviation. Training at the character level rather than the word level allows the system to be more robust to new terms as it can recognize a common abbreviation pattern regardless of whether the word in which it appears was seen during training. The MT model significantly outperforms the deletion model both in terms of the number of abbreviations it is able to correct and the number of false positives generated. To more accurately handle repetitions in informal text, a heuristic is implemented whereby characters with more than two consecutive repetitions

Table 6.8. Percentage of abbreviations correctly normalized in top-N sentence hypotheses when using the MT AMs for hypothesis generation only.

		CRF(DEL)	**MT (DEL)**	**MT (SW)**	**MT+R (SW)**
Top-1	**Unigram**	38.63	59.85	62.36	64.09
	Bigram	46.95	67.26	68.15	69.75
	Trigram	48.78	68.73	69.22	70.70
Top-3	**Unigram**	45.04	63.28	65.03	66.87
	Bigram	51.45	72.16	71.84	73.55
	Trigram	52.72	73.27	72.87	74.51
Top-10	**Unigram**	50.45	69.37	69.36	71.21
	Bigram	54.67	75.50	74.37	76.04
	Trigram	55.67	77.41	76.45	77.18
Found	**Unigram**	52.25	72.04	71.48	73.51
	Bigram	56.07	77.41	75.45	77.34
	Trigram	56.31	78.57	76.45	78.07

Table 6.9. Sentence-level top-N accuracy (%) using MT AMs for pruning.

		CRF(DEL)	**MT (DEL)**	**MT (SW)**	**MT+R (SW)**
Top-1	**Unigram**	5.98	6.57	7.52	8.67
	Bigram	18.25	24.89	25.10	27.00
	Trigram	23.53	31.89	31.64	33.18
Top-3	**Unigram**	11.79	10.84	12.07	13.54
	Bigram	27.20	36.69	35.08	37.26
	Trigram	29.93	41.19	39.64	41.94
Top-10	**Unigram**	17.72	16.89	17.82	19.80
	Bigram	32.89	43.27	42.01	44.31
	Trigram	37.10	49.49	46.69	48.63
Found	**Unigram**	21.99	20.45	21.70	24.07
	Bigram	35.74	47.30	45.94	48.31
	Trigram	39.30	56.90	53.34	55.84

Table 6.10. FP rate (%) and final WER in top-1 sentences using the MT AMs for pruning.

	CRF (DEL)		**MT (DEL)**		**MT (SW)**		**MT+R (SW)**	
	FP	WER	FP	WER	FP	WER	FP	WER
Unigram	13.97	18.6	19.82	21.7	19.72	21.8	18.58	20.6
Bigram	6.35	10.8	8.39	10.7	8.10	10.9	7.85	10.4
Trigram	4.92	9.3	6.53	8.8	6.33	9.2	6.25	8.9

are truncated to only two. This pre-processing step yields a slight increase in performance throughout the evaluations.

CHAPTER 7
SYSTEM EXTENSIONS

This chapter describes two extensions to the methods described in Chapters 5 and 6, as well as providing a comparison of the methods described in this dissertation to past work on a small standard data set. Section 7.1 describes several approaches for combining the deletion and MT methods to create a single abbreviation model. The high false positive rate is addressed in Section 7.2 by using a dictionary to determine whether a term should be expanded.

7.1 System Combinations

Overall, the MT system always outperforms the CRF system, even when tested on solely the deletion abbreviations for which the CRF was designed. When examining the abbreviations that each system was able to correct, there are slight differences between the two methods. For this reason, by combining the two systems into one system a higher performance than either single setup may be achieved.

7.1.1 Methods of Combining the Systems

Assuming for a test case (i.e., an abbreviation A), each system generates a set of word hypotheses, $\mathbf{H_1}$ and $\mathbf{H_2}$ respectively. For a hypothesis W in these sets, the system also provides a score, $S_i(W, A)$ $(i = 1, 2)$. For a combined system, there are multiple ways to generate the word hypotheses (set $\mathbf{H_3}$) and their scores (function $S_3(W, A)$). The following describes the attempted methods for combining information from two systems.

1. **Weighted Average.** In this method, the word candidates are the union of the two systems: $\mathbf{H_3} = \mathbf{H_1} \cup \mathbf{H_2}$. For word candidates that appear in only one system, that

system's scores are used; for a word appearing in both systems, the weighted average of the two scores is taken, that is:

$$S_3(W,A) = \begin{array}{ll} S_1(W,A) & \text{if } W \in \mathbf{H_1} - \mathbf{H_2} \\ S_2(W,A) & \text{if } W \in \mathbf{H_2} - \mathbf{H_1} \\ \alpha S_1(W,A) + \beta S_2(W,A) & \text{if } W \in \mathbf{H_1} \cap \mathbf{H_2} \end{array} \tag{7.1}$$

where α and β are determined empirically on the development set.

2. **Take Highest.** The set H_3 is similar to the method above in that it is formed using the superset of $\mathbf{H_1}$ and $\mathbf{H_2}$. The difference is that for the words that fall in the intersection, the highest score is used: $S_3(W,A) = max(S_1(W,A), S_2(W,A))$if $W \in \mathbf{H_1} \cap \mathbf{H_2}$.

3. **System Preference.** The set H_3 is once again the superset, but to determine the score for a word in the intersection, a preferred system is defined such that its scores are always used. Let I be the preferred system (1 or 2), then $S_3(W,A) = S_I(W,A)$if $W \in \mathbf{H_1} \cap \mathbf{H_2}$.

4. **Case-Specific Knowledge.** This is specifically for cases where one of the systems being tested is a deletion-based system. In this example, let $\mathbf{H_1}$ be the set generated by the deletion-based system and $\mathbf{H_2}$ be generated by some other system. For each candidate, the system is given knowledge of whether it is actually a deletion abbreviation or not. For those that are deletion abbreviations, if $\mathbf{H_1} \neq \emptyset$, the prediction from H_1 and S_1 are used. If $\mathbf{H_1} = \emptyset$ or the abbreviation is formed by a method other than deletions, use the prediction from $\mathbf{H_2}$ and $\mathbf{S_2}$.

   Since the CRF system performs worse than MT, this test may seem counterintuitive. Remember, however, that the CRF method has high precision despite its low recall. Therefore, it may achieve poor results not because it produces incorrect translations, but rather because it has no guess at all for an abbreviation and thus leaves it as-is (which is also considered incorrect). For this reason, we hope that when the CRF system has a guess it will be correct, and when it has no guess the MT system can help.

Table 7.1. Accuracy (%) in n-gram context tests using system combinations.

		System under test					
		MT	**1**	**2**	**3a**	**3b**	**4**
Top-1	**Unigram**	64.19	66.09	64.09	63.23	63.30	41.26
	Bigram	69.69	70.64	70.44	70.26	70.16	64.07
	Trigram	70.44	71.61	71.39	71.32	71.34	64.91
Top-3	**Unigram**	76.91	78.57	78.05	77.84	77.87	51.52
	Bigram	75.17	77.14	76.96	76.94	76.89	68.56
	Trigram	75.87	77.80	77.59	77.50	77.50	68.62
Top-10	**Unigram**	77.74	80.10	80.10	80.09	80.09	57.75
	Bigram	77.74	80.06	80.04	80.02	80.02	69.12
	Trigram	77.74	80.04	80.02	80.00	80.00	69.10
Found		77.74	80.10	80.10	80.10	80.10	69.16

The n-gram context tests were performed on the combinations listed above using the methods described in Section 4.2 and results are shown in Table 7.1. The system numbers correspond to their numbers in the list above; **3a** refers to preferring the CRF system and 3b refers to preferring the MT system. For comparison, the results from the MT+R system are also given. In addition, we use the dictionary heuristic on the top-performing system.

During development, we first run the tests using only trigram context on the development and optimize the top-1 accuracy. As before, we set $\alpha = 1$ and vary β from 0.01 to 100. Once again, lower weights for β give better performance. Generally, 0.1 was selected as the best weight, although 0.5 and 0.1 were occasionally chosen as well.

The averaging system performed best when setting the weight for the MT system to 1 and the deletion system to 0.25. The results shown are from that setup. The averaging method performed slightly better than the MT system during context tests. While that advantage holds for abbreviations during full message decoding, the higher false positive rate causes an overall decrease in performance.

The sentence decoding results are shown in Tables 7.2–7.5. Excepting the case-specific method, the combinations perform well on abbreviations but suffer from a high false positive

Table 7.2. Percentage of abbreviations correctly normalized in top-N sentence hypotheses when using system combinations.

		System under test					
		MT	**1**	**2**	**3a**	**3b**	**4**
Top-1	**Unigram**	62.63	64.34	64.53	63.98	59.26	52.60
	Bigram	66.56	67.90	67.56	67.49	65.42	54.84
	Trigram	67.90	69.40	69.09	69.09	68.01	55.50
Top-3	**Unigram**	70.77	68.13	69.47	68.20	65.35	56.21
	Bigram	73.67	73.90	74.19	73.14	72.66	58.03
	Trigram	73.80	74.81	74.94	74.17	74.37	58.08
Top-10	**Unigram**	75.03	73.26	73.67	72.41	71.05	57.73
	Bigram	76.81	77.75	77.91	76.93	77.36	59.22
	Trigram	76.81	77.75	77.84	77.54	77.79	59.19
Found	**Unigram**	76.24	75.19	75.70	74.31	73.74	58.42
	Bigram	77.38	78.73	78.75	78.11	78.55	59.47
	Trigram	77.25	78.93	79.00	78.50	78.82	59.58

rate during sentence decoding. The case-specific method performs somewhat worse on abbreviations, but performs competitively with respect to other metrics due to its much lower false positive rate. However, this method still achieves a much lower full sentence accuracy than the original MT method because it is unable to compete on abbreviated terms.

The same combination experiments were then repeated using the MT+R setup. Examining the results for n-gram context tests shown in Table 7.6 shows a slight increase in performance across all tests.

Table 7.3. False positive rate (%) among the various combination methods.

	System under test					
	MT	**1**	**2**	**3a**	**3b**	**4**
Unigram	4.07	10.01	12.34	17.26	10.31	4.07
Bigram	3.50	4.88	5.00	7.44	4.87	3.56
Trigram	3.44	4.40	4.43	5.92	4.40	3.47

Table 7.4. WER when using the different combination methods.

	System under test					
	MT	**1**	**2**	**3a**	**3b**	**4**
Unigram	7.9	13.0	15.0	19.4	13.8	9.1
Bigram	6.9	8.0	8.1	10.3	8.3	8.4
Trigram	6.7	7.4	7.5	8.8	7.6	8.2

Table 7.5. Top-n sentence accuracy (%) for the combination methods.

		System under test					
		MT	**1**	**2**	**3a**	**3b**	**4**
Top-1	**Unigram**	31.36	17.02	13.42	8.59	15.04	24.79
	Bigram	36.55	32.43	32.15	26.65	31.36	27.64
	Trigram	37.98	36.00	35.92	32.19	35.48	28.39
Top-3	**Unigram**	40.83	23.08	21.06	13.98	23.20	29.86
	Bigram	45.54	42.25	42.37	37.02	41.26	33.26
	Trigram	46.49	44.67	44.67	41.30	44.19	33.90
Top-10	**Unigram**	46.45	31.16	29.46	22.17	30.65	33.02
	Bigram	50.93	48.23	47.96	43.60	48.11	36.47
	Trigram	54.37	51.40	51.72	48.43	51.20	38.93
Found	**Unigram**	49.18	35.36	34.41	26.37	34.97	34.97
	Bigram	55.92	52.11	52.23	47.72	52.19	40.31
	Trigram	62.49	57.06	56.79	53.70	57.42	44.95

Table 7.6. Accuracy (%) during n-gram context tests using system combinations with MT+R.

		System under test					
		MT+R	**1**	**2**	**3a**	**3b**	**4**
Top-1	**Unigram**	65.91	65.09	65.68	65.54	65.09	41.92
	Bigram	71.32	71.73	71.95	71.98	71.77	65.59
	Trigram	71.93	72.84	72.93	72.86	72.72	66.38
Top-3	**Unigram**	78.50	79.47	79.61	79.52	79.43	53.06
	Bigram	76.69	78.45	78.48	78.43	78.41	70.07
	Trigram	77.39	79.07	79.16	79.11	79.07	70.14
Top-10	**Unigram**	79.26	81.60	81.63	81.60	81.54	59.25
	Bigram	79.26	81.54	81.54	81.54	81.54	70.64
	Trigram	79.26	81.54	81.54	81.54	81.54	70.62
Found		79.26	81.64	81.64	81.64	81.64	70.67

However, at the sentence level, there is generally a decreased performance using the MT+R system instead of the simple MT system, even though the MT+R shows an increase when used alone. This implies that, while the score ranking is internally better for the MT+R system, its scores may not be matched as well with the CRF scores. Perhaps normalizing or otherwise weighting the two systems before implementing the combination methods could help achieve a higher accuracy. The accuracy on abbreviations when decoding at the sentence level is generally lowered for all tests except the case-specific method, with an occasional very slight increase with trigram LM, as shown in Table 7.7. This appears to be due, at least in part, to a cascading effect. When a word is incorrectly decoded, it also causes errors when it appears in the context of other words. The upper bound is raised slightly for most of the tests.

The false positive rate is also negatively affected by the change in the MT system (again with the exception of the case-specific model), as shown in Table 7.8. Using higher order language models helps decrease the false positive rate, but even with a trigram model the rate is still higher than when the unmodified MT method is utilized in the combinations. A

Table 7.7. Percentage of abbreviations correctly normalized in top-N sentence hypotheses when using system combinations with MT+R.

		System under test					
		MT+R	**1**	**2**	**3a**	**3b**	**4**
Top-1	**Unigram**	64.00	54.38	63.42	62.89	55.42	53.48
	Bigram	68.22	64.55	67.69	67.30	64.58	55.97
	Trigram	69.43	68.18	69.68	69.31	68.04	56.57
Top-3	**Unigram**	72.28	57.33	66.95	65.62	59.78	57.17
	Bigram	75.31	68.92	73.41	72.54	70.46	59.18
	Trigram	75.40	72.75	74.89	74.11	73.41	59.20
Top-10	**Unigram**	76.74	60.22	71.38	69.54	65.06	58.83
	Bigram	78.32	73.65	77.52	76.16	75.42	60.33
	Trigram	78.27	76.69	78.28	77.43	77.31	60.19
Found	**Unigram**	77.79	62.57	73.46	71.85	68.13	59.52
	Bigram	78.91	75.70	79.18	77.94	77.64	60.52
	Trigram	78.75	77.94	79.46	78.72	79.00	60.68

corresponding increase in WER is seen in Table 7.9. The top-N sentence (Table 7.10) is also affected negatively in all but the case-specific combination.

Table 7.8. False positive rate (%) among the various combination methods with MT+R.

	System under test					
	MT+R	**1**	**2**	**3a**	**3b**	**4**
Unigram	4.19	32.42	17.49	25.53	17.28	4.07
Bigram	3.58	11.34	6.77	10.87	7.04	3.50
Trigram	3.52	7.37	5.28	7.68	5.42	3.43

The case-specific method shows an increase in performance across the board for tests at the sentence level when using MT+R in combinations; it is less affected by the change in the MT approach. This seems to imply that the scores for deletions were highly affected by the repetition pre-processing. Although we did not see a large increase in accuracy on deletion abbreviations in Table 6.2, the scores may change without affecting the relative ordering of

Table 7.9. WER when using the different combination methods with MT+R.

	System under test					
	MT+R	**1**	**2**	**3a**	**3b**	**4**
Unigram	7.8	33.9	19.6	26.8	20.4	8.9
Bigram	6.8	14.1	9.7	13.3	10.3	8.6
Trigram	6.6	10.1	8.1	10.3	8.4	8.0

Table 7.10. Top-*n* sentence accuracy (%) for the combination methods with MT+R.

		System under test					
		MT+R	**1**	**2**	**3a**	**3b**	**4**
Top-1	**Unigram**	32.07	2.21	8.87	4.07	8.43	25.90
	Bigram	37.82	18.29	28.07	20.39	26.13	29.58
	Trigram	39.08	29.30	34.77	29.50	33.74	30.29
Top-3	**Unigram**	41.94	4.67	14.37	8.11	13.58	31.64
	Bigram	46.85	27.84	39.00	30.57	37.14	35.16
	Trigram	47.88	38.29	44.15	38.81	42.25	35.96
Top-10	**Unigram**	47.60	7.56	20.87	12.79	19.36	34.85
	Bigram	52.87	36.19	46.29	38.65	44.15	38.77
	Trigram	55.72	46.05	51.80	47.20	50.69	40.91
Found	**Unigram**	50.77	10.53	24.35	15.72	22.69	37.06
	Bigram	56.91	41.50	50.81	43.04	49.50	42.33
	Trigram	63.48	52.99	56.75	52.19	56.55	47.08

the pairs. However, even though the case-specific model shows an improvement when using MT+R, it still does not achieve the performance of the MT+R system alone.

7.2 Dictionary Heuristic for the Reduction of False Positives

The high false positive rate of the systems presented in the previous chapters is problematic if these systems are to be used in real world situations. One possible solution to this problem is to use a dictionary to decide whether or not a token should be expanded. Using this heuristic, if a token appears in the chosen dictionary, it is assumed that it is already in standard form and should not be expanded. If the token is not in the dictionary, the hypothesis from the system under test is used. Because false positives do not come into play during the n-gram context tests, only the sentence-level evaluations are performed for this setup. Four dictionaries were tested in combination with the previous best system (MT+R) to see which would yield the best performance.

1. **cmu**: The CMU lexicon used in previous experiments. This is a large, standard lexicon for TTS.
2. **cmu***: The portion of the CMU lexicon that overlaps with entries found at dictionary.com, as used when generating the syllable features in Section 5.1.1.
3. **asp**: The Aspell dictionary used instead of Jazzy's default dictionary in Section 4.3.2.
4. **jaz**: The default dictionary used by the Jazzy spell-checking system.

The dictionaries were incorporated during lattice creation for input to the SRILM lattice tool. When building the lattice, only one link is generated for any token that is in the dictionary being used. If the token has an associated score in the AM under test, that score is used in the lattice; otherwise the token is added with a default score of -0.1. For tokens that do not exist in the dictionary, the same procedure is followed as described in Section 4.2. In Figure 7.2, we show a lattice generated by this process for the message "green tea fraps r guuud :)". Note that this is the same sentence used to build the lattice in Figure 4.2. By using this heuristic, the dictionary used to build the lattice in Figure 7.2 was able to almost halve the number of links in the lattice. Originally, there were 5 links for the token "green" and 14 for the token "tea"; this heuristic lowers than number to 1 for both because they are dictionary tokens. This guarantees that it is impossible for these words to be translated into some other English word, thereby lowering the false positive rate.

The results in Table 7.11 shows that the heuristic is a success. At the unigram level, the FP rate is almost halved for three of the dictionaries and still significantly lowered for the fourth. Adding higher order language models is not very helpful when a dictionary is used;

```
N=8     L=24
I=0
I=1
I=2
I=3
I=4
I=5
I=6
I=7
J=0     S=0     E=1     W=<s>     a=−0.1
J=1     S=1     E=2     W=green   a=−0.952995
J=2     S=2     E=3     W=tea     a=−2.77546
J=3     S=3     E=4     W=fraps   a=−0.1
J=4     S=4     E=5     W=ur      a=−7.70762
J=5     S=4     E=5     W=er      a=−6.34979
J=6     S=4     E=5     W=r       a=−4.55793
J=7     S=4     E=5     W=re      a=−5.02034
J=8     S=4     E=5     W=are     a=−6.01279
J=9     S=4     E=5     W=tor     a=−6.47094
J=10    S=4     E=5     W=ar      a=−5.77696
J=11    S=4     E=5     W=our     a=−6.56922
J=12    S=4     E=5     W=ro      a=−6.08681
J=13    S=4     E=5     W=or      a=−6.73477
J=14    S=4     E=5     W=ra      a=−6.28262
J=15    S=4     E=5     W=ear     a=−5.45143
J=16    S=4     E=5     W=orr     a=−7.92868
J=17    S=5     E=6     W=god     a=−7.16848
J=18    S=5     E=6     W=guide   a=−7.21387
J=19    S=5     E=6     W=guard   a=−5.92715
J=20    S=5     E=6     W=good    a=−7.77716
J=21    S=5     E=6     W=gude    a=−7.32483
J=22    S=5     E=6     W=guuud     a=−7.77716
J=23    S=6     E=7     W=:)      a=−0.1
```

Figure 7.1. The lattice for the input sentence “green tea fraps r guuud :)” after application of the dictionary heuristic

the decrease in FP rate with higher order LMs is very small. The difference between the dictionary methods and the MT+R method is much smaller when a trigram language model is used, but is still a worthwhile improvement in all cases.

Table 7.11. False positive rate (%) after applying each of the four dictionaries.

		MT+R	**+cmu**	**+cmu***	**+asp**	**+jaz**
Top-1	Unigram	4.19	2.29	2.39	2.09	3.03
	Bigram	3.58	2.24	2.38	2.03	2.93
	Trigram	3.52	2.23	2.37	2.02	2.92

However, the lowered false positive rate does not lead to a similar improvement in the other sentence-level metrics. The sentence level N-best statistics are shown in Table 7.12. Interestingly, the dictionary that performed worst in terms of the false positive rate (**jaz**) has the best performance of all the dictionaries in terms of sentence accuracy. The Jazzy dictionary shows an increase in performance in top-1 accuracy for the lower order LMs, but that improvement is not seen across all levels of the table. The other dictionaries perform significantly worse than the MT+R system at all levels except the top-1 sentence with a unigram language model.

Table 7.12. Top-n sentence accuracy (%) after applying the dictionary heuristic.

		MT+R	**+cmu**	**+cmu***	**+asp**	**+jaz**
Top-1	Unigram	32.07	31.32	32.39	32.67	35.36
	Bigram	37.82	32.91	33.66	33.78	38.21
	Trigram	39.08	32.83	34.21	34.33	38.89
Top-3	Unigram	41.94	36.51	38.09	38.49	43.12
	Bigram	46.85	37.18	39.24	39.00	45.46
	Trigram	47.88	37.50	40.00	39.20	46.25
Top-10	Unigram	47.60	40.39	42.57	42.45	48.71
	Bigram	52.87	40.79	43.40	42.61	50.69
	Trigram	55.72	41.02	44.19	42.97	51.96
Found	Unigram	50.77	41.46	43.96	43.24	51.76
	Bigram	56.91	41.66	45.34	43.60	55.40
	Trigram	63.48	41.78	45.54	43.80	56.91

The WERs in Table 7.13 show a similar pattern in terms of the ranking of the dictionaries. All four outperform MT+R with a unigram LM, but only the Jazzy dictionary is able to maintain that advantage when using the higher order language models.

Table 7.13. Word error rates when applying the various dictionaries.

		MT+R	**+cmu**	**+cmu***	**+asp**	**+jaz**
Top-1	Unigram	7.9	7.6	7.5	7.4	7.1
	Bigram	6.8	7.4	7.3	7.2	6.6
	Trigram	6.7	7.3	7.2	7.1	6.5

To see the reason why the Jazzy dictionary outperforms the others in sentence accuracy and WER despite a worse performance regarding false positives, examine the accuracy on abbreviations in Table 7.14. The dictionaries **cmu**, **cmu*** and **asp** show a significant decrease in abbreviation normalization accuracy. This is due to an increase in false negatives when an abbreviated token is found in the dictionary being tested. For example, the CMU lexicon contains the token "im" as a valid word, preventing it from being properly expanded to the word "I'm". The Jazzy dictionary does not contain the token "im", so it can be properly expanded. The dictionaries often contain copious amounts of acronyms and abbreviations resulting in nonstandard words such as "ths" and "abv" to remain unnormalized. For this reason, the Jazzy dictionary is chosen as the best compromise for decreasing the overall word error rate without significantly hurting accuracy on abbreviated text.

7.3 Comparison with Past Work

Although there has been very little work on this task until quite recently, multiple studies have been done using the small 303-term dataset first used by Choudhury et al. (2007). For this reason, the two top performing systems (MT alone, and the average weight combination using MT and CRF) on this small data set. The Jazzy spell-checker is used as a baseline. Because this dataset has no context information the LM-only baseline is not feasible.

Table 7.14. Percentage of abbreviations correctly normalized in top-N sentence hypotheses after applying the dictionary heuristic.

		MT+R	**+cmu**	**+cmu***	**+asp**	**+jaz**
	Unigram	64.00	52.15	54.02	52.10	62.04
Top-1	Bigram	68.22	53.81	55.61	53.40	65.32
	Trigram	69.43	53.86	56.30	54.11	66.19
	Unigram	72.28	55.25	59.28	57.16	69.20
Top-3	Bigram	75.31	55.59	59.94	57.32	70.57
	Trigram	75.40	55.52	59.92	57.28	70.59
	Unigram	76.74	55.75	60.31	57.55	71.18
Top-10	Bigram	78.32	55.77	60.38	57.55	71.62
	Trigram	78.27	55.73	60.26	57.55	71.55
	Unigram	77.79	55.75	60.36	57.55	71.50
Found	Bigram	78.91	55.77	60.40	57.55	71.68
	Trigram	78.75	55.75	60.33	57.55	71.64

The CRF scores used in the average combination use all features and are rescored using a length model trained on all our data. The simple MT system is used rather than MT+R. The abbreviation models are combined with the unigram language model in order to form a prediction by giving the AM a weight of 0.1 and the LM weight is fixed at 1.

System comparisons are shown in Table 7.15. The results shown for other systems (except Jazzy) are those reported in their respective papers; we did not re-implement their systems. Both the MT and the averaging system perform comparably to the work of Liu et al. (2011) on this dataset. The averaging system has a slight advantage, but with only 303 items it is probably not significant. Although we outperform both Choudhury et al. (2007) and Cook and Stevenson (2009) in top-1, they outperform both the MT and averaging systems at top-10 and top-20. With better re-ranking, their system has the higher potential. It is thus important to obtain better coverage in future work.

7.4 Chapter Summary

Two extensions to the work discussed in previous chapters were presented. The first extension was to test various methods for combining the two abbreviation models presented in

Table 7.15. System comparison on the 303-term SMS test set provided by Choudhury et al. (2007).

	Top-1	**Top-3**	**Top-10**	**Top-20**
Jazzy (Idzelis, 2005)	49.86	53.13	54.78	55.44
Choudhury et al. (2007)	59.9	–	84.3	88.7
Cook and Stevenson (2009)	59.4	–	83.8	87.8
Liu et al. (2011)[1]	58.09	70.96	–	–
Liu et al. (2011)[2]	62.05	75.91	–	–
MT	60.39	74.58	75.57	75.57
Average (MT)	62.70	77.55	79.53	79.53

Chapters 5 and 6 to create a single system. Four methods were examined for cases where a word/abbreviation pair was present in both systems: a weighted average of the two system scores, taking the higher of the two scores, always preferring one of the two systems, and a case-specific method that uses knowledge of the abbreviation type. When using the original MT scores, the averaging method shows a slight increase in overall performance, but all of the combination methods are affected negatively by the addition of the pre-processing for repetitions. The second extension attempts to address the high false positive rates seen when using the AMs in previous chapters. A dictionary heuristic is created, whereby a token is not expanded if it appears in the dictionary being used under the assumption that it is already a correct word. For dictionaries were tested to find one able to decrease the false positive rate without significantly increasing the false negative rate. Extensions to this heuristic are planned for future work. Finally, the best performing systems from this dissertation were tested on a small standard dataset as a basis of comparison with past work. The methods presented herein perform well and should thus be considered state-of-the-art in this field.

CHAPTER 8
EFFECT OF NORMALIZATION ON TTS OUTPUT

The end goal of this normalization system is to improve the output of TTS systems when reading informal text. It is therefore important to know whether the current setup yields any performance in TTS performance. The Festival TTS system (Taylor et al., 1998) is used to generate three waveforms for each message:

1. The original informal message gathered from Twitter.
2. The normalized text for the message generated by the **MT+R+jaz** system described in Section 7.2.
3. The "standard" form of the message used in the SW test set.

The **MT+R+jaz** was used because it was the best system at the time the human tests were conducted. For consistency, the phoneme error statistics are also calculated using this same normalized version. We also used Festival's scripting language to obtain the phoneme sequence (without pauses) uttered by Festival for each of the three waveforms.

These waveforms and phoneme sequences are used in both quantitative and qualitative tests to determine the improvement gain our system obtains. As a quantitative experiment, we examine the phoneme error rates (PERs) of our system output and the original informal text to determine the improvement gained by our normalization scheme in that regard. We also perform human listening experiments (qualitative) to determine if the waveforms generated by our normalized output are easier for humans to understand.

8.1 Phoneme Error Rate

When evaluating normalization methods, the metric is usually word error rate (WER). For this metric, a hypothesized word is marked correct if and only if its lexicographic form exactly matches that of the correct standard form. The normalization methods are evaluated using this metric because a low WER is necessary for many NLP tasks to perform well in this domain, for instance search, summarization or sentiment analysis.

For TTS, however, this metric is too strict. When a TTS system pronounces "nite", it correctly sounds like the intended word "night" even though the lexicographic forms do not match. If the pronunciation is correct and understood by the listener, should the word be marked incorrect? To combat this problem, phoneme error rate (PER) has been used to evaluate TTS systems. The gold standard sentence is passed through the TTS system, and the phoneme sequence generated is compared to that generated when using the hypothesized normalized sentence. We show example phoneme sequences in Figure 8.1.

```
(w er k ih n ay hh ae v k aa f iy jh ih t aa z)
(w er k ih ng ay hh ae v k aa f iy jh ih t er z)
```

Figure 8.1. Example phoneme sequences for the message "workinnn i have coffee jittahz =/"

Phoneme edit distance (PED) is computed by using the edit distance algorithm while treating each phoneme as a word; that is, the phoneme "ax" is considered a substitution if the original phoneme were "ai", rather than a replacement of only the character "i" with "x". The PER of a phoneme sequence compared to a reference phoneme sequence is calculated as the PED divided by the number of phonemes in the reference sequence. Overall PER is thus calculated by averaging the message-level PER over the entire set of messages. This measure is important because it allows us to know whether the TTS system pronounces words correctly. The comparative PERs are listed in Table 8.1.

Although the PER across the entire data set is significantly decreased, another interesting statistic is the percentage of messages on which the system is able to decrease the error rate

Table 8.1. Phoneme error rates (PERs) for each message source.

	Original	**Normalized**
PER	7.7	4.7

compared to the original message. It is also important that the system does not frequently increase the error rate. The system is able to decrease the PER on 54.2% of the messages compared to the original PER while only increasing the PER on 21.1%. The increase rate is higher than desired but the ratio is in the appropriate direction.

8.2 Human Listening Tests

Two types of human listening tests were submitted to Amazon's Mechanical Turk[1]. Due to time and cost constraints, all 4500+ messages could not be submitted to the Turk workers. Instead, 100 messages were chosen for this purpose.

We used the PED from the original form of each message to order all of the messages. The 50 messages with the highest PED and 50 messages with an original PED of zero made up the test set of 100 messages for the perceptual evaluation. This setup was chosen to be sure the system can make a positive improvement on those messages that would be difficult to understand otherwise (high PED), but to also be sure the system does not make messages more difficult to understand when they are already pronounced correctly.

For the first perceptual test, workers were given only the audio file and asked to transcribe exactly what they heard. 43 annotators transcribed the 300 audio files over the course of six days. The character error rate (CER) of the transcription against the standard message was calculated and is shown in Table 8.2. Once again we use the Levenstein edit distance but here each character is treated as a word. CER was chosen rather than WER due to spelling errors on the part of the annotators; often the correct word was transcribed but misspelled

[1]`https://www.mturk.com/mturk/welcome`

Table 8.2. Character error rates (CERs) of transcribed audio files compared to the standardized text for each source.

	Original	**Normalized**	**Standard**
CER	21.79	16.85	15.09

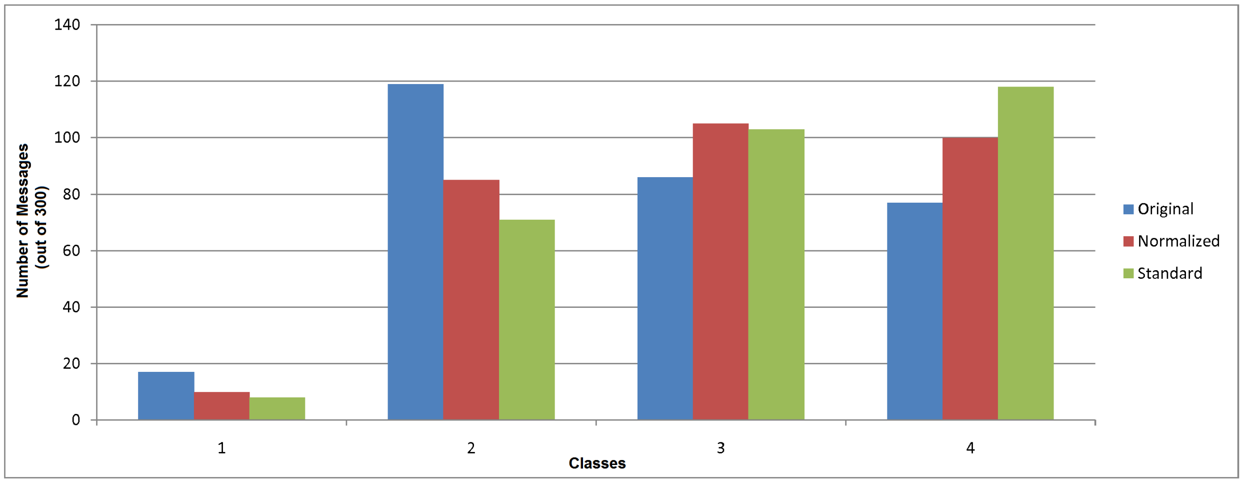


Figure 8.2. Histogram showing distribution of transcription difficulty rating by source.

and with CER the annotators are given partial credit for this word. The CER is lower for the normalized text than for the original text, showing that the speech produced using the normalized text was easier for the annotators to understand. There is still some room for improvement compared to the standard text, which was expected.

After transcribing the message, they were asked to rank the difficulty of the task on the following four point scale:

Class 1: This message was next-to-impossible to transcribe because everything is pronounced strangely.

Class 2: This message was difficult to transcribe. I had to listen to it multiple times and think hard about many words.

Table 8.3. Analysis of transcription difficulty ratings.

	Original	**Normalized**	**Standard**
Mean	2.73	2.98	3.10
Standard Dev.	0.90	0.86	0.85
Fleiss κ	0.14	0.12	0.13

Class 3: This message was sort of easy to transcribe. I only listened once but had to think fairly hard about a word or two.

Class 4: This message was very easy to transcribe, I only had to listen to it once and am very confident in my transcription.

A histogram of their rankings for each message source is shown in Figure 8.2. The distribution of rankings matches our expectations, with the original text having more ratings at the low end and the standard text having more ratings at the high end. As hoped, the normalized version has fewer messages in categories 1 and 2 than the original system.

The mean, standard deviation and Fleiss' κ (Kappa) of the chosen categories for each message source are shown in Table 8.3. Fleiss' κ is a statistical measure of the reliability of agreement any fixed number of raters when assigning categorical ratings to items. This is different from Cohen's κ, which only works when there are exactly two raters. Fleiss' κ only expects that the number of raters k is identical for each item, but does not assume that the *same* k raters categorize all items. Typically, Fleiss' κ is not used for ordered categories, but rather for unordered groupings such as categorizing a rock as "igneous", "metamorphic" or "sedimentary", and would therefore not generally be used for this rating task. Spearman's ρ would be used instead, which takes into consideration the relative position of the rankings rather than the actual categories, *i.e.*, (1, 2, 1, 3) is considered perfectly correlated with (2, 3, 2, 4). This behavior is not desired for this ranking task and the categories can be considered discrete.

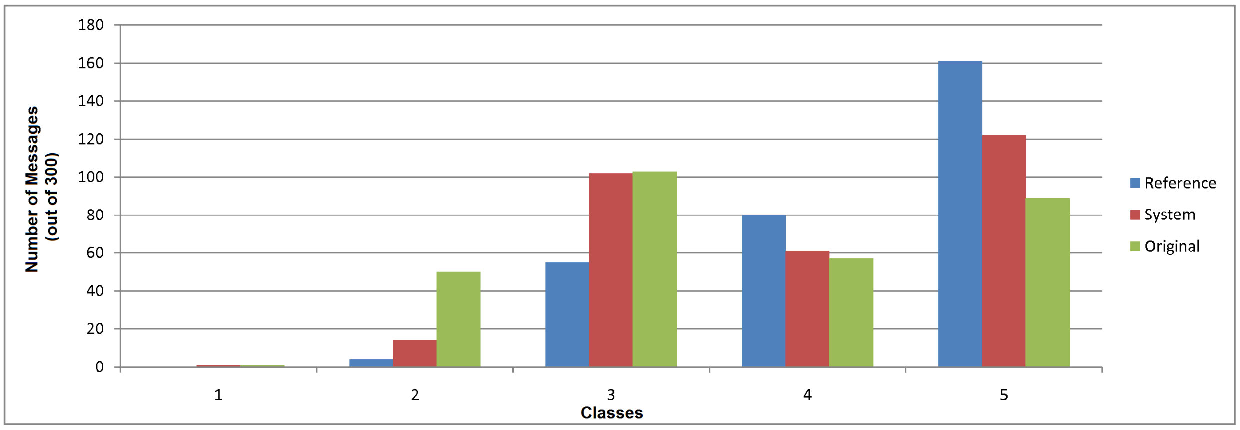


Figure 8.3. Histogram showing distribution of pronunciation rankings by source.

In the second perceptual test, workers are able to see the original form of the message while listening to the audio files. In this way, they first form an expectation of what they should hear in the audio file. The following definitions are listed:

“Pronounced Incorrectly” The wrong word is spoken (example: ”more” instead of ”mister”). The word is spelled instead of read, may be spelled properly (example: ”c-a-m-e” instead of ”came”) or incorrectly (example: ”l-u-v” instead of ”love”). Other gross mispronunciations (ex: ”suh uh uh uh uh uh oh” for ”so” or ”soooo”).

“Almost Correct” The word has a slightly wrong pronunciation, as what you might hear by a non-native English speaker or child learning to read. (example: ”hoose” for ”house” ... ”cot”/”cait” for ”cat”, etc).

They are then asked to rank on a five point scale how well the pronunciation in the audio file matches their expectations. The ranking scale was given as follows.

Class 1: Pronunciation seems to have nothing to do with the message.

Class 2: Many of the words are pronounced incorrectly.

Table 8.4. Analysis of annotators ratings for the second perceptual experiment.

	Original	**Normalized**	**Standard**
Mean	3.61	3.96	4.33
Standard Dev.	1.08	0.96	0.81
Fleiss κ w/5	0.17	0.17	0.14
Fleiss κ w/3	0.20	0.20	0.19

Class 3: Most words are correct or almost correct, but at least one is pronounced incorrectly.

Class 4: Pronunciation is almost correct but still easily understandable.

Class 5: Pronunciation exactly matches what is expected.

Nine unique annotators ranked the 300 audio files in just under 12 hours, with three unique annotators per file. A histogram of their rankings by source is shown in Figure 8.3. The histogram matches exactly what is expected Once again the mean, standard deviation and Fleiss' Kappa of the chosen categories for each message type are shown in Table 8.4. This seems to be a difficult task; the Fleiss' κ is very low for this task. For this reason, a second κ score is calculated by merging the two lowest categories and the two highest categories; for example, ratings of 1 and 2 are treated as a single category. Unfortunately, there is not much improvement when the categories are merged. Interestingly, the lowest agreement occurs on the standardized form. As expected, the system produced text falls between the original text and the reference text on both mean and standard deviation. There is still a lot of room for improvement in terms of the system pronunciation to bring the mean up to the level of the standard version.

8.3 Chapter Summary

The previous chapters provided results only in terms of the word error rate (WER) produced by the normalization system; however, the WER is not necessarily indicative of performance when the output is used in a text-to-speech (TTS) system. For this reason, this chapter provides both quantitative and qualitative evaluations of the effect on TTS performance when using normalized text. For the quantitative evaluation, phoneme error rate (PER) is used as the basis of comparison. Three audio files are generated for each message: the first is generated using the original text from Twitter, the second is generated on the normalized text, and the third uses the standard terms annotated in the corpus. The phoneme sequences for the original and normalized versions are compared to that of the “standard” version, and it is shown that normalizing the text yields a decrease in PER.

Two qualitative tests are performed as well, using Amazon’s Mechanical Turk. In the first test, annotators are given the audio files without any accompanying text and asked to transcribe what they hear. Their transcriptions are compared to the standard text at the character level and the transcriptions of the normalized audio falls between the original and “standard” in terms of character error rate. The transcribers were also asked to rate the difficulty of transcription, and again the difficulty was ranked between that of the original and standard. In the second test, annotators were given both the audio file and the original text from Twitter, and asked to rank the pronunciation of terms on a five-point scale. Once again, the normalization system falls between the original and “standard’ in terms of pronunciation accuracy. Overall, normalization of the informal text gives an increased performance on TTS output.

CHAPTER 9
CONCLUSIONS

As informal text becomes ubiquitous, techniques in all areas of NLP will need to adapt in order to use the vast amount of information found therein. In this dissertation, steps were presented toward an improved TTS system for informal text that can be utilized in screen readers for the visually impaired, to improve safety while driving or to make mobile phone applications more useful or entertaining. Improvements stem mainly from improved abbreviation expansion during the lexical normalization step. The normalization work presented in this dissertation produces state-of-the art results on our own data and on that from other researchers.

The work began with the collection and annotation of an appropriate corpus to use in the testing and training of a normalization system. Although some SMS messages were collected for this purpose, the annotated data used for experiments was obtained from the popular micro-blogging site at `www.twitter.com`. Of the four types of annotations that have been added to the data, the translation of chat abbreviations into their standard English form has been utilized the most. The annotated abbreviations and their standard forms were used to train many different abbreviation models to use in the noisy channel model for normalization. The noisy channel model is typically used in automatic speech recognition or statistical machine translation, but by using an abbreviation model rather than an acoustic or translation model it is applicable to this problem as well. A language model is also required for a noisy channel implementation; the language model used in this dissertation is trained on a corpus of Twitter status messages, which has the advantage of being in-domain for the task. Throughout the dissertation, the order of the language model is varied from 1 to 3 to judge the usefulness of higher order models. In general, it is shown that a bigram language model gives a large improvement over a simple unigram model, and that the improvement from bigram to trigram is smaller, but consistent. A bigram language model may be used without a sharp decrease in performance when memory or computational power is low.

The first type of abbreviation model presented focused on those abbreviations formed only by deleting characters from the original word. Deletion abbreviations are very common (the most common type of abbreviation in the annotated corpus) and TTS systems have greater difficulty in pronouncing them, so we believed that a model trained especially for deletions would be the most useful. Here, statistical classifiers (maximum entropy and conditional random fields) are trained to learn the probability of deleting a particular character from a word when forming an abbreviation. The features used by the classifiers were designed by hand based on the abbreviation patterns seen in the corpus. The posterior probability of deleting each character in the word was combined to form a score by which each possible abbreviation could be ranked. A look-up table was used to store the abbreviations and associates scores for each word in a dictionary. The look-up table is generated only once and would be shipped with the system. In this way, it is useful for a device with high memory but low CPU such as a mobile phone or a communication device for the physically impaired.

Unfortunately, this system did not perform as well as expected, even on the deletion abbreviations for which it was specifically designed. The performance of this method is bounded from above by the number of correct pairs in the look-up table, and unfortunately this value was very low. If this method is to be pursued in the future, much work will need to be done to improve the features used to train the classifiers. Preliminary work has been done to vary the dictionary used during look-up table generation. For the work done in Chapter 5, the CMU dictionary is pruned using `dictionary.com` because this is where the syllable information is gathered for feature extraction. Since then, the feature extraction program has been modified to treat words not listed at `dictionary.com` as a single syllable (though this is often not truly the case) and extract features accordingly. This significantly improves the coverage of the system, and will be investigated further in the future. In addition, improvement may be gained by initially generating far more abbreviations for each word and using an improved method for re-ranking the list. The largest source of error for this system, however, was the very high false positive rate when decoding entire messages.

Due to the drawbacks of the deletion-based method, the next approach pursued was an MT-based system that learns patterns on its own from the data without the need for hand crafted features. In this method, an MT system is trained at the character level to recognize common abbreviation patterns regardless of the word in which it occurs. Not only does a machine translation approach avoid human selected features, but it also has the advantage of addressing all types of abbreviations simultaneously, without the need for classification of the abbreviation type. The disadvantage of this approach is that the MT system is run on the abbreviations as they are seen in text, rather than on full English words as in the deletion-based system. For this reason, it is impossible to generate a look-up table and the device must be able to preprocess each message and run the MT software in order to make use of the system.

A large increase in performance in seen when using the MT system to generate the abbreviation scores. The upper bound on performance is increased drastically, and that performance holds at each step in the N-best list. With more annotated data and optimized parameters in the MT software, it may be able to increase the upper bound even further in future work. The increase in performance is seen even when tested solely on deletion abbreviations, meaning that the MT system is able to utilize some feature in the text that is not covered by those used in this work. A further gain in improvement occurs when repeated characters in the informal text are pre-processed to contain a maximum of two occurrences per repetition before training and testing.

Although the false positive rate is drastically decreased from that of the deletion-based methods, it is still higher than is desirable. For this reason, we created a heuristic that uses a dictionary to determine whether a given token should be expanded or not. This heuristic works reasonably well, provided that the dictionary is very carefully chosen. While the false positive rate was significantly decreased by incorporating any of the four dictionaries tested, there was a large increase in false negatives (abbreviations that should have been expanded but were left as-is) using three of the dictionaries due to the inclusion of acronyms and abbreviations included in the dictionary.

False negatives will be added regardless of the dictionary used; practically every dictionary will contain the word “no”, which is often used as an abbreviation for the word “know”, and “cat” which is used for the word “category”. Even the highest achieving dictionary showed lower performance at all levels of all tests (other than false positive rate) due to the increased number of false negatives. To combat this problem, effort will be made in the future to improve the dictionary heuristic. Rather than making the decision not to expand a word solely based on its inclusion in a dictionary, we can incorporate knowledge of the language model information as well; a word may be expanded if it is already in the dictionary, as long as the LM score is high enough.

In an effort to utilize the strengths of both systems, several methods of combining the deletion-based system with the MT system were evaluated. While most of the combinations led to a decrease in performance, taking the weighted average of the scores from each system shows promise. When combined with the dictionary heuristic, this is state-of-the-art.

Finally, we ask what effect these normalization improvements have on real TTS output. Both quantitative and qualitative tests show a significant improvement in the understandability of the audio produced by a common TTS system. Quantitatively, the normalization was able to lower the phoneme error rate significantly over the output produced by the original text. In the human listening experiments, transcription results improved in accuracy and the audio produced from the normalized messages was rated easier to understand and transcribe than that from the original informal text. Although there is some room for improvement in order to reach the results of the standardized text in both quantitative and qualitative performance, we have made great strides toward a workable system.

Aside from improving the normalization results, in future work one major goal is to improve the naturalness of the TTS output. The annotations already collected will be invaluable in making progress toward more natural sounding audio. TTS output on informal text often has strange phrasing due to the lack of punctuation in the original message. By using the annotated sentence boundaries, sentence boundary information can be restored to help with phrasing. But more importantly, prosodic information is gleaned far easier from informal than formal text. Informal text has a wealth of information available to it regarding the emotion and stress that the user intends to convey through the use of emoticons, capitalization and filler text. The emoticons used in a message can be used to determine the overall feel of the text; a smiling emoticon may indicate that a message should sound happy and have a rising tone, while a frowning face may lead to adding a downward pitch slope. In addition, sound effects such as laughter or scowls are often directly transcribed in informal text. Finally, capitalization can be used as knowledge of whether the user is yelling or intends to stress a word. Use of all of this information could lead to vastly improved TTS naturalness in the future.

To conclude, as language continues to evolve and informal text becomes more prevalent it will become necessary for NLP techniques to evolve as well. The normalization work presented here is only the first step toward that evolution to yield better results on downstream NLP tasks such as search, summarization, sentiment analysis and tasks that haven't yet been imagined.

APPENDIX

This appendix shows a compilation of all of the results throughout the dissertation.

Table A.1. Accuracy (%) of all systems using n-gram context tests.

	DEL Test Set				SW Test Set			
	Sys. Alone	**Unigram**	**Bigram**	**Trigram**	Sys. Alone	**Unigram**	**Bigram**	**Trigram**
LM Baseline	–	–	13.15	17.32	–	–	13.95	18.32
Jazzy	34.68	–	–	–	37.91	–	–	–
ME	36.76	43.00	45.12	45.87	–	–	–	–
CRF	37.27	45.48	46.81	47.44	–	–	-	-
MT	54.71	62.09	69.59	70.58	55.90	64.19	69.69	70.44
MT+R	–	–	-	-	56.64	65.91	71.32	71.93
Avg. (MT)	–	–	–	–	–	66.09	70.64	71.61
Avg. (MT+R)	–	–	–	–	–	65.09	71.73	72.84
High (MT)	–	–	–	–	–	64.09	70.44	71.39
High (MT+R)	–	–	–	–	–	65.68	71.95	72.93
Pref. CRF (MT)	–	–	–	–	–	63.23	70.26	71.32
Pref. CRF (MT+R)	–	–	–	–	–	65.54	71.98	72.86
Pref. MT (MT)	–	–	–	–	–	63.30	70.16	71.34
Pref. MT (MT+R)	–	–	–	–	–	65.09	71.77	72.72
Type Spec. (MT)	–	–	–	–	–	41.26	64.07	64.91
Type Spec. (MT+R)	–	–	–	–	–	41.92	65.59	66.38

Table A.2. Percentage of abbreviations correctly normalized in top-N sentence hypotheses for all systems.

	DEL Test Set			SW Test Set		
	Unigram	**Bigram**	**Trigram**	**Unigram**	**Bigram**	**Trigram**
Maxent	41.89	44.00	44.60	–	–	–
CRF	50.37	51.69	52.48	–	–	–
MT	59.81	64.87	66.34	62.63	66.56	67.90
MT+R	–	–	–	64.00	68.22	69.46
Avg. (MT)	–	–	–	64.34	67.90	69.40
Avg. (MT+R)	–	–	–	54.38	64.55	68.18
High (MT)	–	–	–	64.53	67.56	69.09
High (MT+R)	–	–	–	63.42	67.69	69.68
Pref. CRF (MT)	–	–	–	63.98	67.49	69.09
Pref. CRF (MT+R)	–	–	–	62.89	67.30	69.31
Pref. MT (MT)	–	–	–	59.26	65.42	68.01
Pref. MT (MT+R)	–	–	–	55.42	64.58	68.04
Type Spec. (MT)	–	–	–	52.60	54.84	55.50
Type Spec. (MT+R)	–	–	–	53.48	55.97	56.57
MT+R + cmu	–	–	–	52.15	53.81	53.86
MT+R + cmu*	–	–	–	54.02	55.61	56.30
MT+R + asp	–	–	–	52.10	53.40	54.11
MT+R + jaz	–	–	–	62.04	65.32	66.19

Table A.3. Message level top-1 accuracy (%) across all systems.

	DEL Test Set			SW Test Set		
	Unigram	**Bigram**	**Trigram**	**Unigram**	**Bigram**	**Trigram**
Maxent	2.90	15.58	21.51	–	–	–
CRF	0.05	1.54	4.68	–	–	–
MT	31.47	37.16	38.64	31.36	36.55	37.98
MT+R	–	–	–	32.07	37.82	39.08
Avg. (MT)	–	–	–	17.02	32.43	36.00
Avg. (MT+R)	–	–	–	2.21	18.29	29.30
High (MT)	–	–	–	13.42	32.15	35.92
High (MT+R)	–	–	–	8.87	28.07	34.77
Pref. CRF (MT)	–	–	–	8.59	26.65	32.19
Pref. CRF (MT+R)	–	–	–	4.07	20.39	29.50
Pref. MT (MT)	–	–	–	15.04	31.36	35.48
Pref. MT (MT+R)	–	–	–	8.43	26.13	33.74
Type Spec. (MT)	–	–	–	24.79	27.64	28.39
Type Spec. (MT+R)	–	–	–	25.90	29.58	30.29
MT+R + cmu	–	–	–	31.32	32.91	32.83
MT+R + cmu*	–	–	–	32.39	33.66	34.21
MT+R + asp	–	–	–	32.67	33.78	34.33
MT+R + jaz	–	–	–	35.36	38.21	38.89

Table A.4. WER for top-1 sentences across all systems.

	DEL Test Set			SW Test Set		
	Unigram	**Bigram**	**Trigram**	**Unigram**	**Bigram**	**Trigram**
Maxent	26.6	12.5	9.3	–	–	–
CRF	52.1	39.6	29.7	–	–	–
MT	7.6	6.6	6.4	7.9	7.0	6.8
MT+R	-	-	-	7.9	6.8	6.6
Avg. (MT)	–	–	–	13.0	8.0	7.4
Avg. (MT+R)	–	–	–	33.9	14.1	10.1
High (MT)	–	–	–	15.0	8.1	7.5
High (MT+R)	–	–	–	19.6	9.7	8.1
Pref. CRF (MT)	–	–	–	19.4	10.3	8.8
Pref. CRF (MT+R)	–	–	–	26.8	13.3	10.3
Pref. MT (MT)	–	–	–	13.8	8.3	7.6
Pref. MT (MT+R)	–	–	–	20.4	10.3	8.4
Type Spec. (MT)	–	–	–	9.1	8.4	8.2
Type Spec. (MT+R)	–	–	–	8.9	8.1	8.0
MT+R + cmu	–	–	–	7.6	7.4	7.3
MT+R + cmu*	–	–	–	7.5	7.3	7.2
MT+R + asp	–	–	–	7.4	7.2	7.1
MT+R + jaz	–	–	–	7.1	6.6	6.5

Table A.5. False positive rate (%) for top-1 sentences across all systems.

	DEL Test Set			SW Test Set		
	Unigram	**Bigram**	**Trigram**	**Unigram**	**Bigram**	**Trigram**
Maxent	23.33	7.95	4.49	–	–	–
CRF	52.33	38.64	27.82	–	–	–
MT	4.14	3.61	3.56	4.19	3.58	3.52
MT+R	4.07	3.50	3.44	–	–	–
Avg. (MT)	–	–	–	10.01	4.88	4.40
Avg. (MT+R)	–	–	–	32.42	11.34	7.34
High (MT)	–	–	–	12.34	5.00	4.43
High (MT+R)	–	–	–	17.49	6.77	5.28
Pref. CRF (MT)	–	–	–	17.26	7.44	5.92
Pref. CRF (MT+R)	–	–	–	25.53	10.87	7.68
Pref. MT (MT)	–	–	–	10.31	4.87	4.40
Pref. MT (MT+R)	–	–	–	17.28	7.04	5.42
Type Spec. (MT)	–	–	–	4.07	3.56	3.47
Type Spec. (MT+R)	–	–	–	4.07	3.50	3.43
MT+R + cmu	–	–	–	2.29	2.24	2.23
MT+R + cmu*	–	–	–	2.39	2.38	2.37
MT+R + asp	–	–	–	2.09	2.03	2.02
MT+R + jaz	–	–	–	3.03	2.93	2.92

REFERENCES

Acharyya, S., S. Negi, L. V. Subramaniam, and S. Roy (2009). Language independent unsupervised learning of short message service dialect. *International Journal on Document Analysis and Recognition 12*(3), 175–184.

Aw, A., M. Zhang, J. Xian, and J. Su (2006). A phrase-based statistical model for SMS text normalization. In *Proceedings of COLING/ACL*, Sydney, Australia, pp. 33–40.

Bangalore, S., V. Murdock, and G. Riccardi (2002). Bootstrapping bilingual data using consensus translation for a multilingual instant messaging system. In *Proceedings of COLING*, Taipei, Taiwan, pp. 1–7.

Barbosa, L. and J. Feng (2010). Robust sentiment detection on Twitter from biased and noisy data. In *Proceedings of COLING: Posters*, pp. 36–44.

Bartlett, S., G. Kondrak, and C. Cherry (2008). Automatic syllabification with structured SVMs for letter-to-phoneme conversion. In *Proceedings of NAACL-HLT/ACL*, Columbus, OH, pp. 568–576.

Beaufort, R., S. Roekhaut, L. Cougnon, and C. Fairon (2010). A hybrid rule/model-based finite-state framework for normalizing SMS messages. In *Proceedings of ACL*, Uppsala, Sweden, pp. 770–779.

Bento, C. and N. Gil (2005). Text input disambiguation supported on a hierarchical user model. In *Proceedings of sOc-EUSAI*, pp. 253–258.

Bollen, J., J. Mao, and A. Pepe (2009). Modeling public mood and emotion: Twitter sentiment and socio-economic phenomena. In *Proceedings of WWW*, pp. 450–453.

Casey, D. (2010). Official Google blog: Replay it: Google search across the Twitter archive. `http://googleblog.blogspot.com/2010/04/replay-it-google-search-across-twitter.html`.

Chakrabarti, D. and K. Punera (2011). Event summarization using tweets. In *Proceedings of AAAI ICWSM*, pp. 66–73.

Choudhury, M., R. Saraf, V. Jain, A. Mukherjee, S. Sarkar, and A. Basu (2007). Investigation and modeling of the structure of texting language. *International Journal of Document Analysis and Recognition 10*, 157–174.

Contractor, D., T. A. Faruquie, and L. V. Subramaniam (2010). Unsupervised cleansing of noisy text. In *Proceedings of COLING: Posters*, pp. 189–196.

Cook, P. and S. Stevenson (2009). An unsupervised model for text message normalization. In *Proceedings of NAACL-HLT Workshop on Computational Approaches to Linguistic Creativity*, Boulder, CO, pp. 71–78.

Cook, P. and S. Stevenson (2010). Automatically identifying the source words of lexical blends in English. *Computational Linguistics 36*(1), 129–149.

CTIA (2009). CTIA - Semi-Annual report, 2008 II. `http://www.ctia.org/media/press/body.cfm/prid/1811`.

CTIA (2010a). CTIA - Semi-Annual report, 2009 II. `http://www.ctia.org/media/press/body.cfm/prid/1936`.

CTIA (2010b). CTIA - Semi-Annual report, 2010 II. `http://www.ctia.org/media/press/body.cfm/prid/2021`.

Davidov, D., O. Tsur, and A. Rappoport (2010). Enhanced sentiment learning using Twitter hashtags and smileys. In *Proceedings of COLING: Posters*, pp. 241–249.

Diakopoulos, N. A. and D. A. Shamma (2010). Characterizing debate performance via aggregated Twitter sentiment. In *Proceedings of CHI*, pp. 1195–1198.

Fairon, C. and S. Paumier (2006). A translated corpus of 30,000 French SMS. In *Proceedings of LREC*, Genoa, Italy, pp. 351–354.

Gaudan, S., H. Kirsch, and D. Rebholz-Schuhmann (2005). Resolving abbreviations to their senses in Medline. *Bioinformatics 21*(18), 3658–3664.

Gong, J. (2008). *Improved text entry for mobile devices : Alternate keypad designs and novel predictive disambiguation methods.* Ph. D. thesis, Northeastern University. Available from `http://hdl.handle.net/2047/d10016090`.

Gross, D. (2010). Library of Congress to archive your tweets - CNN.com. `http://www.cnn.com/2010/TECH/04/14/library.congress.twitter/index.html`.

HaCohen-Kerner, Y., A. Kass, and A. Peretz (2008). Combined one sense disambiguation of abbreviations. In *Proceedings of ACL-HLT: Short Papers*, pp. 61–64.

Han, B. and T. Baldwin (2011). Lexical normalisation of short text messages: Makn sens a #twitter. In *Proceedings of ACL-HLT*, pp. 368–378.

Hasselgren, J., E. Montnemery, P. Nugues, and M. Svensson (2003). HMS: A predictive text entry method using bigrams. In *Proceedings of ACL*, pp. 43–49.

How, Y. and M.-Y. Kan (2005). Optimizing predictive text entry for short message service on mobile phones. In *Proceedings of HCII*. Available from `http://www.comp.nus.edu.sg/~kanmy/papers/hcii05.pdf`.

Idzelis, M. (2005). Jazzy: The Java open source spell checker.

Ju, Y. and T. Paek (2009). A voice search approach to replying to SMS messages in automobiles. In *Proceedings of the ISCA*, pp. 987–990.

Ju, Y. and T. Paek (2010). Using speech to reply to SMS messages while driving: An in-car simulator user study. In *Proceedings of ACL: Short Papers*, pp. 313–317.

Klarlund, N. and M. Riley (2003). Word n-grams for cluster keyboards. In *Proceedings of the EACL workshop on language modeling for text entry methods*, pp. 51–58.

Kober, H., E. Skepner, T. Jones, H. Gutowitz, and I. S. MacKenzie (2001). Linguistically optimized text entry on a mobile phone. Technical report, Eatoni Ergonomics, Inc. Available from `http://www.eatoni.com/research`.

Kobus, C., F. Yvon, and G. Damnati (2008). Normalizing SMS: Are two metaphors better than one? In *Proceedings of COLING*, Manchester, UK, pp. 441–448.

Koehn, P., H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, and E. Herbst (2007). Moses: Open source toolkit for statistical machine translation. In *Proceedings of ACL: Interactive Poster and Demonstration Sessions*, pp. 177–180.

Kukich, K. (1992). Technique for automatically correcting words in text. *ACM Computing Surveys 24*(4), 377–439.

Liu, F., F. Weng, B. Wang, and Y. Liu (2011). Insertion, deletion, or substitution? Normalizing text messages without pre-categorization nor supervision. In *Proceedings of ACL-HLT*, pp. 71–76.

MacKenzie, I. S., H. Kober, D. Smith, T. Jones, and E. Skepner (2001). Letterwise: Prefix-based disambiguation for mobile text input. In *Proceedings of the ACM Symposium on User Interface Software and Technology*, pp. 111–120.

NSC (2010, March). Understanding the distracted brain: Why driving using hands-free cell phones is risky behavior. Technical report, National Safety Council.

OConnor, B., M. Krieger, and D. Ahn (2010). Tweetmotif: Exploratory search and topic summarization for Twitter. In *Proceedings of ICWSM*, pp. 384–385.

Pak, A. and P. Paroubek (2010). Twitter as a corpus for sentiment analysis and opinion mining. In *Proceedings of LREC*, pp. 1320–1326.

Pakhomov, S. (2002). Semi-supervised maximum entropy based approach to acronym and abbreviation normalization in medical texts. In *Proceedings of COLING*, pp. 160–167.

Pakhomov, S., T. Pedersen, and C. G. Chute (2005). Abbreviation and acronym disambiguation in clinical discourse. In *Proceedings of the AMIA Annual Symposium*, pp. 589–593.

Pennell, D. and Y. Liu (2010). Normalization of text messages for text-to-speech. In *Proceedings of ICASSP*, Dallas, Texas, USA, pp. 4842–4845.

Pennell, D. and Y. Liu (2011a, November). A character-level machine translation approach for normalization of SMS abbreviations. In *Proceedings of IJCNLP (to appear)*, Chiang Mai, Thailand.

Pennell, D. and Y. Liu (2011b, May). Toward text message normalization: Modeling abbreviation generation. In *Proceedings of ICASSP*, Prague, Czech Republic, pp. 5364–5367.

Pennell, D. L. (2007, July). An improved method for text entry on cell phones. Master's thesis, The University of Texas at Dallas.

Petrovic, S., M. Osborne, and V. Lavrenko (2010). The Edinburgh Twitter corpus. In *Proceedings of the NAACL-HLT Workshop on Computational Linguistics in a World of Social Media*, Los Angeles, California, USA, pp. 25–26.

Pini, S., S. Han, and D. R. Wallace (2010). Text entry for mobile devices using ad-hoc abbreviation. In *Proceedings of AVI*, pp. 181–188.

Q., C. A. H. and A. H. H. (2009). A ngram-based statistical machine translation approach for text normalization on chat-speak style communications. In *Proceedings of CAW2.0*, Madrid, Spain, pp. 1–5.

Raghunathan, K. and S. Krawczyk (2009). CS224N: Investigating SMS text normalization using statistical machine translation. Technical report.

Ramage, D., S. Dumais, and D. Liebling (2010). Characterizing microblogs with topic models. In *Proceedings of AAAI*, pp. 130–137.

Sculley, D. and G. M. Wachman (2007). Relaxed online SVMs for spam filtering. In *Proceedings of ACM SIGIR*, Amsterdam, The Netherlands, pp. 415–422.

Sharifi, B., M. Hutton, and J. K. Kalita (2010a). Experiments in microblog summarization. In *Proceedings of SocialCom*, pp. 49–56.

Sharifi, B., M. Hutton, and J. K. Kalita (2010b). Summarizing microblogs automatically. In *Proceedings of NAACL-HLT*, pp. 685–688.

Singhal, A. (2009). Official Google blog: Relevance meets the real-time web. `http://googleblog.blogspot.com/2009/12/relevance-meets-real-time-web.html`.

Sproat, R., A. Black, S. Chen, S. Kumar, M. Ostendorf, and C. Richards (2001). Normalization of non-standard words. *Computer Speech and Language 15*(3), 287–333.

Stolcke, A. (2002). SRILM — an extensible language modeling toolkit. In *Proceedings of the ICSLP*, Volume 2, pp. 901–904.

Tanaka-ishii, K. (2007). Word-based predictive text entry using adaptive language models. *Natural Language Engineering 13*(1), 51–74.

Taylor, P., A. Black, and R. Caley (1998). The architecture of the Festival speech synthesis system. In *Proceedings of the Third ESCA Workshop in Speech Synthesis*, pp. 147–151.

Teevan, J., D. Ramage, and M. R. Morris (2011). #twittersearch: A comparison of microblog search and web search. In *Proceedings of ACM-WSDM*, pp. 35–44.

Telit Wireless Solutions (2006). *AT Commands Reference Guide* (Revision 1. ed.). Telit Wireless Solutions. Available from `http://www.telit.co.it/product.asp?productID=105`.

Toutanova, K. and R. C. Moore (2002). Pronunciation modeling for improved spelling correction. In *Proceedings of ACL*, Philadelphia, Pennsylvania, pp. 144–151.

Tumasjan, A., T. O. Sprenger, P. G. Sandner, and I. M. Welpe (2010). Predicting elections with Twitter: What 140 characters reveal about political sentiment. In *Proceedings of AAAI*, pp. 178–185.

van den Bosch, A. and T. Bogers (2008). Efficient context-sensitive word completion for mobile devices. In *Proceedings of MobileHCI*, pp. 465–470.

Veloso, A., W. Meira, T. Macambira, D. Guedes, and H. Almeida (2007). Automatic moderation of comments in a large on-line journalistic environment. In *Proceedings of AAAI*, pp. 1–8.

Whitelaw, C., B. Hutchinson, G. Y. Chung, and G. Ellis (2009). Using the web for language independent spellchecking and autocorrection. In *Proceedings of EMNLP*, Volume 2, pp. 890–899.

Willis, T., H. Pain, S. Trewin, and S. Clark (2002). Informing flexible abbreviation expansion for users with motor disabilities. In *Computers Helping People with Special Needs*, pp. 359–371.

Winfrey, O. (2010). Oprah Winfrey's no phone zone. `http://www.oprah.com/packages/no-phone-zone.html`.

Wise, A. (2010). Official Google blog: Google follow finder: Find some sweet tweeps. `http://googleblog.blogspot.com/2010/04/google-follow-finder-find-some-sweet.html`.

Wong, W., W. Liu, and M. Bennamoun (2006). Integrated scoring for spelling error correction, abbreviation expansion and case restoration in dirty text. In *Proceedings of AusDM*, Volume 61, Sydney, Australia, pp. 83–89.

Wu, W., Y. Ju, X. Li, and Y. Wang (2010). Paraphrase detection on SMS messages in automobiles. In *Proceedings of ICASSP*, Dallas, TX, USA, pp. 5326–5329.

Yang, D., Y. cheng Pan, and S. Furui (2009). Automatic Chinese abbreviation generation using conditional random field. In *Proceedings of NAACL-HLT*, Boulder, Colorado, pp. 273–276.

Yarowsky, D. (1993). One sense per collocation. In *Proceedings of the workshop on Human Language Technology*, pp. 266–271.

Yu, Z., Y. Tsuruoka, and J. Tsujii (2003). Automatic resolution of ambiguous abbreviations in biomedical texts using support vector machines and one sense per discourse hypothesis. In *Proceedings of the SIGIR*, Volume 3, pp. 57–62.

VITA

Deana Pennell was born and raised in the small town of Machias, Maine. She moved to the Dallas area in 2003 to attend the University of Texas at Dallas. After obtaining her Bachelor degree in Computer Science *Magna Cum Laude* in May of 2006, she decided to stay at UTD to pursue her graduate education. She received a Master of Science degree in August of 2007, which also focused on on improvements for mobile devices. During her time at UTD, Deana was a founding officer of the ECS Honor Society and a member and officer of Golden Key. She was supported during her graduate education by two prestigious fellowships: the Get-Doc Fellowship offered by UTD and the CHAMPS GK-12 Fellowship sponsored by the National Science Foundation. She also met and married her husband while completing her doctoral research, a fellow classmate and graduate student, Ryan Burchfield. After graduation, Deana is moving to the Washington, D.C. area with her husband and pets to pursue a research career in Natural Language Processing with the U.S. Department of Defense.