Data-driven Natural Language Generation: Making Machines Talk Like Humans Using Natural Corpora

Brian Langner

CMU-LTI-10-007

January 2010

Language Technologies Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Alan W Black – *Chair*
Carolyn P. Rosé
Alexander I. Rudnicky
Maxine Eskenazi
Julia Hirschberg, *Columbia University*

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

This research has been supported by NSF grants 0208835 and 0229715, the NSF IGERT Fellowship program, and a gift from Lycos, Inc.

Keywords: natural langauge generation, corpus-based generation, spoken language generation, speech synthesis, spoken dialog, speech, computers

For my wife Belinda,
always

Abstract

With the significant improvements that have been seen in speech applications, the long-held goal of building machines that can have humanlike conversations has begun to seem more reachable; there exist spoken dialog systems which can now be used effectively by much of the general public. Despite these improvements, however, applications are still frequently limited by their unnatural spoken language generation. This thesis discusses the problem of human-like spoken language generation: how to make machine-generated speech more like natural human speech. The scope of this problem is large, with issues in speech synthesis, natural language generation, and spoken dialog, among other areas. The work in this thesis is primarily focused on natural language generation, with some discussion of the issues related to speech synthesis and the intersection between synthesis and language generation. In particular, we discuss a method that uses signal modifications of the synthesized waveforms to emulate what humans do when trying to be understood better while speaking in noisy conditions.

One of the main differences between human- and machine-produced speech is in understandability; natural human speech is typically easier to understand. We describe a general framework, which we call uGloss, designed to improve the understandability of spoken generation of complex information. The uGloss framework employs a set of tactical generation strategies that attempt to take the expected capability of the human listener into account; by staying within those abilities the resulting spoken output is typically more easily understood. Though uGloss can improve understandability, it is not a complete solution to machine-generated human-like speech.

In many other fields, from speech recognition and synthesis, to parsing and understanding, using corpus-based statistical knowledge has led to improved systems. We propose a similar data-driven approach intended to improve language generation systems, specifically for speech and dialog applications. Our proposed approach – the MOUNTAIN language generation system – is a fully-automatic approach which uses machine translation techniques to generate novel examples from a natural corpus. This system is designed to be a domain-independent method for training a generator that can produce human-like language in speech applications, by translating the machine's internal representation of what it intends to say into a natural language surface form. Further, it is also designed to require no linguistics expertise for use, making it more appealing to non-experts than some other advanced language generation systems. We show the MOUNTAIN approach to be comparable in output quality as those advanced systems, without requiring their amount of manual intervention to design and build.

Finally, we discuss the difficulties involved in the evaluation of machine-generated language. Manual human evaluation of examples is expensive, both in time and personnel, and has potential problems when the human evaluators have only monetary motivations for evaluating, as opposed to real users of a system who care whether things are truly improved. Despite these problems, it is still the most commonly used method of evaluation. We investigate other automatic measures that can be used for evaluation, and determine how well they correlate to human judgments.

Acknowledgements

When thinking about the efforts that have gone into a PhD thesis, it is difficult to fully identify and properly thank all of the people whose contributions and assistance made the end result possible. In my case, certainly Alan W Black heads the list. As my advisor, his support, direction, and encouragement have helped me learn how to do good research, and how to properly talk about what I've done. It is rare that a conversation with Alan would not give me something new, from insight into a problem with my research, to how to improve my teaching skills. The countless hours he has spent with me have helped me make a significant improvement in the kind of academic I am; for that, I am forever in his debt. In many ways, I cannot imagine having a more ideal advisor. I would also like to thank my thesis committee, who have always been available to discuss ideas, and provide many helpful comments and suggestions to improve the things I was working on. Though he was not a part of my committee, I want to thank Stephan Vogel for the conversations we had in the past several months which helped clean up my research and make it stronger.

I also need to thank the good friends I have made since coming to CMU; they have been there with support and friendship, helping with research or just to relax. In particular, Antoine Raux, Arthur Toth, Satanjeev "Bano" Banerjee, Betty Cheng, John Kominek, Wen Wu, Jean Oh, Wilson Tam, Udhay Nallasamy, and Gopala Anumanchipalli were always there when I needed someone. I would wager some of them cheered the announcement of my defense, not just as my friends, but because it also meant I would stop bothering them to do my research studies! Many of them were able to help me prepare for my defense talk; John, in particular, made numerous insightful comments and suggestions after my practice talk that really made the talk stronger (and ultimately, more successful).

In addition to them, there are many other who deserve thanks for the multitude of discussions we have had over the years that helped steer my own research or gave me insight into another research area. Members of the Synthesis and Sphinx groups, and especially Tina Bennett, Kishore Prahallad, Ziad Al Bawab, Sourish Chaudhuri, David Huggins-Daines, Stan Jou, Ben Lambert, Matt Marge, Jahanzeb Sherwani, Stefanie Tomko, Bob Frederking, and Rich Stern have all contributed to my reaching this point.

Finally, the deepest thanks of all are for my wife Belinda. Her patience to let me work on “just one more thing” was incredible, and her support as my thesis neared completion was key to my ability to finish, while helping me stay grounded with the things that matter most. Even when deadlines meant extra stress and less time with me, she was always there to help with exactly what I needed. I could not have done it without her love and support.

Contents

Chapter 1

Introduction

People have long envisioned machines that can converse naturally with them. This, in fact, is one of the tests of artificial intelligence [Turing, 1950] – is a human able to tell if the entity they're talking to is another person or a machine? Though it is a large problem with many sub-areas, each of them has made steady progress such that sometimes machines are able to pass at least a limited version of the Turing Test, though we are still far from creating a machine capable of having an independent conversation about anything. This thesis is focused on one particular sub-area of this problem: human-like, machine-generated conversations. In other words, how can we build a machine that is able to talk like a person?

1.1 Natural Language Generation

Natural language generation is the task of producing natural language surface forms from a machine representation of the information. Broadly, there are two main steps in natural language generation, which can be classified as either *strategic* or *tactical* generation. *Strategic langauge generation* is a task where a decision is made about what meaning is going to be expressed, while *tactical language gen-*

eration is focused on determining which particular words and phrases are used to express that meaning in natural language. Though both are important to the task of language generation, in this thesis we discuss the problem of tactical generation only.

1.1.1 Spoken Language Generation

Natural language generation can also be subdivided based on the goals or use of the language being produced. *Spoken language generation* refers to machine-generated language that is intended to be spoken, as opposed to text-based generation that is designed to be read. Though many of the same problems and issues are applicable to both types of generation, spoken language generation has additional issues that need to be addressed due to the fact that it is often easier to understand sentences when the words can be seen (and re-read as necessary) rather than only heard.

Language Generation for Speech Synthesis

Frequently, spoken language generation is used in speech and language applications by feeding the output of the generator to a speech synthesizer, producing fully-machine generated spoken language. Introducing a speech synthesizer to the process creates additional problems with understanding; synthetic speech has not yet reached the point where it is as understandable as natural speech [Black and Tokuda, 2005, Bennett and Black, 2006, Fraser and King, 2007, Karaiskos et al., 2008, King and Karaiskos, 2009].

1.1.2 Understandability vs. Naturalness

Human-generated language is by definition natural, and also generally understandable. Machine-generated language typically has trouble matching human ability in both of those dimensions. Though they are related, and both dimensions are important in producing human-like language, they are somewhat orthogonal in that one can have understandable but unnatural language (or, natural but incomprehensible language); this is widely known in speech synthesis [Clark et al., 2007], and certainly applies as well to natural language generation. Thus, when discussing how “human-like” machine-generated language is, it is important not to assume that it is natural simply because it is understandable.

1.2 Evaluation

Evaluation is clearly an important part of any research work – how do we know that what we are doing is better? There is not much general agreement among researchers in natural language generation about a common method of evaluation, though there has been significant recent interest in this topic [Belz and Reiter, 2006, Paris et al., 2007, Reiter and Belz, 2009], along with organizing shared-task challenges [Belz et al., 2008] to more easily compare approaches. The most common evaluation method used in the field is human-based evaluation – including task-based evaluations where performance is judged using measures such as task completion, and rating-based evaluations where people are explicitly asked to judge the quality of examples. These sorts of evaluations give good results but can be very expensive to set up and perform, making rapid testing of systems and ideas difficult. Other similar fields, such as machine translation and text summarization, use automatic metrics more frequently than human-based evaluations; these are far less expensive to perform but may not be as consistent and reliable as actual humans at assessing quality. Until recently, there has not been much interest in the language generation community in using automatic measures for evaluation, though as corpus-based generation has become more common this has changed. It is important, though, that any automatic measure of quality be well-correlated to human perception of quality in order to be meaningful.

1.3 Thesis

Many advanced language generation systems are able to produce very natural, high-quality output, if not in the general case at least for some specific application or domain. Though the quality of these systems can be impressive, they are often very expensive to create, because they have a significant amount of manual annotation or handcrafting necessary to build them. Some systems make extensive use of linguistic information, requiring an expert to be heavily involved in its design and implementation, and possibly ongoing use. Though often this is not a problem, there are many potential applications where such an expert may not be available, or where a novice group is trying to build an application; in these cases those advanced systems would not be usable at all. In fact, many commonly used spoken dialog applications use the most basic types of language generation – templates or canned prompts – despite the penalty paid in quality, because they are easy to create and use. We are interested in designing a language generation system that can produce high-quality, human-like output, while requiring only minimal human intervention to build – or no intervention at all. Such a system would make more human-like language available to a greater number of speech systems.

One potential method of improving machine generated language is to design a system that takes human capabilities – such as auditory memory or cognitive load – into account in its generations. Restricting generations that are likely to be difficult for human listeners to understand, like very long utterances or complex abstract groupings, should result in more understandable machine output in general. This type of approach could be used in conjunction with a simple, template based system, improving the understandability of its output for complex information.

Another possibility is to determine how humans talk, including what words they use, and design a generation system that allows a machine to produce similar utterances. An obvious method for this is to use actual human-generated examples to train a generation system. Many areas in natural language processing and speech technology have been able to benefit from applying machine learning and statistical analysis, including automatic speech recognition, spoken dialog, and machine translation, among others. Surprisingly, natural language generation has not seen as much work using trainable methods, though this has recently been changing. Given the general successes in other areas, we believe it is likely such an approach can also result in improvements for natural language generation.

1.3.1 Thesis Statement

Improvements in spoken dialog systems have made it possible to create limited conversational machines, though truly natural conversations remain elusive. Frequently, one of the major factors which limits the perceived naturalness of machine conversations is the simplistic approach to natural language generation that is still typically being used, despite the existence of more advanced systems which are capable of highly natural output. The additional costs of using these superior systems seems to be the reason they are not frequently used in dialog applications.

In this thesis, we present two approaches designed to make machine-generated language more like human-generated language. The first approach is a framework for producing more understandable machine-spoken language; this framework uses a set of tactical generation strategies designed to take human capabilities into account, resulting in more understandable language when discussing complex information. Our second approach is a fully automatic, data-driven method primarily aimed at spoken dialog applications. This approach, which uses statistical machine translation techniques, can produce human-like language using only natural corpora. Though no further linguistic or semantic information is required for generating novel language with this approach, it should be possible to make use of it when available.

1.3.2 Contributions

The main contributions of this thesis are:

- A statistical data-driven method of tactical language generation that performs similarly to other advanced approaches. This is a fully-automatic method, unlike other approaches which require some level of human intervention in the training or building process.

- Demonstration that natural corpora can be used to generate novel natural language sentences that are as clear and understandable as human-written texts.
- A framework for producing more understandable language that will ultimately be spoken by a speech synthesizer.
- An investigation of evaluation methods – both automatic and human-based – for natural language generation output, and what these methods can tell us about output quality.

1.4 Outline

The remainder of this document is structured as follows:

Chapter 2 describes other work related to this thesis, including both work in similar areas in language generation, speech synthesis, and spoken dialog, as well as discussions of and solutions to other dimensions of this problem that are not directly part of this thesis.

Chapter 3 describes evaluation methods for natural language generation and their applicability to this work.

Chapter 4 describes the uGloss framework, a method designed to improve the understandability of machine-spoken natural language generation.

Chapter 5 describes the MOUNTAIN language generation system, a new machine translation based approach to natural language generation.

Chapter 6 summarizes the conclusions and contributions of this work.

Together, these chapters all address the problem of generation of human-like natural language from various different perspectives. The discussion in Chapter 2 examines the general dimensions of producing human-like spoken output, though much of the specific research discussed addresses a different part of the general problem than the core contributions of this thesis. The uGloss framework presented in Chapter 4 attempts to define a general strategy for understandable spoken language generation. It is best described as a component of tactical generation, not a full generation system by itself, and is most useful for presenting complex information (such as lists or groups of items). uGloss is intended to make machine-generated spoken language more human-like by increasing its understandability. Though it mostly succeeds at that task, it does not explicitly deal with other aspects of human-like output in its generations. In comparison, the MOUNTAIN system in Chapter 5 is a full tactical generation system, using statistical machine translation techniques and a corpus of natural responses to generate novel utterances from a suitable internal (machine) representation of the information to discuss. MOUNTAIN attempts to use the naturally-generated examples to get more human-like generation results. The expected MOUNTAIN use case is within a spoken dialog application, though the technique itself should be suitable for other generation tasks as well.

Because the general scope of the problem discussed in this thesis is sufficiently large to touch on several areas of research, it can attract interest from each of these related fields. However, some portions of this work will be of greater interest than others, depending on the perspective of the reader. For those coming from a speech synthesis perspective, parts of Chapter 2 along with the entirety of Chapter 4 are the most directly relevant. Researchers coming from a natural language generation background will find Chapter 2, Chapter 3, and Chapter 5 discuss issues and research in their field. Finally, spoken dialog researchers may be interested in the systems described in Chapters 4 and 5, since their intended use cases are strongly tied to spoken human-machine interaction.

Chapter 2

Background and Related Work

There are many ways of getting machines to speak more like humans do. Modifying the speech synthesizer – changing how the speech is said – is one approach, and can involve prosodic, spectral, and even stylistic changes to the words being said. Though it is clearly important to any solution to this problem, issues with *how* the words are spoken are beyond the scope of this thesis. Working on a different level, there are modifications in spoken language generation – changing or deciding what words to say – in order to produce more natural and understandable machine speech. The proposed approach of this thesis falls into this area of research.

This chapter covers work from both of these areas that is related to the problem of producing human-like machine-generated speech. Additionally, since our approach is a corpus-based method of natural language generation, we discuss significant work that has been done in that area as well.

2.1 Speech Synthesis: More Human-like Output Through Prosodic and Signal Changes

Natural human speech can be highly varied along many dimensions. In fact, people will deliberately change how they speak in different environments; in some cases this is a subconscious process. When they find they are having trouble being understood, people will change the manner in which they produce speech in a variety of ways to improve how understandable they are and how well they can communicate with others. When they want to call attention to some specific part of what they are saying, people use *emphasis* and other prosodic cues that result in different-sounding speech from how they “normally” sound.

Standard speech synthesis typically does not do any of these things, and is one of the main reasons even modern, high-quality synthetic speech can still be perceived as unnatural. There has been some work on improving this, which we describe in this section.

2.1.1 Prosody and Emphasis

Modern synthetic voices have progressed to the point where they can often be perceived as reasonably natural and understandable [Black, 2002]. Unit selection [Hunt and Black, 1996] and limited domain techniques [Black and Lenzo, 2000] can produce voices with highly natural prosody, because they use the natural prosody from the database of recordings. The limitation is that they are restricted to the prosody in the database, which may not be large enough to have all phonetic and prosodic contexts that are desired. Further explicit control over prosody without additional recordings is possible, but only by using signal processing methods that degrade the natural quality.

Work by Raux and Black [2003] took a corpus-based approach to F0 modeling. The approach used F0 units – instead of the more typical phonetic units – for a unit selection voice. This is similar to earlier work [Meron, 2001], which used groups of several syllables rather than single segments. By clustering units together based on various features, such as phoneme name, voicing, stress, part of speech, and neighboring units, and then using similar methods as segmental synthesis [Black and Taylor, 1997], the method is able to produce F0 contours which are then applied to the resulting synthesized waveform.

Results seemed to show that their data-driven approach was able to produce natural F0 models of specific aspects of prosody, which can be used to generate synthetic speech that is preferred over speech synthesized using rule-based F0 models. Additionally, when applied to other speaking styles, such as emphasized speech, this method produced more natural emphasis prosody than the standard rule-based model.

There is further work that has gone into producing expressive synthesis systems [Iida et al., 2003, Eide et al., 2004]; these have generally focused on designing and building corpora of the desired style of speech for the synthesizer. Such an approach is effective (presuming that a large enough database can be collected for the desired style), but is expensive since it requires a large set of recordings for each different style of speech to be generated. However, the results of evaluations show that synthesis systems that include models for emphasis and other speaking styles are improved over standard unit selection systems [Strom et al., 2007].

2.1.2 Speech in Noise

Speech in noise [Langner and Black, 2004b], or speech spoken in poor channel conditions, is a style resulting from delivering speech as if the listener had said, "I can't hear you, can you say that again." Speech in noise can be elicited from people by having them speak in a noisy room. In order to investigate this speaking style, we designed and recorded a database of natural speech in noise. It should be noted that volume is not the sole difference between speech in noise and plain speech. Speech in noise has different spectral qualities, different durations, and different prosody than plain speech, in addition to the power differences. Such speech has been referred to as *Lombard speech* [Lombard, 1911, Lane and Tranel, 1971], but we feel that term is inappropriate for this work, because the level of background noise being considered is relatively small. Furthermore, this work does not deal with more extreme examples of speech in noise, such as shouting, which is not directly comparable to normal speech.

Speech in noise can have different properties depending on the type of noise the speaker is dealing with. For example, speech produced during a rock concert will be different than speech produced near a loud white noise source, and both of those will be different than speech produced in a noisy restaurant. This work uses a recording of human conversational babble from a crowded cafeteria during peak lunch times as the noise source; thus, any conclusions from this work are likely limited to similar noise sources. The noise source was selected for several reasons, including its naturalness, people's familiarity with it, its spectral qualities, and the ease with which it could be obtained. Though our findings may be applicable in other circumstances, this has not yet been shown to be true, and so this work should not be taken as authoritative for all types of speech in noise. However, the speech collection and evaluation methods we have used are relevant for most, if not all, types of speech delivery styles worth investigating, and so this work provides a possible framework for working with speech beyond the specific style detailed here.

While we are interested in the understandability effects of natural speech in noise, our interest is motivated by our ability to get similar increases in understandability for synthetic speech. Despite vast improvements in the quality of speech synthesis in recent years, many people continue to find even the highest quality synthetic speech difficult to understand. If we could see understandability improvements for computer-generated speech like those of natural speech in noise, applications of synthesis such as spoken dialog systems would become significantly more usable in non-research environments. Some preliminary work [Langner and Black, 2005a] suggests that such improvements are possible, though not trivial, to obtain.

Speech In Noise for Speech Synthesis

Since concatenative speech synthesis as well as human perception of speech are significantly degraded if noise is present in the speech recordings, we must have a way of recording the *style* of speech in noise without contaminating our recordings with noise; what we require is recordings of *clean* speech in noise. Combined with the fact that human speakers are annoyingly adaptive, changing their speech production as they “get used” to the conditions they are in, it was necessary to design a recording method [Langner and Black, 2004a] that would account for them. Because of that adaptability, the noise should be randomly played or not played while a specific prompt is being recorded, so that the voice talent is unaware of the noise condition ahead of time.In order to isolate the desired speech from the noise source in the recordings, the voice talent should wear headphones during the recording process. The headphones deliver the noise source as well as the voice talent’s own speech; effectively, this simulates the acoustics of a noisy room to the voice talent without putting the noise in the same channel as their speech. Obviously, the noise source should be pre-recorded to simplify the logistics of playing it through headphones. It should be noted that the volume of the noise source can, and should be, adjusted to the desired level; in our work, it was adjusted to a level where it was noticeable to the voice talent without being uncomfortable.

After recording a database of speech in noise, it is possible to build a voice using that data just as with any other database, with a few caveats. As noted above, speech in noise has different spectral and prosodic qualities than plain speech. This often causes methods of F_0 extraction to give poor results, which in turn lowers the quality of the resulting synthesis. Additionally, it is possible to build a single voice that can produce plain speech or speech in noise, using marked-up text to determine the speaking style, since the recording process generates a full database of both styles. Such a voice is useful for circumstances where having multiple distinct voices either not practical or desireable, but both speaking styles are required.

Modifying Other Synthetic Voices

There are several possible methods to get existing voices to speak in noise. One novel approach is to use *style conversion*. Using techniques that were designed for voice conversion between a source and target speaker [Toda, 2003], we applied such techniques to learn a mapping between plain speech and speech that was generated in noise. This work uses a Gaussian Mixture Model transformation method [Stylianou et al., 1995], and works primarily with the spectral differences between the two styles, as well as some minimal pitch and durational differences. It is important to reiterate that those differences are *not* all that distinguish speech in noise from plain speech, and thus the transformation model is not going to be able to produce natural-quality speech in noise. Results from experiments using this approach were mixed [Langner and Black, 2005b], showing potential understandability improvements can be had if the resulting speech is close in quality to natural speech in noise.

2.2 Spoken Language Generation

There has been quite a bit of work in applying natural language generation to actual interactive systems [Reiter and Dale, 1997]. Many of these systems, at least initially, focused on methods for generating text, taking into account content, referring expression generation [Dale, 1992], discourse planning, and other basic tasks. This work made it possible to build practical systems, and allowed for more complex advancements.

Language generation designed for spoken output is not a new application. However, it has been clear for some time that *spoken* language generation is a different problem than *written* language generation [Prevost, 1996]. Other previous work [Pan and McKeown, 1996] has noted the importance of conciseness in the perceived understandability and usefulness of spoken output, particularly as compared to written output. Furthermore, complex sentence structure which is easily understood when written can be far more challenging when rendered as speech. Most of the work on spoken language generation has focused on intonation of the spoken output [Marsi, 2001], or on emotional variation [Fleischman and Hovy, 2002], which are clearly important for generating more natural-sounding speech. Other factors, some of which are shared with written language generation, including lexical choice, sentence structure, and conversation flow, can also play an important part in understandable spoken output.

As speech applications become more common, we feel it is important that they be usable for all segments of the population, not simply the young, educated technology specialists they are normally tested on now. To maximize the usefulness of these applications, they must work for everyone, rather than only an ideal subset. Thus, we want to demonstrate that it is possible to improve their capabilities, without making them more difficult for people to use.

2.2.1 Language Generation for Speech Synthesis

Often, a language generation system will be able to come up with multiple different, yet equally acceptable, ways of saying the same thing. For text generation, the particular words in a sentence are not necessarily important. However, for generated language that is ultimately going to be spoken by a machine using speech synthesis, word and phrase choices can play a large role in the quality of the spoken utterances. There have been several efforts to more tightly combine language generation and speech synthesis modules within a larger speech application, focusing on intonation and context [Davis and Hirschberg, 1988, Prevost and Steedman, 1994, Hitzeman et al., 1998, Pan et al., 2002]; effectively, this work attempts to use additional information from structure, syntax, and dialog context to improve prosodic predictions, and produce more natural speech.

The output quality is a particularly key issue for unit selection voices. When the output of a unit selection synthesizer is good, it can be difficult to distinguish from natural speech. The problem is that minor changes in words or phrasing can cause drastically different speech quality. When unit selection synthesis is bad, it is often *very* bad, making "problem utterances" especially glaring. Because of that, spoken language generation (using speech synthesis) should take the capabilities of the synthesizer and voice into account when generating utterances, so that low quality output is more likely to be avoided.

Nakatsu and White [2006] did exactly that, ranking the output of their generator using the predicted quality of the output's synthesis. Building on earlier work [Bulyko and Ostendorf, 2002], the approach has the generator produce multiple paraphrases of a single target surface form, and then estimate the quality of the synthesis output for each utterance; when the highest-ranked utterances are then spoken by the synthesizer, the result will more often be natural-sounding speech. Evaluation of this method showed clear improvements in the quality of the output by human judges. Later refinement by Boidin et al. [2009] produced similar results, but needed significantly less annotated data, instead using a larger feature set for training.

2.3 Corpus-based Natural Language Generation

Corpus- and statistically-based approaches to language generation have been around for some time now, though only recently have they been more commonly applied in applications such as spoken dialog. These approaches are appealing due to their ability to leverage machine learning as has been done for other NLP tasks, in order to provide better results than typical approaches. The Nitrogen generation system [Langkilde and Knight, 1998], for example, derived statistical information from a corpus in order to rank and select grammar-generated surface forms. The work by Ratnaparkhi [2000] describes a generation system that can be trained from an annotated corpus, learning to produce surface forms using only a semantic representation. Marciniak and Strube [2005] describe using a corpus annotated with semantic and grammatical information, which is then used as a linguistic knowledge base for generation. Amalgam [Corston-Oliver et al., 2002] uses a similar classfication approach as Nitrogen, but takes logical forms as input. Work by Zhong and Stent [2005] is more similar to our proposed approach, directly using unannotated data to learn surface realizations. Likewise, a similar approach as we describe has used statistical translation methods for summarization and headline generation [Banko et al., 2000], with some degree of success. Additionally, Sripada et al. [2003] generated weather forecasts using a parallel corpus of raw weather data and human-produced forecasts; following up this work, Belz [2005] compared N-gram selection and treebank based methods of statistical NLG in this domain. Other domains, such as machine-generated sportscasting [Chen and Mooney, 2008], have also successfully used corpus-based language generation.

Many of these approaches use significant amounts of linguistic knowledge (in some cases from annotations, and in some cases trained from data) in order to improve the natural language output they produce. In this thesis, we attempt to use a broadly similar method – automatically learning generation output from a corpus of examples – but without any explicit linguistic annotation of the corpus.

2.4 Summary

In this chapter, we discussed several areas working on improving the naturalness and understandability of machine-generated speech, including work in speech synthesis (that is, how the speech gets said) and spoken language generation (what things to actually say). Further, we examined work on optimally combining a language generator and a speech synthesizer, to improve the resulting spoken output. While all of these areas are related to the topic and goals of this thesis, they primarily focus on issues outside the scope of this work or are solving a different part of the larger problem of human-like machine-generated spoken language.

Finally, we discussed related work in corpus-based and data-driven approaches to natural language generation. The work of this thesis can be generally described as a similar approach in this regard; however, where many of these related methods use significant linguistic annotation of their corpus, we propose an approach that does not require any significant linguistics expertise to implement. Our proposed approach is detailed in Chapter 5.

Chapter 3

Evaluation of Language Generation Systems

How best to evaluate language generation systems has been an unresolved question for some time now. Dale and Mellish [1998] discussed many of the pertinent issues and dimensions, of which we are most concerned with output assessment and application potential – in particular, the notions of quality and accuracy, and possibly fluency as well. Aware of this lack of standard assessment, researchers in the field have been attempting to design and agree on a metric [Paris et al., 2007]. Shared task challenges, which are popular and useful in other fields (see the Blizzard Challenge [Black and Tokuda, 2005] for speech synthesis as an example), have also been suggested and introduced for NLG [Rus et al., 2007, Belz et al., 2008].

Directly comparing machine-generated language to human-generated language seems like an obvious approach; however, such comparisons were not routinely done as part of an evaluation until recently [Viethen and Dale, 2006]. When examining human-generated language, what is often found is that different people will describe the same things different ways; in fact, the same person might also describe the same thing differently at different times [Viethen and Dale, 2007, Langner and Black, 2007]. For any evaluation using human-generated answers to determine correctness, this sort of variation must be taken into account [Stent et al., 2005].

3.1 Methods and Metrics

Typically, methods of evaluating natural language generation output will fall into one of two general categories: those using automatic measures of some kind, and those using human judges. We discuss both in this section.

3.1.1 Automatic Measures

While natural language generation researchers are most concerned with how to evaluate machine-produced language, there are other areas of NLP which must deal with this problem as well, including machine translation and summarization. Both involve generating novel natural language; however, they differ from general NLG by having reasonably well-defined “correct” generations. Despite that difference, the automatic metrics used by both fields have been suggested for use in NLG as well [Popescu-Belis, 2007, Rus et al., 2007]. The advantage of automatic measures is that they are low cost and easy to run, allowing for the rapid comparison of different techniques and ideas, and straightforward demonstration of progress and improvements.

Simple string accuracy [Alshawi et al., 1998], analagous to speech recognition’s word error rate, is among the early attempts at finding a quantitative measure. Though it is simple to compute, it is fairly limited and has minimal correlation to perceptions of the actual quality or correctness of generated language. Further experiments aimed at finding an automatic metric examined tree-based accuracy metrics [Bangalore et al., 2000]. In corpus-based summarization, the ROUGE metric [Lin and Hovy, 2003] has been used as an automatic evaluation measure, though in general the field seems to prefer human evaluations to automatic metrics. However, as a metric, it is uncertain if ROUGE correlates to human ratings of summaries; there are claims both for [Dang, 2006] and against [Dorr et al., 2005].

An earlier approach from machine translation is the BLEU score [Papineni et al., 2002]. BLEU effectively works by measuring N-gram agreement between reference and test strings. It is a standard reported metric for MT systems, and is widely used primarily because it is claimed to be well-correlated to human judgements of translation quality and is straightforward to compute. Its success as a metric for MT has led to some language generation work reporting BLEU scores as well [Callaway, 2003], particularly corpus-based NLG [Belz, 2005, Belz and Kow, 2009].

It is unclear if BLEU is truly an appropriate metric to use for NLG evaluation, however. Recent analysis [Belz and Reiter, 2006, Reiter and Belz, 2009] suggests that BLEU (or similar metrics) can be reasonable to use in some applications, but not necessarily in the general case. Though it is correlated to human judgements for translation results, it may not be the best metric to use for evaluating NLG output because it tends to view output as a set of N-grams rather than sentences. This might work for machine translation output, but it seems possible that NLG quality may have other dependencies, particularly in cases where "correct" outputs can be highly varied. METEOR [Banerjee and Lavie, 2005] may be a useful measure for our needs, as an evaluation metric for machine translation that takes more sentence-level information into account than BLEU. Furthermore, METEOR is designed to work on a per-sentence basis, whereas BLEU is best used over entire documents. Since language generation, particularly within speech applications, is fundamentally about sentence-sized examples, METEOR seems like a potentially preferable metric.

3.1.2 Human-Based Evaluation

Natural language generation is most often evaluated using scores given by a human evaluator. Such evaluations are expensive in many ways, from design and setup time, to personnel costs, and even in how long it takes to complete. Even worse, this expense is recurring, since most evaluations are tied to a specific test and are difficult or impossible to reuse. Despite these drawbacks, the information about generation quality is difficult to find in other ways, justifying the expense. The main advantage to this style of evaluation is that it provides an actual human perspective and judgement, which is important for something that is designed to be used by people.

Evaluation typically involves giving people generated examples and having them provide scores indicating their quality, possibly in multiple dimensions (such as coherence, correctness, appropriateness, overall quality, etc.) [Lester and Porter, 1997]. These scores are usually presented as Likert-scale ratings [Likert, 1932]. Similar evaluation techniques can be found in other fields as well, such as mean opinion scores for speech synthesis quality evaluations. Other methods instead show two examples, with the evaluator asked to indicate which one is preferred. Frequently, actual natural examples are mixed in with the machine generated ones, both as a quality control check – it is unlikely that machine generated text will be better than natural examples – and to provide a ceiling for the rating scale.

The other type of human evaluation that is typically used is a task-based evaluation, where the generation output is measured by how well a person is able to accomplish something by using it. Work by Young [1999] is an early example of this kind of evaluation, where the generation of instructions was tested by measuring how well people were able to complete the task specified in the instructions. Task-based evaluation has been claimed to be a highly meaningful measure of NLG output [Reiter and Belz, 2009], though in many domains it can be prohibitively expensive to perform – even more so than other human-based evaulations. However, if a language generation system is part of an application with a built-in user base (such as a spoken dialog system), some of that expense can be mitigated, making this method more attractive. Particularly if the application has actual users, it can provide an excellent platform for evaluating NLP research; actual users are motivated to get correct information efficiently, which happens to correspond to the goals of a language generation system. Ultimately, since successful machine-generated language will be used in applications, it makes sense that it should be tested and tuned for such usage.

3.2 Why Real Users are Important

The two most common methods of getting human evaluators for speech and language tasks is by recruiting participants for a study, often by paying them, and by designing an application which people choose to use for their own reasons, not solely for the purpose of an evaluation study. These two groups frequently have noticeable differences.

For many paid subjects, since they simply do not care about the answers they are getting, they will accept obviously incorrect information as valid if they can “complete” the task faster by doing so. Designing an evaluation task for recruited subjects that does not suffer from this problem tends to be challenging, and there will always be questions about whether the users’ motivation impacts their behavior, and thus the resulting performance of the speech application. This is not a new observation, having been noted at least as early as the Communicator user studies [Rudnicky et al., 2000], and almost certainly prior to that as well; recent studies [Ai et al., 2007] explicitly compared recuited subjects and real users and found noticeable differences. We believe this shows a need to have subjects who are sufficiently motivated. People using a speech application because they want to – perhaps because it provides them information they want – have different motivation. Not only do they want the information from the system, they want the *right* information; in other words, they care if the application makes errors, whereas recruited users may or may not. Even if the recruited users do care, they may care for other reasons based on their motivation, which might cause them to be concerned with different issues of system performance than non-recruited users.

What seems clear from this is that evaluating speech applications can be very difficult to execute successfully, due to the issue of user motivation. One must either sufficiently motivate paid participants, perhaps by compensating on a number-of-successful-compeletions basis, or find users who are already motivated. The former introduces potential problems with the validity of results – are subjects doing better because of our improvements, or because we pay them? – while the latter makes it difficult to get direct feedback from the users about their interactions, which is another important criterion for evaluation.

3.2.1 Evaluation Challenges in a Real-World System

Knowing that our approach is providing improvements, and to what extent, is important. However, unlike speech recognition with word error rate, there is no simple, automatic, objective measure that can be used to evaluate this work in any situation. Within a single task or domain it may be possible to define some measures, though these will not always easily transfer across applications. Among these, task success and user satisfaction seem most likely to be useful measures, despite being possibly difficult to define and subjective. Indirect measures such as dialog length and conversational efficiency can possibly be used, since they are somewhat easier to obtain yet still should correlate with other measures of improvement.

3.2.2 Study with Real-World Users

Design

Since our work is designed to improve spoken dialog applications, the ideal platform for testing our approach is an established, mature dialog system with a large user base. Conveniently, such a platform is available: the Let's Go spoken dialog system [Raux et al., 2006], a publicly available dialog system that has received over 50,000 calls, provides users with bus schedule information over the telephone, using a mixed-initiative but generally system-directed dialog. This system uses Rosetta, a template-based slot-filling generation system [Oh and Rudnicky, 2000], originally designed for the CMU Communicator [Rudnicky et al., 2000] project. Once the system has provided a result, users can "navigate" through other nearby results to their query by asking for the next or previous bus. Such an interface is sufficient, but can be clunky to use at times, particularly if users wanted to know both the bus before and after the original answer. Since the system provides only one result at a time, users must go through each result in sequence, meaning to satisfy this query, the user must ask for the next, previous, and previous bus, hearing the first result twice. This may seem minor, but is actually reasonably common; about 35% of calls to the Let's Go system request at least one next or previous bus after getting an initial result.

For this study, we modified the system to discuss multiple results in a single answer. Our hope was that we were able to improve the efficiency of the dialog and the user's perception of how the dialog has gone, without negatively impacting performance. We had three conditions: the baseline system that was already in use, a multiple-result system that provided results with absolute times (i.e., 7:42 pm), and a multiple result system that used relative times (i.e., 15 minutes later). Generated answers in the multiple-result conditions use grouping to describe the bus numbers; however, in general we believe there were not enough different bus numbers in answers to practical queries for this to have a noticeable difference. Some prompts needed "wordsmithing" in order to be properly rendered by the speech synthesis for this system, but otherwise follow a small set of rules to generate the resulting prompt templates.

Our study ran on the Let's Go system for five weeks. For each call during our study, one of our three conditions (as described above) would be used: baseline (B), absolute times (A), or relative times (R). The conditions would be run in sequence; that is, a call would be in condition B, the following call in condition A, the following in condition R, and then the sequence would repeat.

Results

During part of the time our study was running, there was another study using the system as well. However, only one experimental condition would be used on a given call, so only about half of all calls during that time were used in our study. Furthermore, we excluded all calls that did not give at least one result to the user, since only calls that provided results would hear our modified generation. The number of calls in each condition is similar, but not identical, due to these effects. After accounting for these issues, we received 544 total calls to the system.

Condition	**# Calls**
Baseline	138
Absolute	163
Relative	157

Table 3.1: Number of calls in each experimental condition.

Additionally, we excluded from those 544 all calls that were longer than double the median length, or approximately 15% of the calls. We feel this is appropriate because these long calls typically have some unusual qualities or user behavior – such as poor channel conditions, high word error rates, or significant off-topic user conversation – that degrade dialog performance independently of the system's language generation. However, since these outlier calls occur randomly, not uniformly distributed across all conditions, they introduce a significant level of noise into the data. This leaves us with a total of 458 calls; the total in each condition is shown in Table 3.1.

Analyzing these calls, we see a conditional effect on dialog length. In the absolute condition, average number of turns decreased by 7% from the baseline condition; this is statistically significant to $p < 0.1$. However, the relative condition showed a smaller, non-significant decrease in turn length from the baseline. In terms of temporal length, neither experimental condition showed a significant difference from the baseline; in fact, the relative condition showed a small *increase* in dialog time. A full comparision of dialog length between conditions is shown in Figure 3.1.

We have also looked at the effect on user requests for next and previous buses in our different conditions. One would expect that the multiple result conditions would show reduced frequency of these user requests, since each answer from the system contains more information. Indeed, that is the case for the absolute condition, which shows a significant decrease in both requests for the next bus (47%) and previous bus (73%); these are significant to $p < 0.05$, and $p < 0.01$, respectively. However, the relative condition shows no significant difference on these user requests from the baseline, though the previous bus requests were 25%

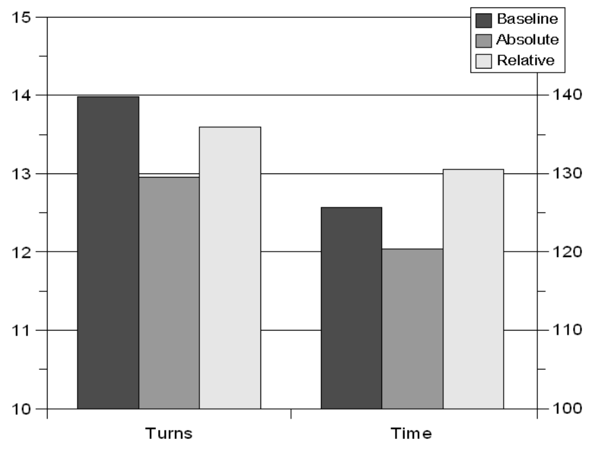


Figure 3.1: Comparison of number of turns and dialog length (in seconds) between experimental conditions.

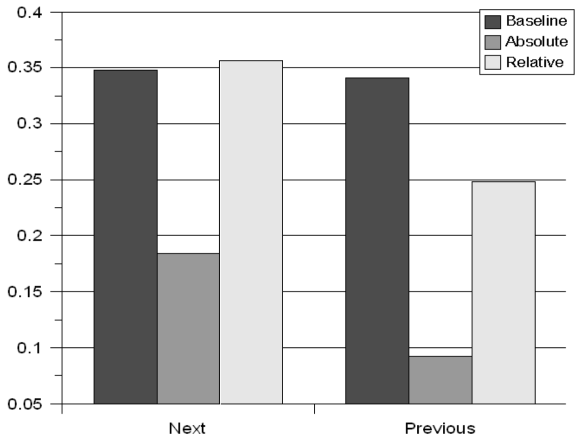


Figure 3.2: Comparison of next and previous requests per session between experimental conditions.

lower in this condition. These results are shown in Figure 3.2.

Additionally, we checked the effect of our different presentation on the frequency of user requests to repeat an answer. We found no noticeable increase of repeat requests in either of our two multiple result conditions; in fact, the baseline condition had the most examples of user repeat requests. However, we saw only 3 total examples out of all the calls we collected, so we do not feel this measure can actually tell us much about user experience in the different conditions.

3.2.3 Recruited Lab Participants

While we believe that using real users for evaluation gives more accurate results, the challenges of performing such an evaluation with most speech systems can make it less appealing than other methods. Though we do not expect to get the same results with recruited subjects, it would be helpful if those results were correlated to real user evaluations.

Design

We created a study with a similar task as the one described in Section 3.2.2; the goal was to have an equivalent system so that results between both studies would be easy to compare. As before, we had three conditions: the baseline system that was already in use, a multiple-result system that provided results with absolute times (i.e., 7:42 pm), and a multiple result system that used relative times (i.e., 15 minutes later).

Subjects were asked to go through six scenarios, two in each condition. Each scenario consisted of a departure point, a destination, and a range of possible departure times. In each case, subjects were told to use the dialog system to find out all the buses between the two points that left between the given times, and write down the answers they received. After each scenario, they were told to rate their impression of the answer they heard on a one to five scale. After completing all six scenarios, there was a short demographic questionnaire as well. Subjects were compensated with $5.

Results

We had a total of 20 subjects for this study. However, a significant number of these did not properly perform the task. In some cases, this was due to very poor speech recognition results; however, the majority were due to subjects either forgetting, or not bothering, to retrieve all buses for a scenario. In all, only 10 subjects sufficiently did the task such that their results were meaningful.

In both experimental conditions, requests for next and previous were about half as frequent as in the baseline system. This result is not overly surprising, given that more bus results are provided in these systems. The experimental conditions also both showed a negligible increase in number of requests to repeat an answer. The comparisons between the experimental conditions are shown in Figure 3.3.

We had hypothesized that the different answer generation types would have an impact on dialog length, an indirect measure of conversational efficiency. However, the results of this study do not show a significant difference, either in raw time, or number of user dialog turns. These results are shown in Figure 3.4. We had also hoped to show a user preference for one of the generation styles in this study based on the ratings subjects provided. Unfortunately, several participants rated

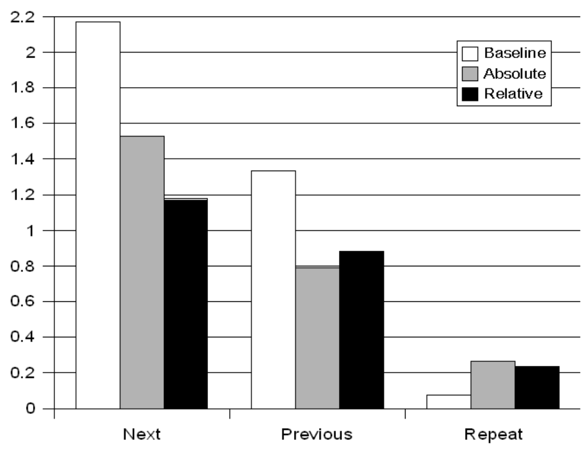


Figure 3.3: Comparison of average user next, previous, and repeat requests between experimental conditions.

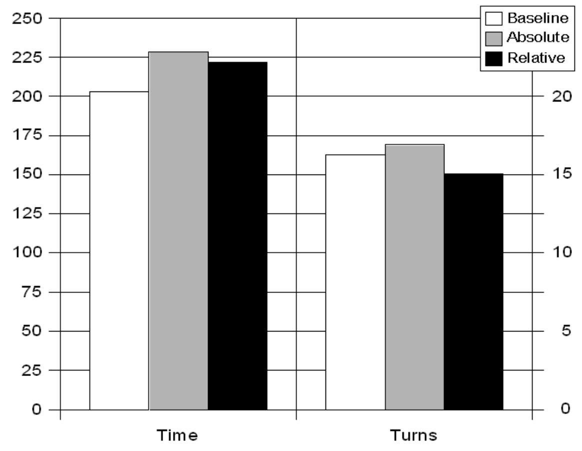


Figure 3.4: Comparison of dialog length (in seconds) and number of turns between experimental conditions.

their general experience with the system, rather than just its final answers. The only correlation for their ratings was to speech recognition performance in that particular dialog, rather than their impressions of the generation as we asked.

These results demonstrate a need to have sufficiently motivated subjects. For many paid subjects, since they simply do not care about the answers they are getting, they will accept clearly incorrect information as valid if they can "complete" the task faster by doing so. However, real users do not have this luxury, as they actually need to get the right answer, and so do not display this sort of behavior. Thus, to evaluate a system that will be used by people, it would seem necessary to have subjects that are similarly motivated, or at least that display the same type of behavior. This is not a particularly new revelation, having been noted and discussed in the Communicator project [Walker et al., 2001] as well.

3.3 Summary

Traditionally, language generation output has mostly used human-based evaluation. Though expensive, this is a justifiable standard metric. However, particularly for task-based evalauations, it is important for the motivation of the human judges to be taken into account. Evaluators who are unconcerned with the output quality (or correctness, or other dimensions) are simply not going to provide as accurate or as useful an evaluation as motivated users of an application. The challenge is to extract useful information from real users about how NLG changes impact task success without disrupting or noticeably altering their interaction.

Because of the expense of human-based evaluations, the search for a cheaper yet still effective evaluation technique has led towards automatic measures. After seeing the success they have had in other fields, there has been increasing interest in using this type of evaluation for language generation as well. Because of the simplicity of using automatic measures, it is likely that they will be popular for natural language generation, as they have been in related areas like machine translation and summarization. However, in these areas, the automatic measures being used have been shown to correlate to human evaluation, whereas this has not yet been conclusively shown for natural language generation. It is important, then, to see if and how well these measures correlate to human judgements of NLG quality. Some initial investigations have been done [Stent et al., 2005, Reiter and Belz, 2009, Belz and Kow, 2009], but we believe further research is warranted.

Chapter 4

Improving Spoken Language Generation: The uGloss Framework

Many current speech applications typically take a fairly naïve approach to presenting anything beyond basic information – usually simple, "read a long list" utterances – if they attempt to do so at all. Besides producing spoken output that is hard to understand (and thus, less useful for the human listeners), this kind of approach fails to take advantage of the underlying structure (and in some cases, context) that is typically characteristic of complex information. What is seen more often, though, are systems that are limited to avoid the problems of complex spoken presentation. Either of these approaches results in a speech system which is perceived as clunky and unnatural, and is less useful than it would otherwise be.

Systems that try to extend a naïve approach to generation beyond basic information presentation tend to be difficult for most people to understand; the machine speech is so unlike what a human would produce that a human listener is overwhelmed trying to decipher it. If the speech application is being used in an environment more challenging for understandability than a quiet office – as a real application likely would be – then the machine-generated output has yet another disadvantage compared to natural speech.

This chapter describes the uGloss framework, a general-purpose algorithm designed to influence the production of understandable machine-spoken output.

4.1 Preliminary Work

The work described in the following sections investigates several different strategies designed to make use of structural (and other) information not explicitly contained in the text in order to produce speech that is more understandable. Several different approaches were investigated, including varying the amount of information presented at a given time, limiting the length of time for generated utterances, and altering the semantic presentation of the raw text. All of these approaches are based on the idea of tailoring the generated language to account for the limitations of the human listener, since that is likely to be easier and more successful than finding a general method of improving the capabilities of those human listeners.

General tactics, such as limiting the length of time of spoken utterances – or in the case of lists, the number of items presented in the list – are often used, since longer utterances in general will be harder to understand fully. There are numerous methods which could achieve this, though all have their challenges or drawbacks. The simplest methods of implementing this approach, such as a "fast-talking" synthesizer or producing only severely abbreviated, disfluent language are not sufficient; at best these methods will be perceived as strange and unnatural, and at worst they will actively hamper human understanding. Fluent or mostly fluent utterances, despite being longer, are often more understandable because they are more like what the user is expecting to hear, and because conversational "filler" phrases can draw attention to the important information and then be ignored, thus not taking up the human's relatively short auditory memory.

Asking follow-up questions to a user to add constraints is a common approach, especially in domains like flight-reservation. This often is a good choice, though it tends to be somewhat domain-dependent. If the domain is very open-ended or has lots of possibilities even when constrained, this is not a viable method. Further, asking questions of the user limits this solution to interactive applications, and while it can be successful there, we are interested in a more general solution that can be used in many different text-to-speech applications. It is also possible to only give part of information to the user, what we refer to as *subsetting*, indicate that there is more information, and allow the user to request it if desired. This has the same interactivity requirement, as well as placing an increased burden on the user, making it less attractive. *Summarizing* long content, rather than providing all of it, is a related possibility, given a functional summarization system.

Yet another option is to group related items, preferably in an automatic fashion, to take advantage of chunking [Miller, 1956]. However, this raises the question of what appropriate groups are. It is easy to come up with valid but convoluted groups that an automatic process could identify that would be utterly inappropriate to present to a human; one such example could be "You can take any bus that comes by whose number is prime and letter comes before F." Such an answer from a bus schedule service, concise and accurate though it may be, would likely create more confused users than satisfied users. It should be possible, though, to identify and create reasonable groups, and so this seems like a good first approach to the problem.

4.1.1 How Much Information to Present

Presenting lists of information understandably is difficult at best for many speech applications, even when using natural recorded speech. The standard simple approaches fail quite badly when the list of items becomes longer, and when using synthetic speech these problems are even more pronounced. Improving spoken list presentation would be useful for a large class of speech-based information-giving systems.

Experimental Results: Initial User Study

We performed two studies to evaluate list presentation. In the first, subjects used headphones to listen to ten sentences with three places that described a trip that

On weekends, the H11, H12, J12, and X11 stop at the beach on the way to the train station from the hospital.

On weekdays, any of the 20s or 30s will go between the high school and the zoo, turning at the library.

Figure 4.1: Example sentences from this study.

a list of (fictional) bus numbers could take. The audio for the sentences was produced with the Festival speech synthesis system [Black et al., 1998], using the standard diphone voice `kal_diphone`. Each sentence was heard individually, and could be listened to only once. Figure 4.1 shows examples of sentences that were used. The bus numbers in the sentences are unlike those used by the local transit system. This was done deliberately, to remove any possible domain familiarity effects for subjects.

Subjects were told to draw an arrow on a map corresponding to the places in the sentence. After drawing the arrow, they needed to identify the bus numbers from the sentence out of a set of 32 possible numbers. The map of locations presented to subjects for each sentence is shown in Figure 4.2. There were 16 locations shown on the map, any of which could be present in a sentence. To prevent confusion as to

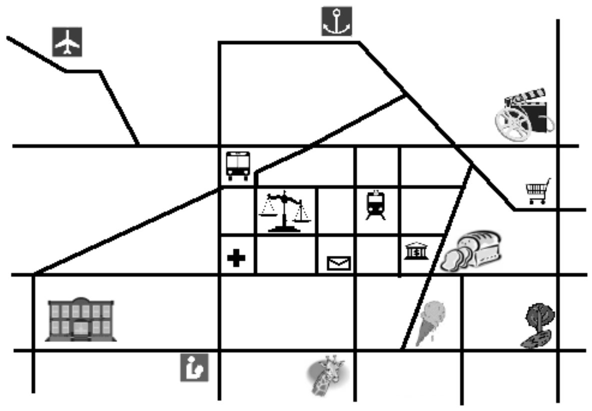


Figure 4.2: Map used in this study. Subjects were directed to draw an arrow on this map showing the path described in the sentence they heard.

List:	“H11, H12, H21, ...”
Groups:	“All of the 25s”
Exceptions:	“Any of the 40s except the 48”

Figure 4.3: Examples of presentation types in this study.

the iconic representation of places, subjects were provided with a key to reference as needed.

Of the 32 possible bus numbers, sentences presented anywhere from 2 to 28 numbers for subjects to remember. Bus numbers were presented randomly as simple lists, with small groups, or with complex groups using exceptions. All subjects heard the same semantic content in the same order; the differences were solely in the way the bus numbers were presented. Figure 4.3 shows examples of these presentation types.

Subjects in this study were primarily undergraduate and graduate students from local universities. Most were not familiar with current speech technologies. They were compensated with \$5 once they finished the task.

We evaluated performance using two different measures: “Bus” Error Rate and full correctness. What we are referring to as Bus Error Rate (or simply Error Rate hereafter) is no more than applying the familiar Word Error Rate metric to the buses selected by participants for each sentence, arranged in ascending alphanumeric order. The formula is shown here:

$$ErrorRate = \frac{ins + del + sub}{total\ reference\ bus\ numbers}$$

Full correctness is the situation where subjects identified all of the buses correctly without including any extra buses – essentially 0% Bus Error Rate. These measures are obviously related, but as our goal is to improve overall understanding, full correctness is more useful and appropriate for evaluating performance.

In general, what we discovered is that most people found this task quite difficult, though there was a large variance in performance. There were 22 participants in this study, with an average Error Rate of 51%. Separating out each condition, we

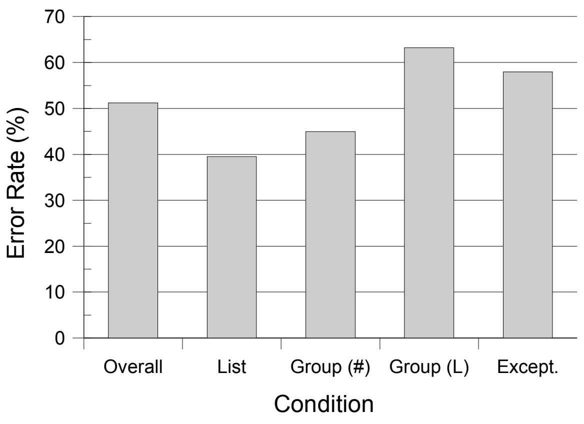


Figure 4.4: Overall and per-condition (lists, group-by-number, group-by-letter, exceptions) Bus Error Rate results.

found that simple lists had 40% error rate; groups by number (all the 40s), 45%; groups by letter (all the L's), 63%; and exceptions, 58%. These values are shown in Figure 4.4.

Again, however, since we are most interested in the ability to have a list of presented items remembered completely, what we would really like to know are the full correctness scores. Bus Error Rate can be deceptively good with a large list – missing one or two items from a set of ten shows a good error rate – while a person still has understandability problems. These results also show that this was a difficult task for people. With the highest possible score on this task being 10, participant scores ranged from 0 to 4 with an average of 1.63, which is quite poor. 25% of the simple list condition examples were fully correct, compared to 11% from the number groups, 19% from the letter groups, and 14% from the exceptions; this can be seen in Figure 4.5. Note that the better error rate for grouping by number compared to grouping by letter masks the lower overall score for that strategy.

This result runs counter to our expectations, namely that presenting a large list of items as a smaller number of chunks of items would increase the number of buses people could remember. This could be for several reasons. First, the results from this study are not statistically significant, most likely due to having too insufficient subjects. Also, though overall the simple list result seems better, this hides the fact that the two best results came from the grouped cases – one with four total bus numbers, and one with eight. It should be noted that in both of these grouped cases that a simple, concise group was able to fully describe the set of bus numbers (i.e.: "all the 25s").

In contrast, the simple list condition did well only when the list was limited to two or three items; beyond that performance degraded drastically. Though this study was not designed properly to show this clearly, from looking at the scores it seems that most people seemed able to remember about 3$\pm$1 items, which fits with the general guidelines LeCompte [2000] suggests.

Additionally, approximately one-third of the subjects were non-native speakers of English. Natives, unsurprisingly, generally outperformed non-natives at this task, by about 20 to 30% error rate. Splitting the natives and non-natives and evaluating them separately, which the numbers (and intuition) suggest would be reasonable to do only further highlights the lack of subjects.

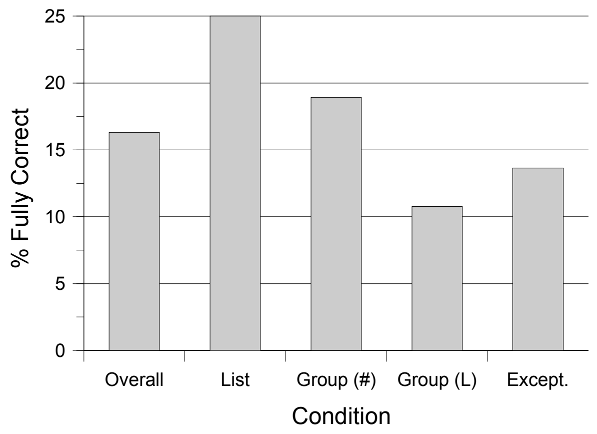


Figure 4.5: Overall and per-condition (lists, group-by-number, group-by-letter, exceptions) full correctness scores.

Experimental Results: Follow-Up Study

As we felt that the previous study was more difficult than intended, we made some minor modifications designed to make the task somewhat easier. First, we changed the format of the bus numbers to be in the same style as those used by the local transit system, feeling that a more familiar number scheme would be easier for people to deal with. However, we continued to select number-letter combinations that are not used by the real buses, to ensure that domain knowledge did not impact performance. The possible bus numbers increased from 32 to 36, but no more than six items were presented in any sentence. Some sentences contained simple groups, such as "all the 37s"; when groups were present in the list, they appeared only at the beginning, end, or both ends of the list. Based on the defined domain of bus numbers, all groups contained exactly three items. This means the maximum number of buses that could be presented in a sentence was ten: six items in the list, including groups of three on each end. Subjects again heard ten sentences, two each with list lengths between two and six items. List items were presented in random order, except for the restriction on placement of groups within the list. Subjects heard half of the 20 possible sentences, which were presented in random order.

Further, only two places were now used to describe the trips made by the buses, and the size of the map was scaled back, reducing the complexity of the task. Figures 4.6 and 4.7 show example sentences and updated map, respectively. The number of places on the map was reduced to 6 from 16. Subjects were still asked to draw an arrow indicating the path given in the sentence, and then to identify the bus numbers from the sentence.

The speech for this study was produced by a high quality commercial synthe-

On weekdays, the 44N and 47L go from the hospital to the airport.

On weekends, all the 35s, the 33J, 32H, 32L, and all the 39s go from the train station to the post office.

Figure 4.6: Example sentences from the follow-up study.

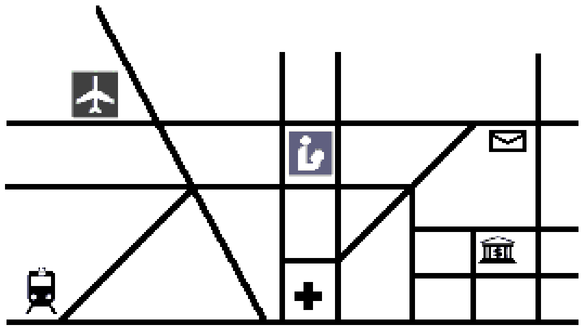


Figure 4.7: Simpler map used in the follow-up study. Subjects were directed to draw an arrow on this map showing the path described in the sentence they heard.

sizer. We felt the understandability gains made by changing to higher quality synthesis, compared to the relatively low quality of diphone speech, would decrease the difficulty of the task and allow us to see if synthesis quality had a significant impact on the poor performance of the participants in the earlier study.

Despite our efforts to make the task easier, most people continued to find it exceedingly difficult, though the variance in performance was somewhat reduced. Of the 34 participants, the average Error Rate was 50%, ranging from 19% on the low end to 75% at the high end. When separating the results by condition, we found that having groups anywhere in the list improved error rates, and not surprisingly, two groups had the lowest error rate, at 40%. These values, along with the standard deviations, are shown in Figure 4.8. It is interesting to note that with only one group present, placing that group at the end of the list resulted in a lower error rate than when the group was at the start of the list, suggesting the recency effect is stronger, at least for this task. Including a group anywhere in the list showed a 48% error rate, compared to 65% with no groups present.

Again, however, while useful, error rate is not precisely what we are most interested in – the full correctness scores are. These score still show this task was difficult; however, it does seem to highlight a border point where the difficulty changes drastically. With the highest possible score on this task once again being

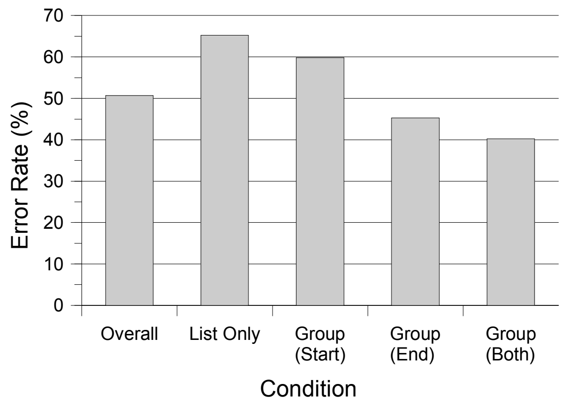


Figure 4.8: Overall and per-condition Bus Error Rate results.

10, participant scores ranged from 0 to 4 with an average of 1.65, which is similar to the previous study and still quite poor. However, as with the error rates, including groups increases performance at this task, with multiple groups showing full correctness 24% of the time. These scores are shown in Figure 4.9.

For lists with only two items, subjects could correctly identify all bus numbers (which ranged from two to six) 46% of the time. When the list contained three items, subjects were still correct 26% of the time. Once the list reached five items,

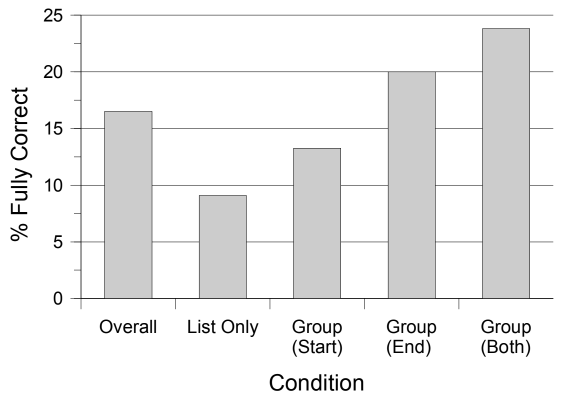


Figure 4.9: Overall and per-condition full correctness scores.

people were only able to be fully correct 4.3% of the time. If we ignore the results from the first two sentences, with the idea that subjects were getting familiar with what the task required, performance increases; two item lists are correctly identified 71% of the time, and three items 38%. The best case result for five items increased as well, but remained quite low at 5.8%. These results, as well as those for the longer list conditions, can be seen in Figures 4.10 and 4.11. Note that lists with four items showed worse (though not significantly worse) performance than lists with five items. The difference in score between the low-item conditions (two or three items) and the high-item conditions (four or more) is statistically significant.

Discussion

The results from these studies seem to confirm our initial intuitions, as well as the prior research results from cognitive psychology: simple, logical groupings will allow people to remember more individual items from a list, but that more complex groups tend to be of much less benefit. Accordingly, sentences making use of groups showed an increase in understandability compared to those that did not. Most people have extreme difficulty with lists once they grow beyond three items. Whether these items are groups or individual pieces of information does not seem to be important.

Both of these studies highlight the difficulties in providing spoken lists understandably. Even in the best case observed in this work, people still failed to understand what was presented to them nearly 30% of the time. Considering that example was with two items in a list – the minimal amount of information that can legitimately be called a list – *and* the subjects were ideal – young, educated natives with no hearing or learning difficulties – this brings into question whether less ideal populations like non-native or elderly listeners would have much hope of understanding spoken lists from a speech system at all at this point. Since speech systems, especially spoken dialog systems, are increasingly being used by people in those challenging groups, these results show the importance of finding a way to

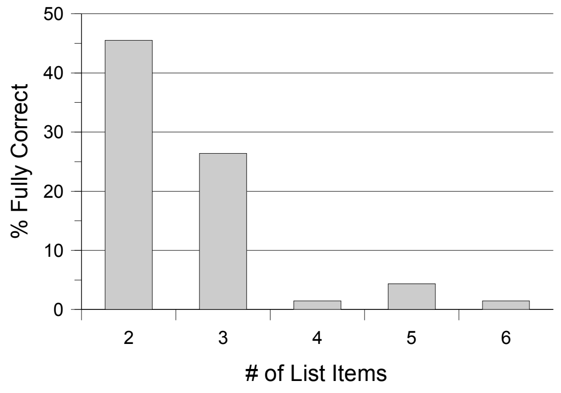


Figure 4.10: Per-item full correctness scores. These scores include all data.

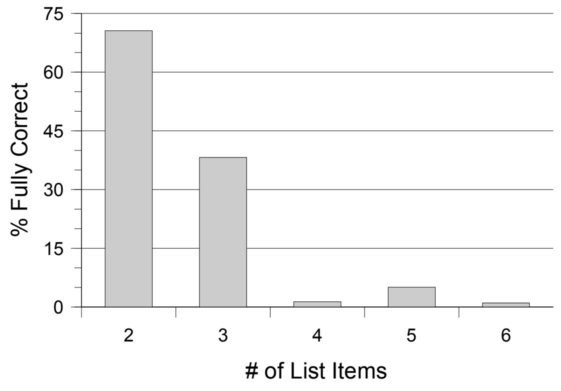


Figure 4.11: Per-item full correctness scores. These scores exclude data from the first two sentences.

improve those systems' understandability. Without improvement, current systems are likely to be frustrating and difficult to use for a large minority of the population. One thing that seemed clear, especially from the second study, is that *good* groups have a significant advantage over suboptimal groups, let alone the simple list strategy.

It should be pointed out that the general similarity in subject performance between the two studies suggests that the relatively low quality diphone speech did not impact understandability significantly compared to the overall difficulty of the task itself. The use of much higher quality commercial synthesis did not result in noticably better results in terms of overall success in remembering lists of any length, let alone show better understanding of longer lists. While there is clearly an understandability difference between low and high quality synthesis, that effect seems to be overwhelmed in this instance by lexical effects.

4.1.2 Time-Constrained Presentation

Given the fairly short restrictions of human auditory memory, one approach for improving output understandability is to limit the time of the output. This can be accomplished, particularly with structured or complex information, by using the inherent structure of the information to be presented, and having that structure guide the generation of utterances that fit within a given time limit.

Proposed Solutions

For the structured information we are investigating, it can be reasonable to represent the information in a tree-like manner, such that the leaves of the tree represent the most detailed information available, and higher-level nodes are increasingly general summaries of their children. An example of this can be seen in Figure 4.12. Our proposed language generation system would thus follow a relatively simple algorithm to produce utterances. We begin by determining the maximum time for the utterance. In most cases, this should be five seconds or less, as a default, since

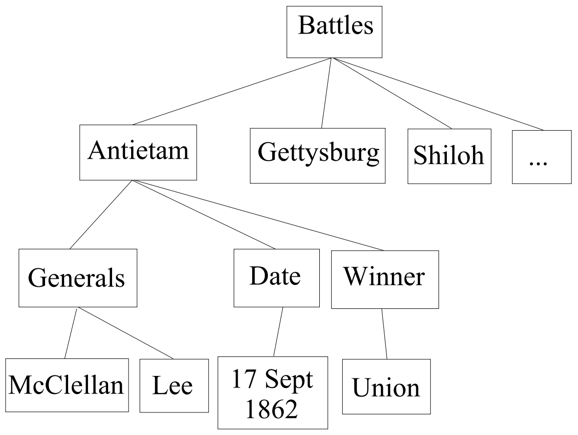


Figure 4.12: Example of structured information, represented as a tree.

this length takes human capabilities into account. Other possible lengths – based on user knowledge, user preferences, and conversational context, among other things – include 10 seconds, 15 seconds, and unlimited. The last should be used extremely rarely, only when absolutely required or explicitly preferred, as there will be a significant understandability decrease as utterances lengthen.

We next establish the maximum pieces of information that can be presented in the allowable time frame. Fortunately, since we are producing synthetic speech, we will have easy access to the length of utterances that we are generating. However, determining this will require taking into account the information we are presenting (the information may be capable of being grouped logically to allow for chunking [Miller, 1956]) as well as any knowledge we have of the user (users familiar with the information being presented will more readily understand abbreviations, and thus we can make use of them). The default, when we have no knowledge of the user's capabilities, is to assume the user is unfamiliar with the subject being discussed. Often, establishing the optimal amount of information that can be presented will require domain-specific knowledge. While this means that the language generation system is also domain dependent, this is already often the case, and we also feel that a domain-independent system can show sufficient understandability improvements even without producing optimal utterances.

In cases where we have multiple pieces of information which are similar or related, either in a single response or in response to subsequent queries, we can use *tapered presentation* as in [Yankelovich et al., 1995] to produce shorter utterances. Tapered presentation refers to reducing or removing extraneous and repetetive words from subsequent utterances. In a first utterance, the template is given for how the information is being presented, and following utterances eliminate repeated words from the template that can be assumed. For example,

```
There is a 61C leaving from CMU at 5:30, arriving at Forbes and Murray at 5:38.
         A 61A leaving          at 5:41, arriving                      at 5:47.
         A 501 leaving          at 5:44, arriving                      at 5:52.
```

The preferred behavior should be to start with the most detailed information that answers the user query, then generate the shortest possible utterance. If that utterance exceeds the allowable length, proceed up the information tree to the next most general node that can satisfy the query. This process should continue until either an utterance is produced that is within the time requirements or no utterance that answers the query can be produced in the allowable time. In the latter case, we should generate the shortest answer that can provide an answer.

Another strategy to examine is prosodic modification of the synthetic output, primarily to emphasize important information. Additionally, we would also like to use altered prosody to call attention to any difficult or unusual information, much as humans will do. However, this is significantly more challenging than other potential tactics, and thus more likely to be attempted after we have implemented the simpler strategies.

Experimental Results

In order to determine which strategies are effective at improving understandability, we designed and performed a simple user study. Subjects were asked to listen to 16 synthetically-produced utterances, with varying time lengths, and then describe in their own words the information from the utterance. The utterances were

There are 12 poster and 12 oral sessions on Wednesday.
There are 9 speech recognition, 3 signal processing, dialog, and prosody, and 4 other sessions on Tuesday.
"Influence of syntax on prosodic boundary prediction" is an oral presentation on Wednesday at 10:40 am.

Figure 4.13: Example utterances from the user study. These examples are from the two shortest time categories.

generated using a modern, high-quality commercial synthesis engine. All of the utterances contained information about papers or sessions from Interspeech 2005; several examples can be seen in Figure 4.13. The utterances took the form of fluent sentences; that is, they did not simply say "25 recognition sessions", but "There are 25 speech recognition sessions". As we did not want to explicitly test memory, but understandability, subjects were allowed to hear utterances twice. Utterances had one of four lengths – under 5 seconds, under 15 seconds, under 30 seconds, and unlimited – and fell into three distinct stylistic categories: naïve "read everything", summaries produced by a freely available text summarization tool, and utterances that could be produced by the algorithm described above. Eight subjects participated in this study, all of whom were familiar with speech technology.

Subjects were evaluated in two ways; first, on how many of the concepts present in each utterance they understood, and second, on how many concepts associated with the information in the utterance they understood. These can differ because shorter utterances may not attempt to convey as much of the information as the longer ones, and thus could look artificially better despite providing less information to the user. For example, an utterance could describe to the user 5 out of the 10 total concepts related to a query, of which the user understood 4. The user would thus have 80% understanding of what was presented to him, but only 40% understanding of the full answer to the query. This contrasts with an utterance presenting 9 out of 10 concepts for which the user correctly understands 5; this would result in 56% and 50% understanding rates, respectively.

Overall, subjects did quite well at understanding the content of the shorter

Utterance Time	**Correct (Presented)**	**Correct (Total)**
Under 5 seconds	95%	42%
Under 15 seconds	56%	33%
Under 30 seconds	44%	34%
Unlimited	13%	13%

Table 4.1: Results from this user study, grouped by utterance length

utterances. Not unexpectedly, their performance dropped steadily as the utterances became longer, with exceedingly long utterances showing very poor understanding. These results can be seen in Table 4.1.

It is interesting to confirm that presenting more information does not necessarily result in more information being understood; in fact it is clear that presenting too much information has a detrimental effect on overall understanding, as can be seen with the Unlimited category. The reverse also seems to be true; the best overall understanding occured when only small percentages of the information were given (the under-5 category), though the result is not statistically significant. It could also be that the shorter utterances had fewer concepts to convey, in which case they would appear superficially better. More results exploring this are needed to explain the cause.

It should be noted that most subjects, when encountering longer utterances, commented on the increased difficulty of remembering and understanding what they were being told. For the time-unlimited utterances (some of which could exceed two minutes in length), *every* subject indicated they felt the task was too difficult. This would seem to be supported by the relatively poor performance for those utterances, and strongly suggests system utterances should not exceed 30 seconds for any reason. No noticeable improvement was observed when using automatically generated summaries as opposed to reading the full abstract; indeed, in some cases, the full abstract had a higher percentage of provided concepts understood. This result is most likely due to the overall low quality of the summaries produced by the text summarization tools that were freely available at the time.

4.1.3 Presentation Style and Fluency

While there has been some work investigating the use of stylistic changes to improve the understandability of synthetic speech [Oh and Rudnicky, 2000], our focus here is on semantic delivery strategies that can impact how understandable and usable our presentation is.

Too often, the strategy speech systems employ when providing a large amount of complex information is a simple, read-all-the-text method that is usually inappropriate for the content being presented. We believe that taking advantage of structured information by presenting it with the structure in mind should improve how understandable it is. Further, we believe removing extraneous, irrelevant information will also help, by allowing people to focus on the most important items, as well as decreasing the amount of speech their auditory memory needs to handle. With this in mind, we have attempted to test our presentation strategies' capability to provide more understandable spoken output.

Experimental Results

We have designed and performed an experiment designed to evaluate the effect of semantic presentation style on understandability. Participants were asked to listen to seven descriptions, over the telephone, of battles from the US Civil War, and then write down the important information about the battle. Specifically, they were asked to write down what they would tell a person who knew nothing about the civil war that asked the question "What can you tell me about this battle?" We deliberately chose *not* to provide any direction to subjects about what constituted the important information, as we felt that would unduly influence their responses.

We initially believed that the concise list style would be the easiest to understand, both because it structures and highlights the desired information, and because the length of the audio is shorter than the other descriptions. Based on other work [Langner et al., 2006], we expected the length of the full text to prove prohibitively long, and hamper understanding. As the summaries are shorter, but

	Concise	**ASum**	**HSum**	**Full**
Non-Elderly	87.7	65.4	68.9	73.1
Elderly	49.3	44.3	41.5	43.1
Overall	66.0	53.5	53.4	56.0

Table 4.2: Percent understanding results for this study, by presentation style.

generally providing the same set of information, we would expect their understanding to be improved, though not by as much as the concise list. We were particularly interested in any performance difference between a good human summary, and a simple machine-generated one, as automatic summaries will be cheaper and easier to obtain.

As we expected, there was a performance difference between the young and elderly groups, with the non-elderly group showing noticably higher understanding. However, the trends for both groups are largely similar, which is reassuring given our expectation that changes leading to understandability improvements for the elderly will be reflected in the general population as well. Our results from this study are shown in Table 4.2. "ASum" and "HSum" refer to the automatic- and human-generated summaries, respectively. All values are percent correct understanding.

These results show a clear understanding improvement with the concise list presentation style; in the non-elderly group, this result is significant with $p < 0.01$. What is surprising here is that neither the automatic nor human summaries are noticiably different from the full text in understandability. The audio length for the summarized versions, though, is still shorter than the full text.

However, looking closer at the results by style, we observed that some of the automatic summaries cropped out relevant information, and thus participants never had the opportunity to understand those items. This, obviously, has a negative impact on the understandability score for this style. If we adjust for the number of relevant items actually presented, our results differ slightly, as you can see in Table 4.3. In terms of style understandability, it is appropriate to make this adjustment; however, from the perspective of *presenting* a specific set of information

	Concise	**ASum**	**HSum**	**Full**
Non-Elderly	87.7	70.0	68.9	73.1
Elderly	49.3	48.6	41.5	43.1
Overall	66.0	58.0	53.4	56.0

Table 4.3: Percent understanding results for this study, by presentation style, adjusted for information presented.

to be understood, such an adjustment is not truly helpful. Effectively, the automatic summary has a defect that causes some information not to be conveyed at all, which is no better than presenting it in an unintelligible manner. Thus, while we report these adjusted scores, we feel the original numbers are a more accurate representation of understanding for any task where the goal is successfully conveying a particular set of information. It is still the case that the most understandable strategy, regardless of the measurement, is the concise list style.

Even without the adjustments, though, the automatic summaries showed comparable understanding to the human produced summaries. This is definitely a useful result, as the automatic summaries can be generated in seconds, versus the 30 minutes the human summaries required. For larger-scale tasks with more information, the time and effort savings given by the automatic process are compelling when summaries are needed.

4.1.4 Discussion

Based on earlier understandability work with the elderly, we expected that participants with hearing problems would have a considerably more difficult time with this task. Our results, however, show no appreciable difference between people who have hearing problems and those who claim to have none, though this is obviously subject to sample size issues. It should be noted that self-reporting of hearing problems is potentially inaccurate, as some people either do not know, or will not admit, that they have them, and so subjects in the “normal hearing” group might have hearing deficiencies. Without performing a hearing test, which we were not able to do for this evaluation, there is little we can do besides trust the subjects to accurately describe their hearing.

Noting, however, the general difficulty the elderly group had with this task compared to the younger group, we attempted to isolate other possible causes besides hearing. One obvious possibility is memory, though the task was designed to limit the memory requirements. The most likely difference, however, was writing speed. The non-elderly group was, in general, capable of writing quickly while listening to the descriptions. Meanwhile, most of the elderly group either did not or could not do this, in some cases due to the early stages of degenerative conditions such as Parkinson's disease. This limited the amount of information they would write down for each description. Other possibilities include fewer elderly participants identifying the same "important" information as the task specified, though we found no noticeable difference between the non-elderly and elderly groups in this respect.

It is quite clear, however, that the concise list presentation style *does* provide more understandable output for desired information. For domains where such a style is possible, it should be used for improved user understanding, except in cases where full sentences are required. Ideally, language generation components of speech systems would be capable of generating this style of output directly from their backend data source, allowing bus timetable information, for example, to be presented in a concise, efficient fashion. It remains to be proven that this strategy will be successful across multiple domains, though we expect that it will. Demonstrating understandability improvements in real applications (such as the Let's Go! Public [Raux et al., 2006] spoken dialog system), rather than artificial tasks such as the study described here, would be the next logical step for this work. Since it would be required when integrated into such a system, we intend to create a language generation tool that is capable of producing this sort of concise output when appropriate.

Other possible directions with this work include attempting to produce a more fluent-sounding style than the "spoken bullet points" method here. If such a style were successful, it would limit concerns from system developers about perceived unnaturalness while still providing better understandability than is currently found.

4.2 Towards a General Approach: Design Considerations

There are several different requirements a language generation system should be concerned with. To be successful, language generation needs to balance machine capabilities with the limitations of human performance, without producing unnatural-sounding speech. Interactive applications also necessitate fairly rapid response times. For simple information, this is a fairly straightforward, though not trivial, task, and the increasing abundance of these systems – from pizza ordering [Choularton and Dale, 2004], to flight reservations [Walker et al., 2001], to bus timetables [Raux et al., 2006], and even commercial customer service phone systems – shows this can be done with varying degrees of success. However, even these fairly simple tasks have examples where more complex information could or should be presented, but the systems are either unable to do so in an understandable fashion, or have had their presentation extensively tailored manually to the specific information. Though it can be effective, manual tailoring tends to be both expensive, requiring expert attention to be done well, and inflexible, requiring additional expert attention to redesign utterances with even minor changes in the information content. Since complex information is often structured in some fashion – in a table or list, for example – we feel there should be a more automatic solution that can take advantage of that structure to generate understandable spoken language.

How best to take advantage of that structure will of course be influenced by human capabilities. Results from cognitive psychology provide some direction to our approach. The primacy and recency effects [Murdock, 1962, Bousfield et al., 1958] are well-studied phenomena, suggesting important items should be presented either first or last in a list. Furthermore, there is the standard "seven, plus or minus two" rule [Miller, 1956] that is commonly used, though more recent work [Baddeley, 1994, 1999] suggests that this "rule" is at best a guideline, and not applicable except in specific, limited cases [Jones, 2002]. In fact, it seems that the number of items being presented is not the limiting factor in remembering them, but instead the length of the sound being listened to; humans have approximately two seconds of useful auditory memory [Baddeley et al., 1975] to work with. Finally, though concentrating on visual interfaces, other work has made a strong case that even if the 7$\pm$2 rule were applicable, it does not make sense to use that as a basis for designing a human-computer interface, as this is the upper limit of human performance [LeCompte, 2000]. Interfaces that require users to continuously operate at their limits will quickly be regarded as frustrating, stressful, and "too much work" to be effective. Thus, three to four items is suggested as a more reasonable memory limit to use.

Further, there is also the consideration that different levels of information are appropriate in different circumstances, consistent with Grice's *cooperative principle* of conversation [Grice, 1975] and its application to language generation systems [Dale and Reiter, 1995]. For example, questions about what a restaurant serves are better answered with a higher-level outline of choices (such as "seafood, steak, and pasta") rather than a detailed description of the menu, whereas a good answer to a question about available pizza toppings isn't "various meats and vegetables".

Taking all of these different constraints into account suggests that in the general case, generated utterances should be fairly short in length, with a small number of different items, and as detailed yet concise as possible. At least one recent study [Langner et al., 2006] suggests that time-limited utterances, particularly those five seconds and under, are more understandable than longer ones, given the same speaking rate; likely this is due to humans' relatively short auditory memory. Other research suggests generic answers to specific queries is inadequate [Varges et al., 2006], given that there may be a mismatch between the system and user understanding, and thus replies reiterating the query constraints would serve as implicit confirmation of the user's request, making the overall conversation more understandable. Note that these issues will tend to work against each other; finding a balance between a short, but informative and confirming utterance would appear to be significant in generating understandable speech.

Our earlier work in this area demonstrates that there is considerable room for improvement for synthetic speech understandability, even when applying fairly simple approaches. This confirms the generally held view that natural speech is more understandable; however, the challenge now is to obtain increased understandability for synthetic speech in general. While natural speech is an ideal baseline and can provide guidance towards improving speech synthesis, there are other possibilities besides simply imitating a human speaker. Indeed, the “concise list” presentation style is fairly unnatural as an approach, but can demonstrate improved understanding from human listeners.

4.3 The uGloss Framework

4.3.1 Algorithmic Description

We propose a general-purpose algorithm that can be used to influence the generation of understandable spoken output. We feel that this approach is capable of being used in multiple domains, while still allowing for domain-specific knowledge to be used for further improvements.

Our algorithm, given some internal representation of the information to be presented, as well as the desired time limit for the speech and optionally a desired style of spoken presentation, should take that information and produce speech fitting those constraints. The default presentation would be fluent text, though this can be overridden if appropriate to concise, “bullet point” style presentation, or others.

A pseudocode version of this algorithm is shown in Figure 4.14.

```
GenerateText (InternalRepresentation IRep, TimeLimit T)
  Form groups from IRep
    identify structural features of information in IRep (automatic/by expert)
    use structural features to mark information similarities in IRep
    identify items with similar features from entire domain *
    group items based on similarities
    flag groups which comprise all domain items with that featureset *
  Generate (style-appropriate) text from groups and IRep
  Determine length of time L required to speak this text
  If (L > T)
    produce new groups
    if no new groups can be formed, use current text (best-effort)
    otherwise return to generation step
  Synthesize speech from generated text

* denotes optional step
```

Figure 4.14: Our proposed algorithm for influencing understandable spoken language generation.

4.3.2 Implementation

The uGloss framework starts with the premise that shorter, more concise utterances are more understandable, and thus is geared towards generating those types of utterances. The main way our approach attempts to reduce utterance speaking time is through grouping relevant items together, rather than speaking each item individually. Complex information typically has an inherent structure, and the uGloss approach aims to take advantage of that structure by using it to group items before presenting them.

Grouping can either be done in advance by a human expert, or learned automatically from the structural features. For small simple tasks, using a human expert might be feasible, but for most applications this is not a viable choice. Some domain knowledge may be useful in selecting appropriate structural features, but should not be absolutely required. Using those features, it should be possible to learn which items are similar, and therefore capable of being grouped together.

Group generation is a bottom-up method; we start with the individual items, and proceed to make larger and larger groups as needed.

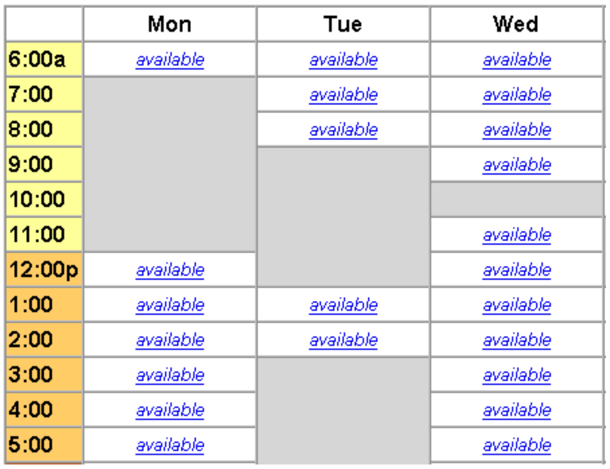
Optionally, we can also look at the entire database and identify groups from there, rather than just the subset that fulfills some request. The advantage to this extra step would be to allow generation of utterances that say “All of these” rather than a range that includes those items, by matching the subset groups that are the same as full-domain groups. Different levels of grouping from the entire database would allow for different “all” constructions. For example, in the bus schedule domain, possible answers to the question “What bus can I take” could be “Any of the 61’s”, “Any of the next 3 buses that come” or “Any bus will work”, rather than providing a long list of valid bus numbers.

The other major parameter to the uGloss algorithm is a time constraint. This constraint is not a hard limit – that is, we would generate nothing if there was no utterance short enough to convey the information within the limit – but a “best effort” guideline. We attempt to generate utterances within that limit, but if we cannot we generate the shortest utterance we are able to.

Based on these parameters, we iteratively form groups and generate utterances until we have an utterance that can be spoken within the time limit, or we are unable to produce a shorter utterance that conveys the desired information. uGloss is sufficiently general that it should be possible to use across multiple domains. Further, should we find other parameters that are useful in influencing output understandability, it would not be hard to incorporate them.

4.3.3 Evaluation and Effectiveness

We performed a very small study to determine the effectiveness of this approach, as well as to compare to human-generated utterances. Given a schedule showing the availability status of a tennis court for the week, people were asked to answer questions from someone trying to reserve the court at various times. The requests generally were to reserve the court for one hour out of a several hour block by specifying a general time range (e.g. Wednesday afternoon, Monday evening, etc.).



	Mon	Tue	Wed
6:00a	available	available	available
7:00		available	available
8:00		available	available
9:00			available
10:00			
11:00			available
12:00p	available		available
1:00	available	available	available
2:00	available	available	available
3:00	available		available
4:00	available		available
5:00	available		available

Figure 4.15: A partial example of the presented schedule.

These ranges corresponded to times where the court was available for the entire, or only part of, the requested range, as well as when the courts were completely unavailable. Subjects were told to answer naturally, as if someone had said to them, “I want to play on <day-time range>, what time can I reserve a court for?” Figure 4.15 shows an excerpt from the schedule, showing the morning and afternoon availability for a few days.

All of the subjects were educated young adult, native speakers. Human responses to these requests were varied, but generally consistent. The “obvious” conditions, where the courts were either available or unavailable for the entire range produced effectively identical answers, with some variation in the exact wording used. In the case where no reservation could be made, most people used some form of the phrase “I’m sorry” in their answer; it is interesting to note that about one third of the participants offered suggestions for other reservations they would be able to fill. The most intriguing answers came for the request to make a reservation on Monday morning. All but one subject made note of the fact that the only morning time was at 6:00, and typically used language that showed their expectation was that this would not be an acceptable time despite fulfilling the stated request. The implication is that unexpected or unusual answers should be presented differently than “normal” answers. Several examples of the human responses are shown

Wednesday Afternoon
“You can reserve a court noon through 5pm.”
“The court is open the entire afternoon.”
“Sure, what time would you like?”

Monday Morning
“If you’re willing to come in really early, you can reserve a court from 6 to 7.”
“Only the slot at 6am.”
“The only time on Monday morning is at 6am, is that okay?”

Figure 4.16: Example responses from two time conditions.

in Figure 4.16.

For the most part, our framework seems capable of generating similar utterances, though there are some differences with human responses in some more complex conditions. The main areas where the human responses seem to be of higher quality than our generation are ackowledgement of unusual or strange times, implicit reference to an unconstrained situation, non-repetitive phrasing, and attempts to resolve a non-fulfillable request. We feel it should be possible for a generation system to deal with all of these to some degree, so the perceived quality should be able to be improved.

4.3.4 Other Considerations

As described earlier, summarization, or the identification of items that can convey the important and central meaning of a set of information, could be used to more efficiently present information in spoken output. Related to summaries are subsets, where some, but not all, of the information is presented, ideally in a fashion that indicates there is more available information. In some ways, both subsets and a high-quality summary are similar to what we are attempting to accomplish with uGloss: a compact yet accurate (spoken) presentation of some set of information that can be readily understood by a human listener. For subsets, this is accomplished by omitting some amount of the information in the initial presentation, and for summaries, by altering the language without changing the underlying meaning. Indeed, using a summarization tool as an aid in producing understandable output may provide some improvements. However, the core task we are proposing is not a summarization task, merely one that could perhaps benefit from a good automatic summarization system. Work in summarization has shown that good summaries do make information easier for people to find and understand [McKeown et al., 2005], so this would not be surprising.

Though we do not intend to build a summarization system, it might be possible to use similar techniques to modify the language generated for spoken output. Most work in speech summarization is using output from speech recognition systems, which will have errors, yet still can show improvements. For a language generation task, we control the text, which means that any summarization should be easier: there are no recognizer errors to "work around". While speech summarization routinely makes use of features, such as acoustic features, that would not be applicable for our task, there has been work with other features that would likely be helpful, such as structural [Maskey and Hirschberg, 2003], lexical, and discourse, among others [Maskey and Hirschberg, 2005]; this work demonstrates that these features can be used to generate effective summaries. If we apply this to our task – rather than summarizing articles and documents, we summarize the originally-generated output – it could possibly provide a more concise, yet still accurate, presentation. It may be worthwhile to explore this possibility using a summarization system in concert with our language generation framework.

Chapter 5

The MOUNTAIN Language Generation System

5.1 Motivation

Recent years have seen noticeable improvement in dialog systems, but even state-of-the-art systems still have limitations. Improvements in speech recognition and dialog management have made it possible to have more natural interactions, but frequently the potential of dialog systems to have truly natural conversations will be held back by their rudimentary language generation components. This observation is not new, having been noticed for years [Rambow et al., 2001, Chambers and Allen, 2004], but continues to be an issue.

Templates and canned text, because they are conceptually simple and require minimal expertise to write, are still one of the most commonly encountered methods of language generation in dialog systems. They are far more frequently seen than more state-of-the-art language generation systems, which tend to require a linguistics expert to be able to use effectively. This is despite the drawbacks of using templates, such as generally unnatural and repetitive output, significant creation cost, and a lack of portability between applications. Efforts to provide more varied template output [Oh and Rudnicky, 2000] have been able to reduce the perceived repetition, but the end result is still noticeably different from natural, human-generated output. Further, as the number and complexity of the templates increases, they become increasingly more expensive to design, create, and maintain.

There has been increasing interest in applying machine learning to spoken dialog research. Nearly all of the typical components of a dialog system have had some effort made to use machine learning to improve them; these are nicely summarized by Lemon and Pietquin [2007]. It seems, though, that trainable language generation for dialog has seen comparatively less work than other modules like ASR, dialog management, and related areas like user simulation. Given the general success of these efforts in related areas, we feel it is likely that such an approach can also work well for language generation.

The motivation for this work is to allow for speech applications with more human-like output than templates are typically capable of achieving, while maintaining the general simplicity of creating a typical template-based system. Some users of spoken dialog systems will interact with a system using a human metaphor [Edlund et al., 2006], and these users expect human-like responses. For such users, the quality of the dialog system they are interacting with in part depends on how natural its responses are [Edlund et al., 2008].

5.2 Approach

Because language generation for dialog systems has several key differences from general text generation, as discussed by Horacek [2003], we feel a dialog system should have language generation that is tailored for its needs. In particular, for many dialog platforms, including the Olympus framework [Bohus et al., 2007], generation is primarily only for realizing natural language surface forms that correspond to some internal state; often this is referred to as *tactical* generation. In practical terms, what is done is to convert the machine's representation of the dialog state into fluent and natural-sounding sentences. If one thinks of the dialog state representations as a highly structured (and possibly simplistic) language, then this task can be viewed as a *translation* problem, where the goal is to translate from the highly structured internal language of states to fluent and natural English (or any other language) sentences. Wong [2007] describes a language generation system that uses machine translation techniques; the approach effectively is the inverse of a semantic parser [Wong and Mooney, 2007], and translates meaning representations into natural language. We propose a similar approach, taking as a source language an internal state representation (such as a dialog state) and generating a natural language surface form with a machine translation engine.

Though this type of approach has been tried in some areas of natural language generation – particularly paraphrase generation [Quirk et al., 2004], though also more recently tested with more general-purpose generation [Belz and Kow, 2009] – we have not seen it applied for spoken dialog applications. For general-purpose language generation, the more linguistically-aware translation approaches of Wong and Mooney [2007] can probably outperform linguistically-blind statistical machine translation methods. However, as mentioned above, generation for spoken dialog applications is different than general NLG; in the former case, the source language to be translated is highly structured and likely not to have rich linguistic information. Given that, it seems possible that a purely statistical approach may work sufficiently well to be usable.

Generation using this approach requires a parallel data set, so that a mapping between the internal "language" and the natural surface forms can be learned. Unlike some other advanced NLG systems, we are not concerned with any significant amount of linguistic details, such as part of speech, agreement, or semantic relations, among others, in our input data. While these can clearly be helpful in producing high-quality output, there are significant costs for systems that require a high level of linguistic expertise or detailed annotation to be able to use; not all developers of dialog systems will be able to devote resources to have an expert design and annotate corpora for their generation module. Therefore, one of the considerations in our approach is that its implementation require only similar developer skills as writing templates.

Since most dialog applications exist within a known domain – that is, they do not have fully open-ended conversations – the set of things that can be talked about is closed, if possibly large. Thus, for tasks whose domain can be covered with a template-based system, it should also be possible to create a reasonably-sized parallel training corpus.

5.3 MOUNTAIN: Machine Translation NLG

We present MOUNTAIN, a **m**achine **t**ranslation approach for **n**atural **l**anguage **g**eneration. In our implementation, we have used the Moses machine translation system [Koehn et al., 2007], which makes use of the Giza++ [Och and Ney, 2003] translation model training toolkit and the SRILM [Stolcke, 2002] language model toolkit. Though we have used Moses as the translation engine, because it and its support tools are freely available, nothing in the approach for MOUNTAIN restricts us to this specific engine.

MOUNTAIN requires only a parallel corpus of states in an internal language aligned with corresponding natural language surface forms. This parallel corpus is used to train a translation model which is capable of translating from the structured internal language to appropriate natural language. Additionally, the natural language corpus is used to train a language model for the target language. As described above, the internal language can be a structured representation of the dialog state – what the dialog manager intends to convey to the user.

Once the models have been trained, MOUNTAIN uses the translation engine to generate output utterances, given “sentences” from the internal language. Moses uses the trained models to translate into the target natural language; the resulting output is the best result from the translation engine. Because of the way Moses works, the output may not only consist of examples lifted straight from the training corpus, but can also combine several examples to form novel sentences in the target language.

It should be noted that the entire process used by MOUNTAIN, from training to generation, does not require any specific linguistic analysis or domain knowledge, and thus can be considered a domain-independent approach. In fact, MOUNTAIN is also *language*-independent, provided the target language is able to be used by the training tools (such as the tokenizer and language model trainer).

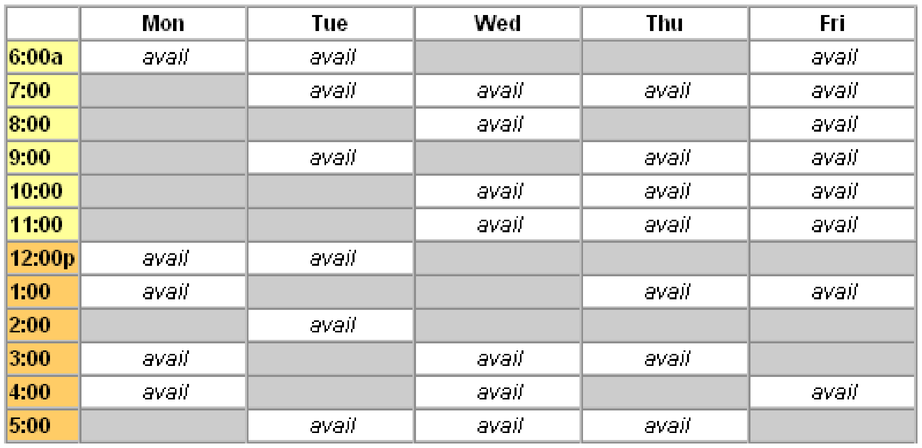
5.4 Training and Use of MOUNTAIN

5.4.1 Application

To demonstrate our generation approach, we chose a fairly simple, but reasonable application: a scheduling task for a limited resource. In this case, the resource is a tennis court, available to reserve for one-hour blocks throughout the day. We have collected a corpus of human-generated responses in this domain. Given a schedule showing the availability status of a tennis court for the week, people were asked to answer questions from someone trying to reserve the court at various times. The requests generally were to reserve the court for one hour out of a several hour block by specifying a general time range (e.g. Wednesday afternoon, Monday evening, etc.). These included examples where the court was available for the entire range, only for part of the time, and where it was completely unavailable. Responders were told to answer naturally, as if someone had said to them, "I want to play on <day-time range>, what time can I reserve a court for?" Figure 5.1 shows an excerpt from an example schedule. Though trivial, this interaction can easily be seen as part of a larger spoken dialog application.

5.4.2 Corpus

The collected corpus consists of about 800 responses, which are labeled with information about the specific schedule situation they describe. That information, which can be thought of as a dialog manager state that would be passed to an



	Mon	Tue	Wed	Thu	Fri
6:00a	*avail*	*avail*			*avail*
7:00		*avail*	*avail*	*avail*	*avail*
8:00			*avail*		*avail*
9:00		*avail*		*avail*	*avail*
10:00			*avail*	*avail*	*avail*
11:00			*avail*	*avail*	*avail*
12:00p	*avail*	*avail*			
1:00	*avail*			*avail*	*avail*
2:00		*avail*			
3:00	*avail*		*avail*	*avail*	
4:00	*avail*		*avail*		*avail*
5:00		*avail*	*avail*	*avail*	

Figure 5.1: A partial example of the presented schedule.

NLG module, is effectively an internal language. Thus, this corpus is, in fact, an aligned, parallel bilingual data set, defining equivalent English surface forms for the internal language. Sentences in the internal language consist of 3 tokens: a code corresponding to the day, a code corresponding to the time period, and a string that represents the court availability during that time. Example sentences include “111111 d2 t3” and “001110 d5 t1”.

Approximately 20 people provided responses to form this corpus, all of whom were young adult, native speakers. Human responses to these requests were varied, but generally consistent. For cases where no reservation could be made, many responses used some form of the phrase “I’m sorry”. It is also interesting to note that several responses offered suggestions for other similar reservations that could be made instead.

We had nearly one-quarter of the corpus scored by human evaluators. In keeping with the task design, the evaluator was shown a schedule and request, along with the response from the corpus, which was presented as their co-worker’s answer. They were then asked to rate that answer on a 5-point Likert scale, and additionally provide their own answer to the request. Figure 5.2 shows a histogram of the ratings. Most responses are rated well; however, a significant portion of the corpus has a low score. This is likely due to errors made by the human responders.

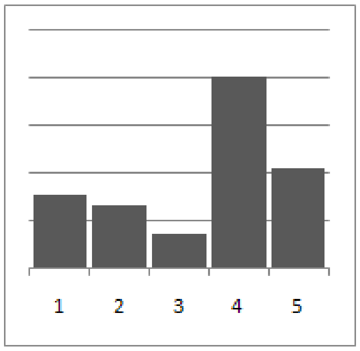


Figure 5.2: Histogram of ratings given to the collected corpus.

For example, some responders misread the schedule and reversed the availability (instead of all slots free, they responded as if all slots were full).

5.4.3 Training for Generation

To make the collected corpus more complete, we first make several fairly standard modifications. Since it seems reasonable in this domain to have the output be independent of the day-of-week – that is, a sentence used to describe a full court on Monday afternoon can also be used for Thursday afternoon, changing only the word for the day – we boosted the training data by cloning responses for all 7 days, changing the day word in the sentence as appropriate. Additionally, since the internal language uses a fixed token to correspond to particular days (i.e., d3 = Wednesday), we added these translations directly to the bilingual dictionary. Likewise, the tokens for time (i.e., t1 = morning) were similarly added. The result is that the training corpus size is increased to about 4500 state-response pairs.

With the modified parallel corpus, we then train a translation model using the Moses tools. The first step is to tokenize and case-normalize both the internal and English training sentences. Due to the designed structure of the internal language, the English tokenizer can be used for both sets of sentences. Once this is complete, we train a trigram language model for the target language (English in our case) using the SRILM tool. Finally, we train a phrase model using the Moses toolkit,

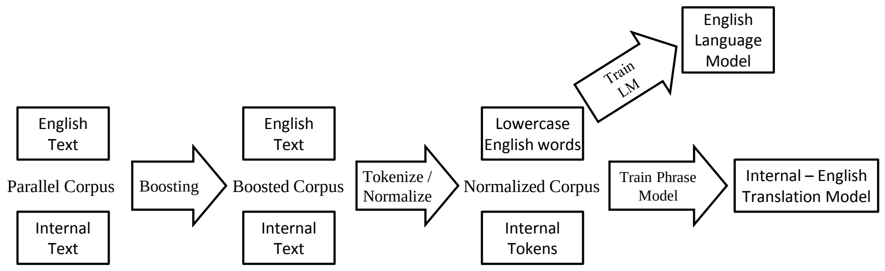


Figure 5.3: The MOUNTAIN training process from initial parallel corpus to trained models.

which produces the necessary phrase and reordering tables for translation. This process is shown in Figure 5.3.

Once the model is trained, the model can be tuned using a separate, held out corpus. For this initial work being described, we have not done any tuning, though it seems likely that this would improve the resulting generation.

5.4.4 Output

Table 5.1 shows several example input-output pairs. Though several outputs have come entirely from single examples in the training data, likely due to the presence of these exact pairs in the parallel corpus, most of the outputs have combined phrases from two or more examples, producing a novel English sentence unseen in the training data. Overall, more than three quarters of the generated responses are not present in the original corpus.

The examples in the table show the variety of ways availability can be expressed. Though the method we are using here generates the same response for a particular input, similar but non-identical inputs can produce noticeable surface variation – but similar semantics – in the resulting output. The source of this is likely the variation present in the training corpus.

Though many of the generated responses are interesting and appropriate answers, there are also responses that contain errors of varying severity. Some, like

000000 d5 t3	friday evening is completely closed
100000 d2 t2	the only time available is noon
111111 d4 t1	the court is open all morning
111111 d1 t3	you can reserve a court anytime on monday evening
100011 d5 t3	six , ten or eleven
010011 d3 t2	you can reserve a court at 1pm , 4pm and 5pm on wednesday
011001 d4 t3	any time but 6 , 9 and 10
111011 d7 d2	afternoon except the 3pm block
111100 d1 t2	you can reserve a court is free anytime from noon until 3
110111 d6 t3	saturday evening . ooh , that

Table 5.1: Example output from the MOUNTAIN system

the second-to-last example in the table, are relatively trivial with some grammaticality problems. These are obviously not ideal, though with appropriate prosody when spoken, they could be made to sound like a person's natural, unplanned, conversational response. Other errors, however, are more problematic, such as the last example in the table. Here we see what appears to be a partial translation failure, since the day and time are correctly rendered in the response; however, the remaining important content information has been omitted. Minimizing or eliminating these sorts of errors is crucial to the effectiveness of MOUNTAIN as a generation method.

5.5 Evaluation of MOUNTAIN Generation

While an informal examination of MOUNTAIN output seems to show it is more or less capable of generating acceptable responses (though with some errors), it is clear that a more structured evaluation is required. Given the issues raised by Dale and Mellish [1998] and the earlier discussion of evaluation in Chapter 3, the best approach appears to be to evaluate using automatic metrics and human-based metrics. This also provides an additional opportunity to test the performance and correlation of automatic metrics compared to human judgements, continuing the work by others in that regard [Stent et al., 2005, Reiter and Belz, 2009].

As discussed earlier, the most common means of evaluating natural language generation is with human-based measures involving rating generation examples, with increasing interest recently in using automatic measures like those used in other fields. The strongest push for evaluation using automatic metrics has come from the corpus-based NLG groups, which makes sense since these groups are better able to benefit from optimizations that can happen when evaluation results are available quickly compared to handcrafted systems which take significantly longer to build.

Most commonly, NLG evaluations have reported NIST scores [Doddington, 2002] or BLEU scores [Papineni et al., 2002]. Both metrics basically measure n-gram agreement between test and reference strings. NIST is more heavily weighted to unigram recall (approximately 80% of the contribution to the score), and thus may not be an ideal measure of adequacy, fluency, or human preference. This is probably a reason why it has only shown occasional and at best, moderate, correlation to human judgements of generation output, though BLEU has not been any better [Stent et al., 2005, Belz and Reiter, 2006, Reiter and Belz, 2009].

METEOR [Banerjee and Lavie, 2005] is another MT metric that has been discussed, but not generally used for NLG evaluation. We are unsure why this is, since its design appears to account for more features than NIST or BLEU that might be relevant to NLG output quality.

5.5.1 Automatic Evaluation: BLEU and METEOR

To evaluate MOUNTAIN using BLEU, we used a held-out test set of about 120 responses from the collected corpus that were not part of the training process. These responses were taken from throughout the corpus at various intervals – approximately every eighth entry, to ensure the resulting test set was representative of the corpus and domain. The internal language half of this bilingual set is used as input to MOUNTAIN to produce the test output; the English half of the bilingual set is used as a reference. Individual N-gram scores for the default MOUNTAIN system are as follows:

1-gram	2-gram	3-gram	4-gram	5-gram
0.3198	0.1022	0.0525	0.0300	0.0202

While it is not unusual to see scores decrease with larger N-grams, our default system has a fairly steep dropoff. However, there are several potential areas for improvement available to us. First, recall that the training corpus contains some amount of error, as described in Section 5.4.2, and that some of the corpus has been scored by human evaluators. We can use the scores to exclude some responses from the training set, with the assumption being that removing poor or incorrect responses will improve the resulting generation. We performed this analysis using various exclusion thresholds; these results are shown in Table 5.2. The training corpus for these systems includes only responses with ratings above the indicated threshold, plus any unscored responses from the original corpus.

These results show clear improvement over the baseline system, with statistically significant improvement ($p < 0.1$) for all systems; values in bold are significant with $p < 0.05$. Removing only the poorest examples from the training data does not help as much as removing more – even some which were considered good by a human. However, once the threshold becomes too high, the BLEU scores begin to decrease, likely due to a combination of data sparsity and the well-rated examples being overwhelmed by unrated examples in the corpus. To test this, we

System	1-gram	2-gram	3-gram	4-gram	5-gram
Baseline	0.3198	0.1022	0.0525	0.0300	0.0202
Rating $>$ 1	0.4376	0.1729	0.1079	0.0746	0.0597
Rating $>$ 2	**0.4491**	**0.1919**	0.1169	0.0872	0.0747
Rating $>$ 3	**0.4742**	**0.1963**	0.1212	0.0866	0.0722
Rating $>$ 4	**0.4596**	0.1762	0.1023	0.0693	0.0611

Table 5.2: BLEU scores for systems with excluded training data based on human scoring

System	1-gram	2-gram	3-gram	4-gram	5-gram
Rating $>$ 3	0.4742	0.1963	0.1212	0.0866	0.0722
Rating $>$ 3 (resample)	0.4598	0.1744	0.1011	0.0688	0.0597
Rating $>$ 4	0.4596	0.1762	0.1023	0.0693	0.0611

Table 5.3: BLEU scores after resampling to account for amount of training data

tried resampling the Rating $>$ 3 training set so that it contained the same number of rated examples as the Rating $>$ 4 training set. The results, shown in Table 5.3, seem to confirm this hypothesis, as the similarly-sized training sets show comparable or better scores with the higher exclusion threshold.

We also investigated the effects of the model's *distortion limit* on the resulting output. Distortion limit refers to the amount of word reordering allowed when translating source sentences to the target language. The default value in the Moses training process is 6 – that is, words may appear up to 6 words away from their source-language neighbor in the target language. Using our baseline system, we examined values from 0 (no reordering) to 8, as well as unlimited reordering. We found identical BLEU scores for systems with a limit ≥ 3, and minimal differences (mostly less than .001) between systems with other values; these results are shown in Table 5.4.

Though we had expected a performance effect as the distortion limit changed, our results showed almost no impact whatsoever. Our original intuition was that a

Distortion	1-gram	2-gram	3-gram	4-gram	5-gram
0	0.3194	0.1053	0.0544	0.0302	0.0203
1	0.3194	0.1053	0.0544	0.0302	0.0203
2	0.3222	0.1050	0.0542	0.0300	0.0202
3	0.3198	0.1022	0.0525	0.0300	0.0202
6	0.3198	0.1022	0.0525	0.0300	0.0202
∞	0.3198	0.1022	0.0525	0.0300	0.0202

Table 5.4: BLEU scores for systems with various distortion limits (baseline=6)

higher distortion limit would improve results, due to the nature of our source and target sentences: 3-token source sentences map to much longer target sentences. We had thought that a distortion limit that was too low might cause some correct (and possibly preferred) translations from being generated; however, this does not seem to be the case. Additionally, varying the distortion limit with our improved systems that excluded training data also showed no noticeable difference.

Though BLEU has been shown to be well-correlated to translation results, it may not be the best metric to use for evaluating NLG output because it tends to view output as a set of N-grams rather than sentences. METEOR [Banerjee and Lavie, 2005] may be a useful measure for our needs, as a translation metric that takes more sentence-level information into account than BLEU.

We used METEOR to score the same systems described earlier, trained with some of the corpus data excluded. Results from the same test set, showing precision, recall, f1, and total METEOR score, are in Table 5.5. As with the BLEU scores, there is an improvement seen over the baseline system by excluding training data, and this improvement decreases when the exclusion threshold is too high. Likewise, the systems with thresholds of 2 and 3 are similar, with the former system being marginally better in performance.

System	Precision	Recall	f1	Total
Baseline	0.4225	0.2013	0.2727	0.1950
Rating > 1	0.4489	0.2097	0.2859	0.2028
Rating > 2	0.4533	**0.2248**	**0.3009**	**0.2218**
Rating > 3	**0.4834**	0.2148	0.2974	0.2146
Rating > 4	0.4481	0.2030	0.2794	0.1971

Table 5.5: METEOR scores for systems with excluded training data

5.5.2 Human-scored Evaluation

We performed a small scale evaluation using 4 human evaluators, all young adult, native speakers. The evaluation task was structured similarly to the corpus evaluation described in Section 5.4.3, substituting MOUNTAIN-generated output for the original human responses in the corpus. Overall, the MOUNTAIN output has a lower average rating than the human responses (3.1 compared to 3.4), but a broadly similar rating pattern. Figure 5.4 shows a histogram of these ratings (compare to Figure 5.2 for human-generated responses). The main difference is a much higher incidence of a poor rating (1) for the machine-generated output, but similar rates for good (4) and excellent (5) scores. This seems to indicate MOUNTAIN can generate output similar in quality to human answers, but when it fails, the output is unacceptably poor.

The limited amount of human evaluation done precludes a formal examination of inter-person agreement. However, a per-person histogram (Figure 5.5) shows similar rating patterns from each of the human evaluators. Clearly, a significantly larger-scale human evaluation is warranted, which would give us a more complete subjective measure of the output quality.

5.5.3 Discussion

The results from this small test application show that MOUNTAIN is capable of generating natural, human-like output. In the examples, the day name is frequently not present in the output. Though a slot-filling generation system will typically fill in specific information such as that, in a human-human conversation once the day has been established it does not need to continue to be said. MOUNTAIN, because it is trained from a natural corpus, is able to produce this sort of human-like output.

This initial effort did not attempt to do much in the way of tuning, instead using mostly default settings for various components, such as the language model and parameters in the translation engine. These results thus do not accurately represent the potential of the MOUNTAIN approach, but a useful baseline. The next section

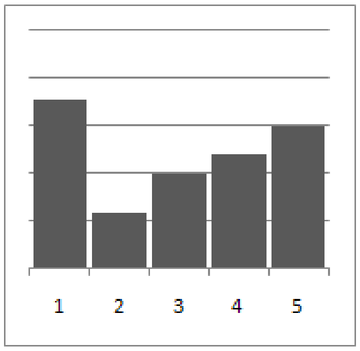


Figure 5.4: Histogram of ratings for MOUNTAIN-generated responses.

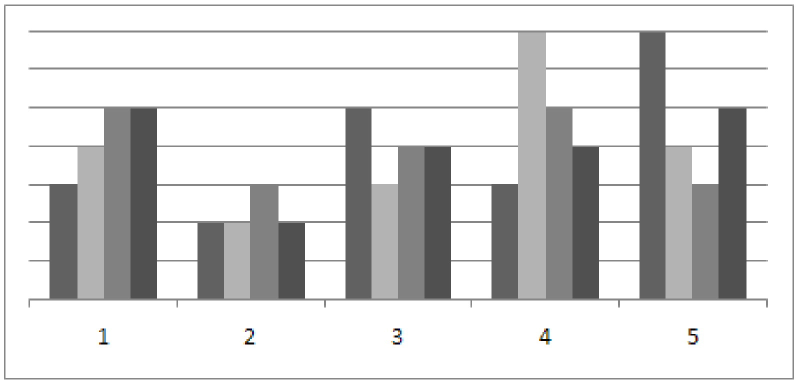


Figure 5.5: Per-person histogram of ratings for MOUNTAIN-generated responses. Each color represents an individual evaluator.

describes various attempts to improve on this baseline by tuning and optimizing for improved generation output.

5.6 Tuning to Improve Generation Results

5.6.1 Training Corpus

As noted earlier, the training corpus we collected was generally good, but not consistently high-quality. This is due to several factors, including human error in generating responses to questions. The initial effort showed that eliminating some amount of data likely to be poor improved the resulting generation; this is not a surprising result. However, the effects were limited due to the relatively small amount of rated data we had in the corpus. With the evidence that using the ratings to exclude data from training gives a significant improvement in generation, we made the effort to have people rate the entire corpus.

Since the corpus is fairly large, this is a substantial task. Further, since people often will have differing opinions about the quality of responses, we should get several ratings for each response in the corpus. The large number of ratings this necessitates precludes us from using our previous method of recruiting local raters – the task is simply too large. We instead used a different approach: the Amazon Mechanical Turk (MTurk), a web-based crowdsourcing application that allows *Requesters* to design and publish tasks for *Workers* to complete. This has been used successfully for several natural language processing tasks already [Kittur et al., 2008, Snow et al., 2008], including transcription of speech for ASR, with results seeming to be of reasonably good quality.

We set up our rating task on Mechanical Turk as before, with people asked to rate their “co-worker’s” answer given a schedule and reservation request, and provide their own answer as well. The task was structured so that Workers could rate as many entries as they wanted, though never the same entry more than once. Additionally, each entry would get at least two ratings. The main advantage to using

Mechanical Turk is speed; by putting us in contact with dozens of Workers, the entire corpus – with just under 2000 ratings – was able to be rated in approximately two days, with minimal effort on our part. Rating quality was initially a concern; however, comparing the MTurk-derived ratings to the earlier ratings from recruited evaluators showed no significant differences.

Note that though we collected additional responses from the rating process, we did not include them in the training data. This is primarily due to uncertainty about MTurk responses' relative quality, since they have not been rated. Further, since we have an additional response for each rating we obtained, this unrated data would put us back in the situation of having significantly more unrated than rated training data.

5.6.2 Language Model

The default settings for the training process build a 3-gram language model. With the amount of data in this application, this could be sufficient, though some of the generation errors suggest their cause is related to combining disparate 3-grams. Using a larger N-gram language model might help alleviate that, provided there is enough training data to deal with sparsity issues. To boost the amount of data for building the language model, we used the responses collected from Mechanical Turk raters as well as the training data for the translation model. This effectively more than doubled the amount of domain text used to build our language model.

We tested 4- and 5-gram models; the 4-gram model had a marginal but positive effect. The 5-gram model did not show additional improvement, suggesting that there is insufficient training data for that model size to be useful.

5.6.3 Internal Language

We hypothesized that one of the reasons for some poor output sentences in this domain is the choice of internal language representation. Though reasonable, the representation used initially embeds many pieces of information in a single token corresponding to the schedule of availability. This is not ideal, partly because it is more difficult for the translation model to learn mappings with a wide disparity in token lengths. That is, having a single token from the internal language correspond to many different English phrases which can range in length from 4 to nearly 20 words in some cases makes learning a good translation needlessly difficult.

To account for this, we tested several different tokenizations that would result in input sentences closer in token length to their English counterparts. The variations which showed more improvement were those that separated the single original schedule token into several tokens. The most trivial change was simply to split the schedule into 6 tokens by adding spaces (i.e., from “011001” to “0 1 1 0 0 1”). This tokenization had a small but positive impact on the generated text. A further refined tokenization was to add the time (using the hour for each slot) in addition to the availability information to each of the individual schedule tokens, since one of the common errors we had observed was incorrect times in the output. That is, times not present in the input would appear in the English translation. This tokenization (i.e., “12b 13a 14a 15b 16b 17a”) more explicitly describes the content to translate rather than making the model infer it from other tokens; it had a more pronounced positive effect on output quality, and made time confusion errors rare.

5.6.4 Translation Parameters

The Moses engine has many different parameters that can affect the translation output. Our initial effort simply used the defaults, however it is likely that tuning them will result in better outputs. One of these, the translation table size, controls how many translation options are considered for each phrase. If this value is too small, valid (and possibly preferable) translations will be pruned, resulting in poorer output; however too large a value will both increase the running time as well as the number of bad translations that get considered, which can also result in poorer output. The default value for the `ttable` parameter is 20; we tested several different values and discovered that a larger value (50) gave better generations without a large impact on speed. This makes some sense, intuitively; our training data has a larger than average amount of variation in translations for a given input and so can benefit from examining more possible translations.

Another parameter, which controls the preferred word length of translations, also impacted our results. Since we observed MOUNTAIN would generate responses several words shorter, on average, than those in the training corpus, we adjusted the length penalty parameter to encourage longer translations. Overall, this provided a slight boost to output quality.

Finally, because the default training configuration produces model weights of "questionable quality",[1] minimum error rate tuning to adjust these weights is recommended. We held out an additional subset of the training data to use as a development set for tuning. Since minimum error rate tuning effectively optimizes for BLEU score, it is not surprising that this will have a positive effect on the resulting output; the drawback is that it reduces the amount of training data, which can be an issue when there is only limited data available.

5.6.5 Overall Effect on Generation

With the changes to our training data, it was necessary to create a new, different test set than we used in our previous evaluations. We tested the original MOUNTAIN system described previously, and compared the results to tuned and optimized systems with the modifications described in this section. Tables 5.6 and 5.7 show the BLEU and METEOR results for these systems. The modification with the largest impact is the improved tokenization of the internal language, followed by the minimum error rate tuning. If we had more data available for a larger development set to use for tuning, it is likely that we would have seen still further improvements.

[1]`http://www.statmt.org/moses/?n=FactoredTraining.Tuning`

System	1-gram	2-gram	3-gram	4-gram	5-gram
Fully-rated Baseline	0.2747	0.1010	0.0442	0.0183	0.0109
LM + Simple Tokenization	0.2913	0.1122	0.0496	0.0217	0.0139
LM + Improved Tokenization	**0.3605**	**0.1983**	**0.1276**	**0.0994**	**0.0808**
LM + Token + Min Error Tuning	**0.4170**	**0.2351**	**0.1549**	**0.1215**	**0.1039**

Table 5.6: BLEU scores for tuned systems. All systems shown use the best rating threshold for excluding training data.

System	Precision	Recall	f1	Total
Fully-rated Baseline	0.3402	0.1818	0.2369	0.1753
LM + Simple Tokenization	0.3250	0.2311	0.2701	0.2048
LM + Improved Tokenization	0.2938	**0.3075**	0.3005	0.2742
LM + Token + Min Error Tuning	**0.4538**	**0.3347**	**0.3853**	**0.3187**

Table 5.7: METEOR scores for tuned systems. All systems shown use the best rating threshold for excluding training data.

5.7 Testing in Another Domain

5.7.1 Weather Forecasting: The SUMTIME-METEO Corpus

While we feel MOUNTAIN is a useful new approach to natural language generation, it would be useful to compare its performance to other NLG systems – ideally in an easily-comparable domain. The SUMTIME-METEO corpus [Sripada et al., 2002] has been used by multiple different generation systems, and offers an ideal opportunity to compare against other approaches in NLG. This corpus consists of wind and precipitation forecasts written by three different human weather forecasters, along with the weather data used by the human forecasters to create them. Specifically, we wanted to repeat the same evaluation used by Belz and Kow [2009], which compared 10 different approaches using the wind forecast data. This subset of the corpus, referred to as Prodigy-METEO [Belz, 2009], extracts the wind forecast statements from the SUMTIME-METEO corpus, as well as the wind data in the form of vectors of 7-tuple numeric data, ultimately forming an aligned parallel corpus suitable for corpus-based NLG. Overall, the corpus contains 465 entries with data and a corresponding forecast text.

5.7.2 Training

We used the same basic training process as the earlier application for the METEO domain. The parallel corpus was tokenized and case-normalized, and we trained an English trigram language model. We also tried using a 4-gram language model; however, for this domain there was no effect on the output. This is likely due to insufficient training data. Finally, as before, we train a phrase model using Moses.

As mentioned above, the input data for the METEO corpus consists of vectors of 7-tuples that correspond to the collected wind data. Each 7-tuple has a sequential ID for the forecast period, wind direction, minimum speed, maximum speed, minimum gust speed, maximum gust speed, and timestamp. If a value is missing or not relevant for a particular 7-tuple, it is represented with a single dash. Each 7-tuple has its values listed in order and separated by commas, with the entire block enclosed in square brackets. Each forecast can have multiple 7-tuples associated with it; they are represented as a vector, also enclosed in square brackets, with individual 7-tuples separated by commas. We considered using the raw vectors in MOUNTAIN with only the default tokenizer, but decided on a custom tokenization instead that would be better suited for training a translation model. Our tokenization strips all square brackets and underscores, replaces commas with spaces, and adds a disambiguating character to the 7-tuple ID token. We also separate various infrequent qualifiers to wind direction from the main direction token (for example, `M-SE`, which means "MAINLY SE", is tokenized to `M- SE`). Figure 5.6 shows an example of both the raw and the tokenized input.

Additionally, the METEO corpus is designed to use 5-fold cross-validation, where the entire corpus is split into 5 overlapping training and test sets, with each fold being trained and tested as independent systems. Since previous evaluations using this corpus have done this, we chose to do so as well to have easily comparable results. Each fold uses slightly more than 400 examples for training and approximately 45 for testing. We did not do any minimum error rate tuning as in the reservation application due to the limited amount of training data in each of the folds. However, we note that such tuning, if we use sufficient data for training and development sets, would likely improve the resulting output from MOUNTAIN.

Raw Input: `[[1,␣NW,25,30,40,-,0600],[2,␣VAR,8,-,-,-,1800],[3,␣SE-E,15,20,-,-,0000]]`
Tokenized: `t1 NW 25 30 40 - 0600 t2 VAR 8 - - - 1800 t3 SE-E 15 20 - - 0000`

Figure 5.6: Example of the raw vector of 7-tuples and the corresponding tokenized form used as input text by MOUNTAIN.

5.7.3 Systems in the Prodigy-METEO Corpus

This section details the systems used by Belz and Kow [2009] in their evaluation. Their generation outputs are distributed along with the aligned forecasts and weather data in the Prodigy-METEO corpus.

SMT-based systems

Two phrase-based statistical machine translation systems were built using the Prodigy-METEO corpus. Both systems used the Moses toolkit [Koehn et al., 2007] and trigram language models; the primary difference between them is their source language representation. Neither system used factored translation models or any tuning.

PBSMT-unstructured: This approach was the simplest, taking the raw input vector and converting it to a sequence of nonterminal symbols, and adding wind direction and speed change information. Otherwise, no normalization or disambiguation of the input tokens was done.

PBSMT-structured: This approach also used the raw input vector, but additionally tagged each of the 7-tuples with explicit predicate-argument structure information. The structure markers were themselves treated as additional tokens in the source language.

PCFG-based systems

There are also five probabalistic context-free grammar generation systems built using the Prodigy-METEO corpus. The particular systems all used a Probabalistic Context-free Representationally Underspecified (*p*CRU) approach [Belz, 2008], which interprets the set of all generation rules such that they themselves define a context-free language, and use a single probabalistic model for generation.

The model can be estimated using either annotated or unannotated source texts, though it appears that only simple tasks are suitable for unannotated data.

The *p*CRU approach requires building a CFG by hand to cover the entire set of input-to-output possibilities, and then estimating a probability distribution to decide among the various different generation rules for a given input. If annotating the source texts is also required, this approach will need a significant amount of skilled manual work to be usable. However, the generated texts distributed with the corpus used systems trained with unannotated data.

All of the PCFG systems used the same input as the PBSMT-unstructured system for their distributed generations.

PCFG-random: This baseline PCFG approach chooses generation rules randomly at each possible decision point, rather than using a trained probabalistic model. Thus, it represents a floor for how well a trained system should be able to do by chance alone.

PCFG-2gram: Another baseline approach, it also does not used a trained probabalistic model, and instead generates all possibilities for a given input. Then, using a bigram language model, the most likely possible generation is determined and subsequently output as the generation result.

PCFG-greedy: This approach uses the simplest method. After estimating the probability distribution from the corpus via *p*CRU, it then applies the single most likely rule for any given decision point.

PCFG-viterbi: This approach expands all non-terminal rules for a given input, and then uses a Viterbi search to determine the most likely output.

PCFG-roulette: This approach decides to expand non-terminal rules based on a non-uniform random distribution. This distribution is proportional to the likelihoods of the various possible expansion rules at that decision point.

PSCFG-based systems

The Prodigy-METEO corpus also has generated output from Probabalistic Synchronous Context Free Grammar (PSCFG) systems, built using the WASP^{-1} method [Wong and Mooney, 2007]. The WASP^{-1} training process takes a grammar of the meaning representation language and a parallel data set of meaning representations and output sentences, and produces a target language model and weighted SCFG of the MR grammar. This is done in a two-step process, first producing a non-probabalistic SCFG and then subsequently finding probabilities for the SCFG rules using the training data; the end result is the weighted PSCFG. To generate, it takes a meaning representation, and finds a sentence that maximizes the conditional probability of the sentence given the meaning representation according to the trained PSCFG. That conditional probability is hard to model due to the difficulty in determining if the output sentences are grammatical [Wong, 2007]; thus instead of directly using that, WASP^{-1} uses a language model combined with a translation model to determine if its generated output is grammatical and also corresponds to the meaning of the original input.

Prodigy-METEO distributes outputs from two different PSCFG-based systems.

PSCFG-unstructured: This approach uses the same input as the PBSMT-unstructured and PCFG systems. The input is encoded as a MR grammar, and then is given to WASP^{-1}, which uses that to train the SCFGs used for generation.

PSCFG-semantic: This approach uses a different input grammar, augmenting the grammar used for the PSCFG-unstructured system with recursive predicate-argument structure [Belz and Kow, 2009]. This structure resembles semantic forms, and was produced from the raw input vectors of wind data.

Other systems

SUMTIME-hybrid: This approach is the original, hand-crafted, SUMTIME system [Reiter et al., 2005]. It is a deterministic rule-based system that generates output from content representations via microplanning and realization. This system is a more traditional language generation approach, using skilled experts to craft generation rules.

Natural corpus: Though not truly a language generation system in the sense of the others, the natural, human-written corpus can also be considered as a language generation approach, albeit an expensive one with high latency. However, comparing against a human standard can be helpful, so we also included these "generations" as well.

5.7.4 Evaluation

The earlier results in this domain from Belz and Kow [2009] report NIST and BLEU scores for automatic measues. Thus, we report those in our evaluation, plus METEOR which we feel is more likely to be correlated to human judgements (see Section 3.1.1).

As described in Section 5.7.2, the Prodigy-METEO corpus can be split into overlapping folds for cross-validation, which we have done. This necessitates training (and then testing) what amounts to 5 separate systems; the average of scores from all 5 systems will give a reasonably unbiased view of system performance. Table 5.8 shows the results of all 3 automatic measures for MOUNTAIN output on each fold of the corpus.

Not much can be said about those scores in isolation, however, so a comparison to other systems would be helpful. The Prodigy-METEO distribution includes the outputs from the 10 systems evaluated by Belz and Kow [2009], making direct comparison of scores a straightforward task. Table 5.9 shows the results of multiple different generation systems in the METEO domain; these scores are the

Fold	# of texts	NIST	BLEU	METEOR
1	48	6.6767	.5933	.8054
2	40	6.2034	.5567	.7594
3	46	6.5037	.5773	.7735
4	44	6.5914	.5793	.7685
5	43	6.6113	.5801	.7757
Average	44	6.5173	.5773	.7765

Table 5.8: Results of automatic measures for MOUNTAIN output in the METEO domain using 5-fold cross-validation.

System	NIST	BLEU	METEOR
corpus	9.3074	1.000	1.000
PSCFG-semantic	7.0725	.6352	.8159
PSCFG-unstructured	6.8815	.6259	.8081
PCFG-greedy	6.6325	.5968	.7969
MOUNTAIN	6.5173	.5773	.7765
SUMTIME-hybrid	6.0865	.5252	.6889
PCFG-2gram	5.4687	.4862	.6867
PCFG-viterbi	5.4624	.4864	.6863
PCFG-roulette	5.9056	.4617	.6853
PBSMT-unstructured	5.6846	.4863	.4877
PBSMT-structured	4.1860	.3143	.4173
PCFG-random	3.3683	.2069	.2102

Table 5.9: Results of automatic measures for MOUNTAIN compared to the NLG systems and original human-written forecasts used in Belz and Kow [2009]. Reported values are averaged from the 5-fold cross-validation test sets.

average of the 5 folds in the cross-validation setup. Overall, MOUNTAIN performs fairly well according to automatic measures compared to other systems. outperforming other SMT-based approaches, many of the weaker CFG-based systems, as well as the hand-crafted human-designed SUMTIME-hybrid system. The only systems which are clearly better than MOUNTAIN are the PSCFG-based approaches [Wong and Mooney, 2007]; these approaches are not fully automatically trained like MOUNTAIN is, however. All three of the automatic metrics are fairly consistent in how systems are ranked relative to each other.

5.7.5 Do Automatic Measures Correlate with Ratings by Humans?

One of the key issues with NLG evaluation is its expense, primarily due to the high costs of human-based evaluation. Automatic measures are cheap and simple to use, but it isn't clear that they measure the same things. Thus, it would be helpful to determine what correlation, if any, there is for these automatic measures and human judgements.

We set up the same human evaluation used by Belz and Kow [2009]; human evaluators were shown forecast texts and asked to rate them on a 7-point Likert scale for both clarity and readability. These terms were explicitly defined for the evaluators: clarity refers to how clear and understandable a forecast is, and readability refers to how fluent and easy to read a forecast is. In addition to the MOUNTAIN approach, we used 5 systems from the earlier evaluation: the SMT-based systems, the most consistent CFG-based system (PSCFG-semantic), the handcrafted SUMTIME-hybrid system, and the original human-written corpus. The choice of these systems was made because they were either similar to the MOUNTAIN approach, strong performers in the previous evaluation, or a human-generated standard; additionally they encompass widely varying approaches, from manually annotated to fully automatic. They also cover the entire range of systems in the earlier evaluation, from highly- to poorly-performing, so our results should be easy to compare. Table 5.10 shows an example of a METEO forecast text from each of these systems.

We used 12 randomly selected forecast dates (taken from each fold of the corpus), and included outputs from all 6 systems. Thus, our evaluation set consists of 72 distinct forecast texts. Using a repeated Latin square design, raters were presented with 2 different forecasts texts from each system, where the systems were presented in random order. Because earlier work in this domain demonstrated that non-experts produced ratings that were highly correlated to those given by domain experts [Belz and Reiter, 2006], we did not attempt to find weather experts to rate the forecasts. In fact, we used Mechanical Turk workers to provide the ratings. MTurk workers were randomly assigned a system order from the Latin square, then shown only the forecast texts and told to rate them for clarity and readability as defined above. We had a total of 38 raters complete this task.

Raw Input	`[[1,SW,38,42,55,-,0600],[2,-,45,50,65,-,0900],[3,WSW,-,-,-,-,0000]]`
corpus	`SW 38-42 GUSTS 55 SOON INCREASING 45-50 GUSTS 65 FOR A TIME THEN VEERING WSW LATE EVENING`
SUMTIME-hybrid	`SW 38-42 GUSTS 55 SOON INCREASING 45-50 GUSTS 65 THEN VEERING WSW 38-42 BY MIDNIGHT`
PSCFG-semantic	`SW 38-42 GUSTS 55 SOON INCREASING 45-50 GUSTS 65 THEN VEERING WSW LATER`
PBSMT-struct	`GUSTS SW 38-42 GUSTS 55 TO AND INCREASING BY GUSTS 45 GUSTS 50 GUSTS 65 BY NINE TO THEN 1 VEERING WSW BY EVENING LATER`
PBSMT-unstruct	`BACKING SW 38-42 GUSTS 55 GRADUALLY INCREASING 45-50 GUSTS 65 BY MIDDAY VEERING WSW BY LATE EVENING`
MOUNTAIN	`SW 38-42 GUSTS 55 INCREASING 45-50 GUSTS 65 BY MIDDAY THEN WSW BY LATE EVENING`

Table 5.10: Examples from the 8 Sep 2000 forecast in the METEO domain. Shown is the raw input vector of 7-tuples, and the corresponding outputs from the systems used in the evaluation, including the human-written corpus.

Results

Figure 5.7 shows the mean clarity and readability scores for each system. Though it matches the results from Belz and Kow [2009], we are still surprised that the original human corpus is rated poorer than most of the machine-generated texts. The handcrafted SUMTIME system has the highest ratings in both categories, though it is by far the most expensive system in terms of creation effort. The semi-automatic PSCFG system from Wong and Mooney [2007] is the only system with a higher-rated clarity than readability, though this is not statistically significant. The MOUNTAIN approach is significantly better than the other SMT-based systems, and in fact is rated slightly higher than the natural corpus. It is not quite as good as the SUMTIME or PSCFG systems; however, it should be noted that those systems include linguistic knowledge and are not fully automatic like MOUNTAIN. If MOUNTAIN can exploit that as well, it is likely to see a performance improvement.

Figures 5.8 and 5.9 show the frequency of each clarity and readability rating, respectively, for all 6 test systems. These graphs make it clear why the PBSMT-structured system scored so poorly – it has the highest proportion of ratings of 1 or 2 of any system as well as the lowest proportion of 7s. Conversely, it is also clear

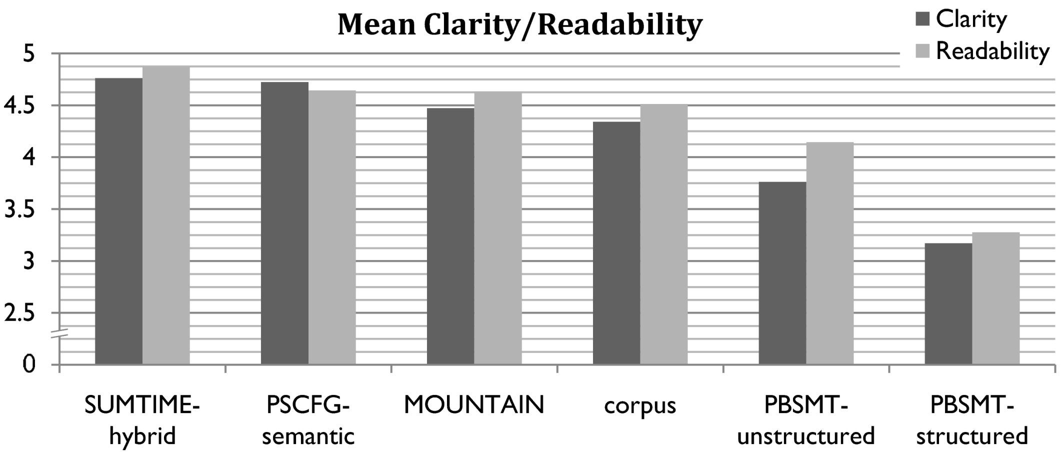


Figure 5.7: Mean clarity and readability ratings from human evaluation of 5 NLG systems and the original human-written forecasts in the METEO domain.

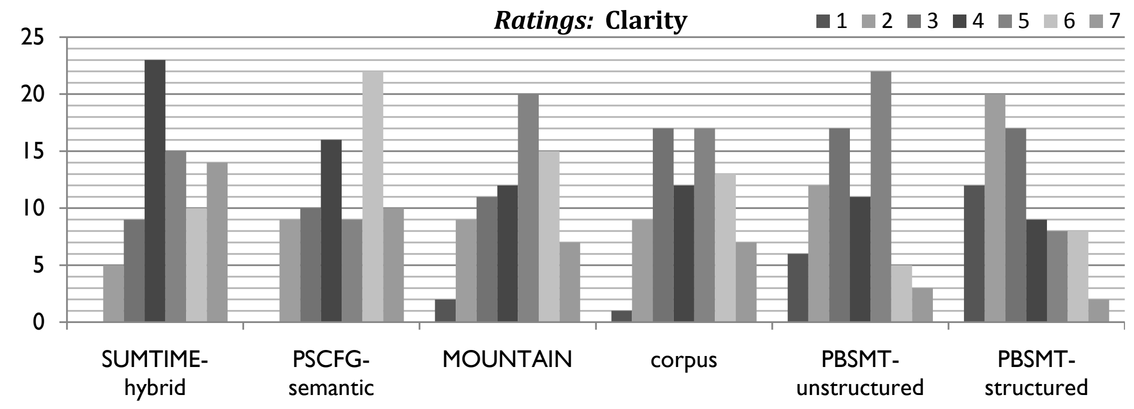


Figure 5.8: Histogram of clarity ratings (1–7) for each of the 6 systems from the human evaluation.

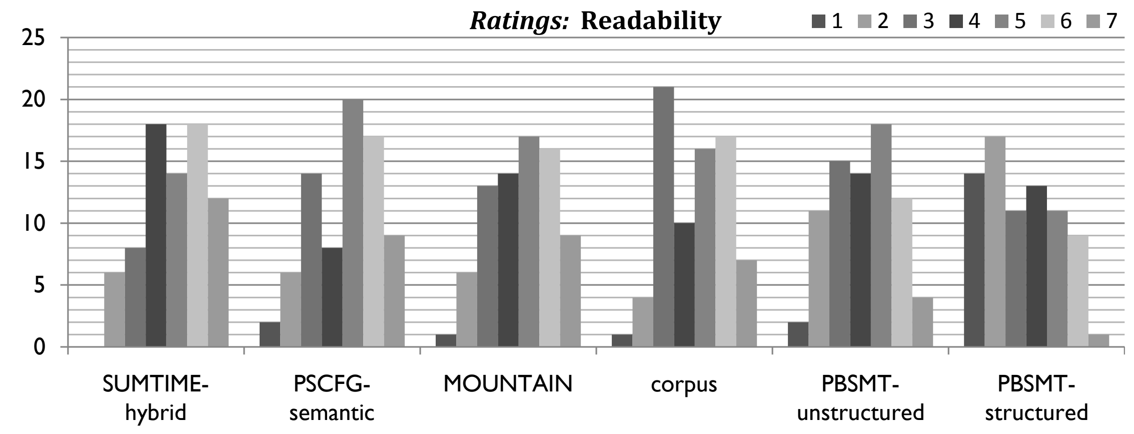


Figure 5.9: Histogram of readability ratings (1–7) for each of the 6 systems from the human evaluation.

why the SUMTIME system scored very well, since it has not a single rating of 1 in either category. The MOUNTAIN output has a rating pattern similar to that of the corpus text, which is not overly surprising since that is its source text. However, its mean score appears to be higher due to having additional ratings of 5 that were 3s for the corpus. The main difference between MOUNTAIN and the better-rated systems ahead of it appears to be fewer 7 ratings and some 1s.

Table 5.11 shows both automatic and human-based scores for all the systems based on the 12 forecast texts used in the human evaluation. This is a different test set than reported previously (see Table 5.9), which explains the different values for the automatic metrics. Ignoring the "corpus" system, which is guaranteed to have a perfect score from the automatic measures, there is definite similarity in rankings between the human and automatic scores – except for the handcrafted SUMTIME-hybrid system. Just as in the previous evaluation, this system was not among the better systems according to the automatic metrics, but was given the highest scores from the human evaluators. However, we did not see the same strong performance of the PBSMT-unstructured system on the automatic measures, as our results showed it to be similarly ranked by both the human and automatic metrics.

As for MOUNTAIN, these results show it to be significantly better than the other SMT-based approaches, according to all of the metrics. It is not immediately clear why MOUNTAIN is so much better. Further, it is comparable to or better than the original corpus according to the human raters, which is a positive but somewhat puzzling result. Our best hypothesis for this is that MOUNTAIN regularizes the output from the corpus, making its forecasts' language similar from one to another, while the corpus is written by several different human forecasters who have distinct writing styles and might not get identical ratings. Additionally, the other trained

System	Clarity	Readability	NIST	BLEU	METEOR
SUMTIME-hybrid	4.7632	4.8684	5.3174	.5548	.6840
PSCFG-semantic	4.7237	4.6447	5.7959	.5876	.7967
MOUNTAIN	4.4737	4.6316	5.6724	.5860	.7784
corpus	4.3421	4.5132	7.4103	1.000	1.000
PBSMT-unstructured	3.7632	4.1447	4.8723	.4619	.4834
PBSMT-structured	3.1711	3.2763	3.4419	.3088	.1623

Table 5.11: Mean human-based scores and automatic measures for 5 NLG systems and the original human-written forecasts. Each system used an identical 12 dates of forecast texts from the METEO domain.

system (PSCFG-semantic) was rated as having the same readability and somewhat better clarity than MOUNTAIN. As we mentioned above, though, that system is not fully automatic like MOUNTAIN, and also makes use of information MOUNTAIN does not; if MOUNTAIN *did* use that information it is likely to improve its output. Finally, though the handcrafted SUMTIME-hybrid system has clearly better output, MOUNTAIN has an even clearer edge in creation effort; the SUMTIME system was reported to take a year to build, whereas MOUNTAIN was able to train a new system in this domain in less than a week.

Our results in this evaluation showed high correlation between the automatic measures and the human-based measures. Pearson's $r > 0.9$ for all comparisons, with BLEU and METEOR being generally more highly correlated than NIST for both clarity and readability. The correlation coefficients for each comparison are shown in Table 5.12. Spearman's rank correlation $\rho = 0.7$, between all of the metrics and both clarity and readability, since each metric ordered all 5 systems identically and the human-based scores were also ranked identically in both dimensions.

However, it is not clear that the automatic metrics are designed to measure the same thing that the human evaluation did. Though clarity and readability are important when considering NLG quality, it seems that the automatic measures are actually measuring *adequacy*, or how well a particular example says what a reference does. In fact, this is something that Stent et al. [2005] suggested based on their results of comparing different evaluation metrics – the automatic measures are more highly correlated to NLG output adequacy than other dimensions.

Adequacy is somewhat more challenging to test well in a human evaluation, though. The best method is to show the original data used to generate an example

	NIST	BLEU	METEOR
Clarity	.926	.957	.953
Readability	.944	.966	.951

Table 5.12: Pearson's correlation for human-based clarity and readability scores versus the automatic measures.

– in the METEO case, that would be the raw input – as well as a machine-generated text, and ask the human to rate how well the text communicates the information from the input. The drawback here is that it often requires some expertise to be able to tell if an output sentence is particularly good for a given input. The alternative is to present a human-written example instead of the input data, which requires less expertise for evaluators but can bias the perception of the machine-generated output.

5.8 Discussion

MOUNTAIN clearly outperforms the other PBSMT systems from Belz and Kow [2009], though it is not immediately obvious why that would be. The most likely difference is in the design and tokenization of the internal language; the Belz and Kow systems used "simply the augmented corpus input vectors", and also tried tagging the vectors with structure information. With the expected use of MOUNTAIN, the system designer has the ability to control the structure and vocabulary of the internal language, which corresponds to underlying structural information (like dialog state). There are likely to be multiple valid and reasonable representations of that information, but not all of them are guaranteed to be equally suitable as input for statistical machine translation. For example, in the METEO data, we took the input language to be the raw vector of weather data, with some basic tokenization. Our tokenization left the "–" tokens in the internal language training sentences, even though they do not appear to convey useful information in the English translation. It is possible that not including them could result in a better translation model by reducing the complexity of the input language. Optimally specifying an input language for a given application is likely to be a challenging problem; finding an automatic method that can produce reasonable options would help in getting the most out of this language generation approach.

Compared to other NLG systems, MOUNTAIN requires relatively little time to set up. The largest and most expensive part is corpus collection; once the training corpus is available, the training time itself is minimal. For the test application described in Section 5.4, it was about 20 seconds, though a real application would clearly require a larger training corpus. Still, for well-defined, and mostly closed-vocabulary tasks – which most spoken dialog systems could be described as – the training time should be measurable in minutes. The time required to generate output is nearly instantaneous, or similar to a template-based system. Obtaining a suitable training corpus is still expensive, though there are potential solutions. Besides including responses from the system developers, which is similar in cost and skill to template-writing, other data sources such as transcribed Wizard-of-Oz interactions could be used. Potentially, any available human-human dialogs for the application domain could also be included in the training corpus, as long as they could be transcribed and annotated with the internal language.

Some idea of how much training data is needed for reasonable output from MOUNTAIN would be useful, as that would permit cost analysis to be done on the most expensive part of the training process. Though we recognize that there are likely application- or domain-specific issues that will impact the training data requirements, we wanted to see if we could gather any information about this problem in general. To test this, we halved the amount of training data from the METEO domain, using only about 200 training examples in each fold instead of more than 400. Repeating the 5-fold cross-validation technique used earlier, we computed the same automatic measures as in Section 5.7 for this smaller training set. The results, compared to the earlier system trained on the full METEO corpus, are shown in Table 5.13. The reduction in training data clearly causes MOUNTAIN to perform more poorly. However, note that the scores from NIST and METEOR would only drop MOUNTAIN one spot in the results from Table 5.9, still significantly outperforming the SMT-based approaches as well as most of the CFG-based approaches. BLEU

System	NIST	BLEU	METEOR
Full-data	6.5173	.5773	.7765
Half-data	5.7076	.4636	.7008

Table 5.13: Results of automatic measures for MOUNTAIN after halving the training data in the METEO domain.

differs greatly from the other measures, and would rank this system among the worst 3.

As described in Section 5.3, the output of MOUNTAIN discussed in this work was the single-best result from the translation engine. Though this can result in reasonable generation for many inputs, it also means that an input will always have the same generated response, which will have the same repetitiveness problems as template-based NLG. However, Moses can be configured to output N-best lists of translation results, rather than a single-best result. If MOUNTAIN instead selects its responses from the N-best list, that will provide more variation in the resulting output, and possibly prevent a human listener from perceiving it as unnaturally repetitive. Further investigation of this, including determining an appropriate size for the list and evaluating user perception of the generation, is planned.

5.9 Summary

This chapter described the MOUNTAIN language generation system, a fully-automatic system which uses machine translation techniques to generate novel natural language examples using a corpus. We detailed the training process required to use MOUNTAIN, and showed results from evaluations in two different domains. Further, we compared MOUNTAIN output to several different approaches, and found that it outperformed other fully-automatic trainable systems. MOUNTAIN also was shown to approach the performance of other advanced systems, including a handcrafted system designed especially for the testing domain. Based on these results, we are confident that if MOUNTAIN can be further improved, particularly to make use of further information other advanced systems use, its performance will equal or surpass those systems.

Chapter 6

Concluding Words

6.1 Summary

This thesis investigated methods of creating machine-generated human-like natural language. We discussed several aspects of this problem, including making speech synthesizers sound more natural (Section 2.1) and the applicability to spoken language generation (Section 2.2). We discussed the various dimensions of "human-like" natural language, including understandability and naturalness. We have also described a framework designed to produce more understandable spoken language: uGloss (Chapter 4). Additionally, we discussed the major issues with evaluation of natural language generation (Chapter 3), such as the tradeoffs between automatic and human-based measures, and the importance of accounting for motivation of the evaluators when doing a human-based evaluation. Finally, we have also detailed the MOUNTAIN language generation system (Chapter 5), a fully-automatic, data-driven approach that uses statistical machine translation techniques to generate human-like language.

The two systems described in this thesis – uGloss and MOUNTAIN – are both, at their core, approaches for improved language generation that is more human-like in quality. Human-like language, for our purposes, refers to spoken utterances comparable in understandability, clarity, naturalness, and efficiency to human-generated language. The uGloss approach is more tightly focused, emphasizing understandable utterances – particularly utterances spoken with speech synthesis – when considering challenging inputs that even people can find difficult to speak understandably. uGloss has a direct impact on the types of utterances to be generated, as it is designed to use heuristics and other learned rules as part of the tactical generation process. Because the intent of this framework is to achieve human-like understandability, the process does not always result in fluent, natural-sounding utterances. This means uGloss will not always generate completely human-like language in order to fulfill its goals.

In contrast, the MOUNTAIN language generation system is able to generate language that is human-like in many dimensions, not only understandability. MOUNTAIN tries to sidestep the issue of how to get a machine to have sufficient knowledge to generate highly natural and understandable language by utilizing human-generated corpora to do tactical generation. Rather than directly learning characteristics of human-like utterances, MOUNTAIN uses models trained from the examples in its training corpora to generate utterances. The resulting generations will implicitly be natural and understandable – or at least as much as the examples in the corpora themselves are. In this way, MOUNTAIN can fully-automatically generate comparable output to humans, without needing to learn specific rules for how to do that. The major advantages over the uGloss approach are that this method is fully automatic once the training corpus is collected for a particular application, and the naturalness and understandability of the corpus itself is reflected in the generations MOUNTAIN will produce.

Results from evaluations in multiple domains show the MOUNTAIN approach is capable of producing output of similar quality to human-written texts.A previous evaluation conducted by another group in the Prodigy-METEO domain examined four general types of systems (handcrafted rule-based, probabilistic context-free grammars, probabilistic synchronous context-free grammars, and phrase-based statistical machine translation), with some types having multiple implementation variants. Of these, the PBSMT approaches were generally among the poorer systems according to both automatic and human-based measures. The results of our evaluation from this domain showed that MOUNTAIN, which is also a PBSMT-based system, can have performance remarkably close to the handcrafted and PSCFG systems, though still not quite as good. Results from human evaluators showed that generated output from MOUNTAIN has comparable clarity and readability to human-generated output and the PSCFG-based systems. However, the MOUNTAIN approach is fully automatic, whereas the other systems included some amount of manually-encoded linguistic knowledge to achieve their output quality. The fully automatic approach is appealing due to its lower cost of implementation and maintenance, while still generating human-like utterances. Additionally, MOUNTAIN is clearly better than previous PBSMT-based systems in this domain.

6.2 Contributions

This thesis provides the following main contributions:

- A statistical, data-driven, fully-automatic method of tactical language generation. Our method performs similarly to other advanced approaches and has generation quality similar to human-written text in multiple dimensions.
- Demonstration that natural corpora can be used to generate novel natural language sentences that are as clear and understandable as human-written texts.
- A framework for producing more understandable language that will ultimately be spoken by a speech synthesizer.
- An investigation of evaluation methods – both automatic and human-based – for natural language generation output, and what these methods can tell us about output quality.

Bibliography

H. Ai, A. Raux, D. Bohus, M. Eskenazi, and D. Litman. Comparing spoken dialog corpora collected with recruited subjects versus real users. In *8th SIGDial Workshop on Dialogue and Discourse*, Antwerp, Belgium, 2007.

H. Alshawi, S. Bangalore, and S. Douglas. Automatic acquisition of hierarchical transduction models for machine translation. In *36th Annual Meeting of the Association for Computational Linguistics*, Montréal, Canada, 1998.

A. D. Baddeley. The magical number seven: Still magic after all these years? *Psychological Review*, 101:353–356, 1994.

A. D. Baddeley. *Essentials of Human Memory*. Psychology Press, 1999.

A. D. Baddeley, N. Thomson, and M. Buchanan. Word length and the structure of short-term memory. *Journal of Verbal Learning and Verbal Behavior*, 14:575–589, 1975.

Satanjeev Banerjee and Alon Lavie. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *ACL 2005 Workshop on Intrinsic and Extrinsic Evaluation Measures for MT and/or Summarization*, Ann Arbor, MI, 2005.

S. Bangalore, O. Rambow, and S. Whittaker. Evaluation metrics for generation. In *INLG 2000*, Mitzpe Ramon, Israel, 2000.

Michele Banko, Vibhu O. Mittal, and Michael J. Witbrock. Headline generation based on statistical translation. In *ACL 2000*, pages 318–325, Hong Kong, 2000.

Anja Belz. Statistical generation: Three methods compared and evaluated. In *ENLG 2005*, pages 15–23, Aberdeen, Scotland, 2005.

Anja Belz. Automatic generation of weather forecast texts using comprehensive probabalistic generation-speace models. *Natural Language Engineering*, 14(4): 431–455, 2008.

Anja Belz. Prodigy-METEO: Pre-alpha release notes (Nov 2009). Technical Report NLTG-09-01, Natural Language Technology Group, CMIS, University of Brighton, 2009. `http://www.nltg.brighton.ac.uk/home/Anja.Belz/Publications/release-notes-pre-alpha.pdf`.

Anja Belz and Eric Kow. System building cost vs. output quality in data-to-text generation. In *ENLG 2009*, Athens, Greece, 2009.

Anja Belz and Ehud Reiter. Comparing automatic and human evaluation of NLG systems. In *11th Conference of the European Chapter of the Association for Computational Linguistics (EACL 06)*, Trento, Italy, 2006.

Anja Belz, Eric Kow, Jette Viethen, and Albert Gatt. The GREC Challenge 2008: Overview and evaluation results. In *INLG 2008*, Salt Fork, OH, 2008.

C. Bennett and A. Black. The Blizzard Challenge 2006. In *Blizzard Challenge 2006*, Pittsburgh, PA, 2006.

A. Black. Perfect synthesis for all of the people all of the time. In *IEEE 2002 Workshop on Speech Synthesis*, Santa Monica, CA, 2002.

A. Black and K. Lenzo. Limited domain synthesis. In *ICSLP2000*, volume II, pages 411–414, Beijing, China, 2000.

A. Black and P. Taylor. Automatically clustering similar units for unit selection in speech synthesis. In *Eurospeech97*, volume 2, pages 601–604, Rhodes, Greece, 1997.

A. Black and K. Tokuda. Blizzard Challenge – 2005: Evaluating corpus-based speech synthesis on common datasets. In *Interspeech 2005*, Lisbon, Portugal, 2005.

A. Black, P. Taylor, and R. Caley. The Festival speech synthesis system. http://festvox.org/festival, 1998.

D. Bohus, A. Raux, T. Harris, M. Eskenazi, and A. Rudnicky. Olympus: An open-source framework for conversational spoken language interface research. In *HLT-NAACL 2007 workshop on Bridging the Gap: Academic and Industrial Research in Dialog Technology*, 2007.

C. Boidin, V. Rieser, L. van der Plas, O. Lemon, and J. Chevelu. Predicting how it sounds: Re-ranking dialogue prompts based on TTS quality for adaptive spoken dialogue systems. In *Interspeech Special Session: Machine Learning for Adaptivity in Spoken Dialogue*, Brighton, UK, 2009.

W. A. Bousfield, G. A. Whitmarsh, and J. Esterton. Serial position effects and the "Marbe effect" in the free recall of meaningful words. *Journal of General Psychology*, 59:255–262, 1958.

I. Bulyko and M. Ostendorf. Efficient integrated response generation from multiple targets using weighted finite state transducers. *Computer Speech and Language*, 16:533–550, 2002.

C. Callaway. Evaluating coverage for large symbolic NLG grammars. In *18th International Joint Conference on Artificial Intelligence (IJCAI 2003)*, Acapulco, Mexico, 2003.

Nathanael Chambers and James Allen. Stochastic language generation in a dialogue system: Toward a domain independent generator. In *5th SIGdial Workshop on Discourse and Dialogue*, pages 9–18, Cambridge, MA, USA, 2004.

David L. Chen and Raymond J. Mooney. Learning to sportscast: A test of grounded language acquisition. In *25th International Conference on Machine Learning (ICML 2008)*, Helsinki, Finland, 2008.

S. Choularton and R. Dale. User responses to speech recognition errors: Consistency of behaviour across domains. In *SST-2004*, Sydney, Australia, 2004.

Robert A. J. Clark, Monika Podsaidlo, Mark Fraser, Catherine Mayo, and Simon King. Statistical analysis of the Blizzard Challenge 2007 listening test results. In *Blizzard Challenge 2007*, Bonn, Germany, 2007.

Simon Corston-Oliver, Michael Gamon, Eric Ringger, and Robert Moore. An overview of Amalgam: A machine-learned generation module. In *INLG 2002*, pages 33–40, New York, 2002.

R. Dale. *Generating Referring Expressions*. MIT Press, 1992.

R. Dale and E. Reiter. Computational interpretations of the gricean maxims in the generation of referring expressions. *Cognitive Science*, 19:233–263, 1995.

Robert Dale and Chris Mellish. Towards evaluation in natural language generation. In *Proceedings First International Conference on Language Resources and Evaluation*, pages 555–562, 1998.

Hoa Trang Dang. DUC 2005: Evaluation of question-focused summarization systems. In *Proceedings of the Workshop on Task-Focused Summarization and Question Answering*, Sydney, Australia, July 2006. Association for Computational Linguistics.

J. R. Davis and J. Hirschberg. Assinging intonational features in synthesized spoken directions. In *ACL 1988*, Buffalo, NY, 1988.

G. Doddington. Automatic evaluation of machine translation quality using n-gram co-occurance statistics. In *ARPA Workshop on Human Language Technology*, 2002.

Bonnie Dorr, Christof Monz, Stacy President, Richard Schwartz, and David Zajic. A methodology for extrinsic evaluation of text summarization: Does ROUGE correlate? In *ACL 2005*, Ann Arbor, MI, 2005.

J. Edlund, M. Heldner, and J. Gustafson. Two faces of spoken dialogue systems. In *Interspeech 2006 Dialogue on Dialogues Workshop*, Pittsburgh, PA, 2006.

J. Edlund, J. Gustafson, M. Heldner, and A. Hjalmarsson. Towards human-like spoken dialogue systems. *Speech Communication*, 50(8-9):630 – 645, 2008.

E. Eide, A. Aaron, R. Bakis, W. Hamza, M. Picheny, and J. Pitrelli. A corpus-based approach to <AHEM/> expressive speech synthesis. In *5th ISCA Workshop on Speech Synthesis*, Pittsburgh, PA, 2004.

M. Fleischman and E. Hovy. Towards emotional variation in speech-based natural language generation. In *INLG 2002*, New York, 2002.

Mark Fraser and Simon King. The Blizzard Challenge 2007. In *Blizzard Challenge 2007*, Bonn, Germany, 2007.

H. P. Grice. Logic and conversation. In Cole and Morgan, editors, *Syntax and Semantics: Speech Acts*, volume 3. Academic Press, 1975.

J. Hitzeman, A. Black, C. Mellish, J. Oberlander, and P. Taylor. Use of automatically generated discourse-level information in a concept-to-speech synthesis system. In *ICSLP98*, Sydney, Australia, 1998.

H. Horacek. Text generation methods for dialog systems. In *2003 AAAI Spring Symposium*, pages 52–54, Palo Alto, CA, 2003.

A. Hunt and A. Black. Unit selection in a concatenative speech synthesis system using a large speech database. In *ICASSP96*, volume 1, pages 373–376, Atlanta, GA, 1996.

A. Iida, N. Campbell, F. Higuchi, and M. Yasumura. A corpus-based speech synthesis system with emotion. *Speech Communication*, 40:161–187, 2003.

D. M. Jones. The 7$\pm$2 urban legend. In *MISRA C 2002 Conference*, Oct. 2002. URL `www.knosof.co.uk/cbook/misart.pdf`.

Vasilis Karaiskos, Simon King, Robert A. J. Clark, and Catherine Mayo. The Blizzard Challenge 2008. In *Blizzard Challenge 2008*, Brisbane, Australia, 2008.

Simon King and Vasilis Karaiskos. The Blizzard Challenge 2009. In *Blizzard Challenge 2009*, Edinburgh, UK, 2009.

Aniket Kittur, Ed H. Chi, and Bongwon Suh. Crowdsourcing user studies with Mechanical Turk. In *CHI '08: Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, pages 453–456, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-011-1. doi: http://doi.acm.org/10.1145/1357054.1357127.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. Moses: open source toolkit for statistical machine translation. In *ACL 2007: Interactive Poster and Demonstration Sessions*, pages 177–180, Prague, Czech Republic, 2007.

H. L. Lane and B. Tranel. The Lombard sign and the role of hearing in speech. *Journal of Speech and Hearing Research*, 14:677–709, 1971.

Irene Langkilde and Kevin Knight. Generation that exploits corpus-based statistical knowledge. In *ACL/ICCL 1998*, pages 704–710, Montreal, Quebec, Canada, 1998.

B. Langner and A. Black. Creating a database of speech in noise for unit selection synthesis. In *5th ISCA Workshop on Speech Synthesis*, Pittsburgh, PA, 2004a.

B. Langner and A. Black. An examination of speech in noise and its effect on understandability for natural and synthetic speech. Technical Report CMU-LTI-04-187, Language Technologies Institute, Carnegie Mellon University, Pittsburgh, PA, 2004b.

B. Langner and A. Black. Improving the understandability of speech synthesis by modeling speech in noise. In *ICASSP05*, Philadelphia, PA, 2005a.

B. Langner and A. Black. Using speech in noise to improve understandability for elderly listeners. In *ASRU 2005*, San Juan, Puerto Rico, 2005b.

B. Langner and A. Black. uGloss: A framework for improving spoken language generation understandability. In *Interspeech 2007*, Antwerp, Belgium, 2007.

B. Langner, R. Kumar, A. Chan, L. Gu, and A. Black. Generating time-constrained audio presentations of structured information. In *Interspeech 2006*, Pittsburgh, PA, 2006.

D. C. LeCompte. 3.14159, 42, and 7±2: Three Numbers That (Should) Have Nothing To Do With User Interface Design. `http://www.internettg.org/newsletter/aug00/article_miller.html` , 2000.

Oliver Lemon and Olivier Pietquin. Machine learning for spoken dialogue systems. In *Interspeech 2007*, Antwerp, Belgium, 2007.

J. Lester and B. Porter. Developing and empirically evaluating robust explanation generators: The KNIGHT experiments. *Computational Linguistics*, 23(1):65–101, 1997.

R. Likert. A technique for the measurement of attitudes. *Archives of Psychology*, 140:1–55, 1932.

Chin-Ye Lin and Eduard Hovy. Automatic evaluation of summaries using n-gram co-occurance statistics. In *HLT-NAACL 2003*, Edmonton, Canada, 2003.

Étienne Lombard. Le signe de l'elévation de la voix. *Annales Maladiers Oreille, Larynx, Nez, Pharynx*, 37:101–119, 1911.

Tomasz Marciniak and Michael Strube. Using an annotated corpus as a knowledge source for language generation. In *Workshop on Using Corpora for Natural Language Generation*, Birmingham, UK, 2005.

E. Marsi. *Intonation in Spoken Language Generation*. PhD thesis, Netherlands Graduate School of Linguistics, 2001.

S. Maskey and J. Hirschberg. Automatic summarization of broadcast news using structural features. In *Eurospeech 2003*, Geneva, Switzerland, 2003.

S. Maskey and J. Hirschberg. Comparing lexical, acoustic/prosodic, structural and discourse features for speech summarization. In *Interspeech 2005*, Lisbon, Portugal, 2005.

K. McKeown, R. Passonneau, D. Elson, A. Nenkova, and J. Hirschberg. Do summaries help? a task-based evaluation of multi-document summarization. In *SIGIR 2005*, Salvador, Brazil, 2005.

J. Meron. Prosodic unit selection unit an intonation speech database. In *4th ISCA Workshop on Speech Synthesis*, Pitlochry, Scotland, 2001.

G. A. Miller. The magical number seven plus or minus two: Some limits on our capacity for processing information. *Psychological Review*, 63:81–97, 1956.

B. B. Murdock. The serial position effect of free recall. *Journal of Experimental Psychology*, 64:482–488, 1962.

C. Nakatsu and M. White. Learning to say it well: Reranking realizations by predicted synthesis quality. In *ACL 2006*, Sydney, Australia, 2006.

Franz Josef Och and Hermann Ney. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51, 2003.

A. Oh and A. Rudnicky. Stochastic language generation for spoken dialogue systems. In *ANLP/NAACL 2000 Workshop on Conversational Systems*, pages 27–32, Seattle, WA, 2000.

S. Pan and K. McKeown. Spoken language generation in a multimedia system. In *ICSLP96*, volume 1, Philadelphia, PA, 1996.

S. Pan, K. McKeown, and J. Hirschberg. Exploring features from natural language generation for prosody modeling. *Computer Speech and Language*, 16:457–490, 2002.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: A method for automatic evaluation of machine translation. In *ACL 2002*, pages 311–318, Philadelphia, PA, 2002.

C. Paris, D. Scott, N. Green, K. McCoy, and D. McDonald. Desiderata for evaluation of natural language generation. In M. White and R. Dale, editors, *Workshop Final Report*, chapter 2, pages 9–16. Workshop on Shared Tasks and Comparative Evaluation in Natural Language Generation, 2007.

A. Popescu-Belis. Evaluation of NLG: Some analogies and differences with machine translation and reference resolution. In *Machine Translation Summit XI*, Copenhagen, Denmark, 2007.

S. Prevost. An information structural approach to spoken language generation. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, pages 294–301, Santa Cruz, CA, 1996.

S. Prevost and M. Steedman. Specifying intonation from context for speech synthesis. *Speech Communication*, 15:139–153, 1994.

C. Quirk, C. Brockett, and W. Dolan. Monolingual machine translation for paraphrase generation. In *EMNLP 2004*, Barcelona, Spain, 2004.

Owen Rambow, Srinivas Bangalore, and Marilyn Walker. Natural language generation in dialog systems. In *HLT 2001*, pages 1–4, San Diego, CA, 2001.

Adwait Ratnaparkhi. Trainable methods for surface natural language generation. In *NAACL 2000*, pages 194–201, Seattle, WA, 2000.

A. Raux and A. Black. A unit selection approach to F0 modeling and its application to emphasis. In *ASRU2003*, St Thomas, USVI, 2003.

A. Raux, D. Bohus, B. Langner, A. Black, and M. Eskenazi. Doing research on a deployed spoken dialogue system: One year of Let's Go! experience. In *Interspeech 2006*, Pittsburgh, PA, 2006.

E. Reiter, S. Sripada, J. Hunter, J. Yu, and I. Davy. Choosing words in computer-generated weather forecasts. *Artificial Intelligence*, 67:137–169, 2005.

Ehud Reiter and Anja Belz. An investigation into the validity of some metrics for automatically evaluating natural language generation systems. *Computational Linguistics*, 35(4):529–558, 2009.

Ehud Reiter and Robert Dale. Building applied natural language generation systems. *Journal of Natural Language Engineering*, 3(1):57–87, 1997.

A. Rudnicky, C. Bennett, A. Black, A. Chotimongkol, K. Lenzo, A. Oh, and R. Singh. Task and domain specific modelling in the Carnegie Mellon Communicator system. In *ICSLP2000*, volume II, pages 130–133, Beijing, China, 2000.

V. Rus, Z. Cai, and A. C. Graesser. Evaluation in natural language generation: The question generation task. In *Workshop on Shared Tasks and Comparative Evaluation in Natural Language Generation*, Arlington, VA, 2007.

R. Snow, B. O'Connor, D. Jurafsky, and A. Y. Ng. Cheap and fast – but is it good? Evaluating non-expert annotations for natural language tasks. In *Conference on Empirical Methods in Natural Language Processing (EMNLP 2008)*, Honolulu, HI, 2008.

S. Sripada, E. Reiter, J. Hunter, and J. Yu. SUMTIME-METEO: Parallel corpus of naturally occurring forecast texts and weather data. Technical Report AUCS/TR0201, Computing Science Department, University of Aberdeen, Aberdeen, Scotland, 2002.

S. Sripada, E. Reiter, J. Hunter, and J. Yu. Exploiting a parallel text-data corpus. In *Corpus Linguistics 2003*, Lancaster, UK, 2003.

A. Stent, M. Marge, and M. Singhai. Evaluating evaluation methods for generation in the presence of variation. In *CICLing 2005*, Mexico City, Mexico, 2005.

Andreas Stolcke. SRILM – An extensible language modeling toolkit. In *ICSLP 2002*, pages 901–904, Denver, CO, 2002.

V. Strom, A. Nenkova, R. Clark, Y. Vazquez-Alvarez, J. Brenier, S. King, and D. Jurafsky. Modelling prominence and emphasis improves unit-selection synthesis. In *Interspeech 2007*, Antwerp, Belgium, 2007.

Y. Stylianou, O. Cappé, and E. Moulines. Statistical methods for voice quality transformation. In *Eurospeech95*, pages 447–450, Madrid, Spain, 1995.

T. Toda. *High-Quality and Flexible Speech Synthesis with Segment Selection and Voice Conversion*. PhD thesis, Nara Institute for Science and Technology, 2003.

Alan Turing. Computing machinery and intelligence. *Mind*, 59:433–460, 1950.

S. Varges, F. Weng, and H. Pon-Barry. Interactive question answering and constraint relaxation in spoken dialogue systems. In *7th SIGdial Workshop on Discourse and Dialogue*, Sydney, Australia, 2006.

J. Viethen and R. Dale. Algorithms for generating referring expressions: Do they do what people do? In *INLG 2006*, Sydney, Australia, 2006.

J. Viethen and R. Dale. Evaluation in natural language generation: Lessons from referring expression generation. *Traitment Automatique des Langues*, 48(1):141–160, 2007.

M. Walker, J. Aberdeen, J. Boland, E. Braat, J. Garofolo, L. Hirschman, A. Lee, S. Lee, S. Narayanan, K. Papineni, B. Pellom, J. Polifroni, A. Potamianos, P. Prabhu, A. Rudnicky, G. Sanders, S. Seneff, D. Stollard, and S. Whittaker. DARPA Communicator dialogue travel planning systems: The June 2000 data collection. In *Eurospeech 2001*, Aalborg, Denmark, 2001.

Yuk Wah Wong. *Learning for Semantic Parsing and Natural Language Generation Using Statistical Machine Translation Techniques*. PhD thesis, University of Texas at Austin, 2007.

Yuk Wah Wong and Raymond J. Mooney. Generation by inverting a semantic parser that uses statistical machine translation. In *NAACL-HLT 2007*, pages 172–179, Rochester, NY, 2007.

N. Yankelovich, G. Levow, and M. Marx. Designing SpeechActs: Issues in speech user interfaces. In *Conference on Human Factors in Computing Systems*, Denver, CO, 1995.

Michael Young. Using Grice's maxim of quantity to select the content of plan descriptions. *Artificial Intelligence*, 115:495–535, 1999.

Huayan Zhong and Amanda Stent. Building surface realizers automatically from corpora. In *Workshop on Using Corpora for Natural Language Generation*, Birmingham, UK, 2005.