Improving Speech Recognizer Performance in a Dialog System Using N-best Hypotheses Reranking

by

Ananlada Chotimongkol

Master Student
Language Technologies Institute
School of Computer Science
Carnegie Mellon University

Master Thesis

Committee Members:

Alexander Rudnicky
(Thesis Advisor)

Robert Frederking

Roni Rosenfeld

Acknowledgements

I would like to thank my thesis advisor, Alexander Rudnicky, for his valuable suggestions, guidance and also his patience throughout the lengthy process of the thesis writing; the thesis committee members, Robert Frederking and Roni Rosenfeld, for their comments and suggestions; my research group colleagues, Dan Bohus and Rong Zhang for the discussion on many technical issues; Christina Bennett and Ariadna Font Llitjos for general thesis writing discussion; Sphinx group members, Rita Singh and Mosur Ravishankar for their advice and help on the Sphinx speech recognizer; my former advisor Surapant Meknavin who led me into the field of natural language processing and who has continued to give me advise throughout the years; and the last but not least my parents and my friends for all their support.

Table of Contents

List of Figures

List of Tables

Abstract

This thesis investigates N-best hypotheses reranking techniques for improving speech recognition accuracy. We have focused on improving the accuracy of a speech recognizer used in a dialog system. Our post-processing approach uses a linear regression model to predict the error rate of each hypothesis from hypothesis features, and then outputs the one that has the lowest (recomputed) error rate. We investigated 15 different features sampled from 3 components of a dialog system: a decoder, a parser and a dialog manager. These features are *speech recognizer score*, *acoustic model score*, *language model score*, *N-best word rate*, *N-best homogeneity with speech recognizer score*, *N-best homogeneity with language model score*, *N-best homogeneity with acoustic model score*, *unparsed words*, *gap number*, *fragmentation transitions*, *highest-in-coverage*, *slot bigram*, *conditional slot*, *expected slots* and *conditional slot bigram*. We also used a *linear rescaling with clipping* technique to normalize feature values to deal with differences in order of magnitude. A searching strategy was used to discover the optimal feature set for reordering; three search algorithms were examined: *stepwise regression*, *greedy search* and *brute force search*. To improve reranking accuracy and reduce computation we examined techniques for selecting utterances likely to benefit from reranking then applying reranking only to utterances so identified.

Besides the conventional performance metric, word error rate, we also proposed *concept error rate* as an alternative metric. An experiment with human subjects revealed that concept error rate is the metric that better conforms to the criteria used by humans when they evaluated hypotheses quality.

The reranking model, that performed the best, combined 6 features together to predict error rate. These 6 features are *speech recognizer score*, *language model score*, *acoustic model score*, *slot bigram*, *N-best homogeneity with speech recognizer score* and *N-best word rate*. This optimal set of features was obtained using greedy search. This model can improve the word error rate significantly beyond the speech recognizer baseline. The reranked word error rate is 11.14%, which is a 2.71% relative improvement from the baseline. The reranked concept error rate is 9.68%, which is a 1.22% relative improvement from the baseline. Adding an utterance selection module into a reranking process did not improve the reranking performance beyond the number achieved by reranking every utterance. However, some selection criteria achieved the same overall error rate by reranking just a small number (8.37%) of the utterances. When comparing the performance of the proposed reranking technique to the performance of a human on the same reranking task, the proposed method did as well as a native speaker, suggesting that an automatic reordering process is quite competitive.

Chapter 1
Introduction

In recent decades, computers have become an important tool for humans in various fields. With the invention of the Internet, computers have also become one of the most important information sources. However, conventional interactions between users and computers, e.g. typed-in commands, can prevent the users from achieving the most out of computer system efficiency. For example, they have to learn and remember SQL commands and then type them correctly in order to retrieve desired information from a database. Having to learn a new language to communicate with a computer can cause a novice user a lot of trouble.

Given the increasing computational power of current computer systems, effort can be put on the machine's side to understand human natural language. The invention of speech recognizers and advances in natural language processing algorithms make natural speech an alternative mode of interaction between users and computers. Speech is clearly a preferred mode of interaction over keyboard type-in for novice users. Furthermore, speech is an ideal solution for the situation in which a key-in device is not applicable or is impractical, for instance, accessing information over a telephone. With speech technologies, an interaction between users and computers becomes friendlier and more accessible which also implies that we can benefit more from computers.

A dialog system is one of the applications that makes use of speech technologies. In a dialog system, speech technologies and natural language understanding techniques are integrated to provide requested information and/or solve a particular task. An interaction between a user and a dialog system is in spoken language. The system is capable of recognizing and understanding user input speech. The system interprets a user's intention and then undertakes an appropriate action. Examples of system actions are: providing requested information, asking a clarification question and suggesting a solution. A response from the system is also in spoken language. The system formulates an output sentence from information it would like to convey to the user and then synthesizes the corresponding speech signal using a speech synthesizer.

However, current dialog systems are still not perfect. In many cases, the system misunderstands a user's intention due to errors in the recognizing and understanding components. The cost of system misunderstanding ranges from user confusion, a longer conversation, to the worst an incomplete task. Between recognition errors and parsing errors the former occurs more often in the CMU Communicator system. The detail of the CMU communicator system is giver in section 1.1. Even the state of the art speech recognizer can fail for many reasons. Two examples of such reasons are noise in the environment and pronunciation variations. Recognition errors are very crucial since a recognizer is the first module that handles user input. Errors caused by the speech recognition module will be propagated to modules that follow, e.g. a parser and a dialog manager. Even though the parser and the dialog manager were designed to be able to handle erroneous input, errors are still unavoidable in some cases. For example, a robust parser is tolerant to noise and errors in unimportant parts of user utterances; nevertheless, if errors occur at content words, misinterpretation is unavoidable. A dialog manager has confirmation and clarification mechanisms that can resolve the problem when such errors occur. However, this may lengthen the dialog and decrease user satisfaction. Moreover, if

recognition errors occur again in a clarification utterance, it is much harder to resolve the problem. It has been shown that the word error rate of the recognition result is highly correlated to task completion [Rudnicky, 2000]. The lower the word error rate, the higher the chance that the user get information he/she wants.

Most of the recognizers are able to give us a list of plausible hypotheses, an N-best list, that they considered before outputting the most probable one. By analyzing N-best lists of CMU Communicator data, which has a 12.5%[1] overall recognition word error rate, the most correct hypothesis of each utterance (the one that has the lowest word error rate) is not always in the top-1 position, but sometimes it is in the lower rank of the list. If the most correct hypothesis in a 25-best list is chosen for each utterance, we can reduce the word error rate to 7.9% on average, which is 37.0% relative improvement. In order to make the most correct hypothesis become the first rank in an N-best list, we need additional information that has not yet been considered by a recognizer and also an approach to apply this information to rerank the list. From an experiment on human beings, we found that they were able to recover information from N-best lists to determine the more correct hypotheses and achieved 11.1% overall word error rate, which is 10.9% relative improvement, on the same task. This improvement shows that there is information in an N-best list that can be used to recover the more correct hypothesis from the lower rank.

The goal of this thesis is to 1) determine the sources of information in an N-best list that are useful in moving the hypothesis that is more correct up to the first position of the list and 2) discover an effective approach in extracting and applying these information in a hypotheses reranking task. We aim at achieving significant word error rate reduction in the level that closes to the human performance or even better if possible.

In this chapter, we first introduce the architecture of the CMU Communicator system, which is the domain that we worked on. The details of the speech recognizer and the semantic parser, components in the system that are involved in the hypotheses reranking task, are given in the sections that follow. Then, we present the overview of the proposed solution to the hypotheses reranking problem. Finally, the organization of this thesis is given.

1.1 CMU Communicator system

The CMU Communicator system, which we will refer to as Communicator, is a telephone-based dialog system in an air travel domain. The system helps a user plan a travel itinerary, which includes flight, hotel and car reservations [Rudnicky *et al*., 1999]. The Communicator system employs the DARPA Hub architecture as a system architecture. The Hub architecture is a distributed architecture consisting of several modules that work in parallel. These modules are a speech recognizer, a parser, a dialog manager, domain agents, a natural language generator and a speech synthesizer. A diagram in Figure 1.1 shows modules in the Communicator system, the flow of the data from input speech to output speech and data representations inside a parser and a natural language generator. The details of the speech recognizer and the parser, which involve directly in a hypotheses reranking task, are described in the next sections.

[1] The numbers presented in this chapter are from our preliminary experiment.

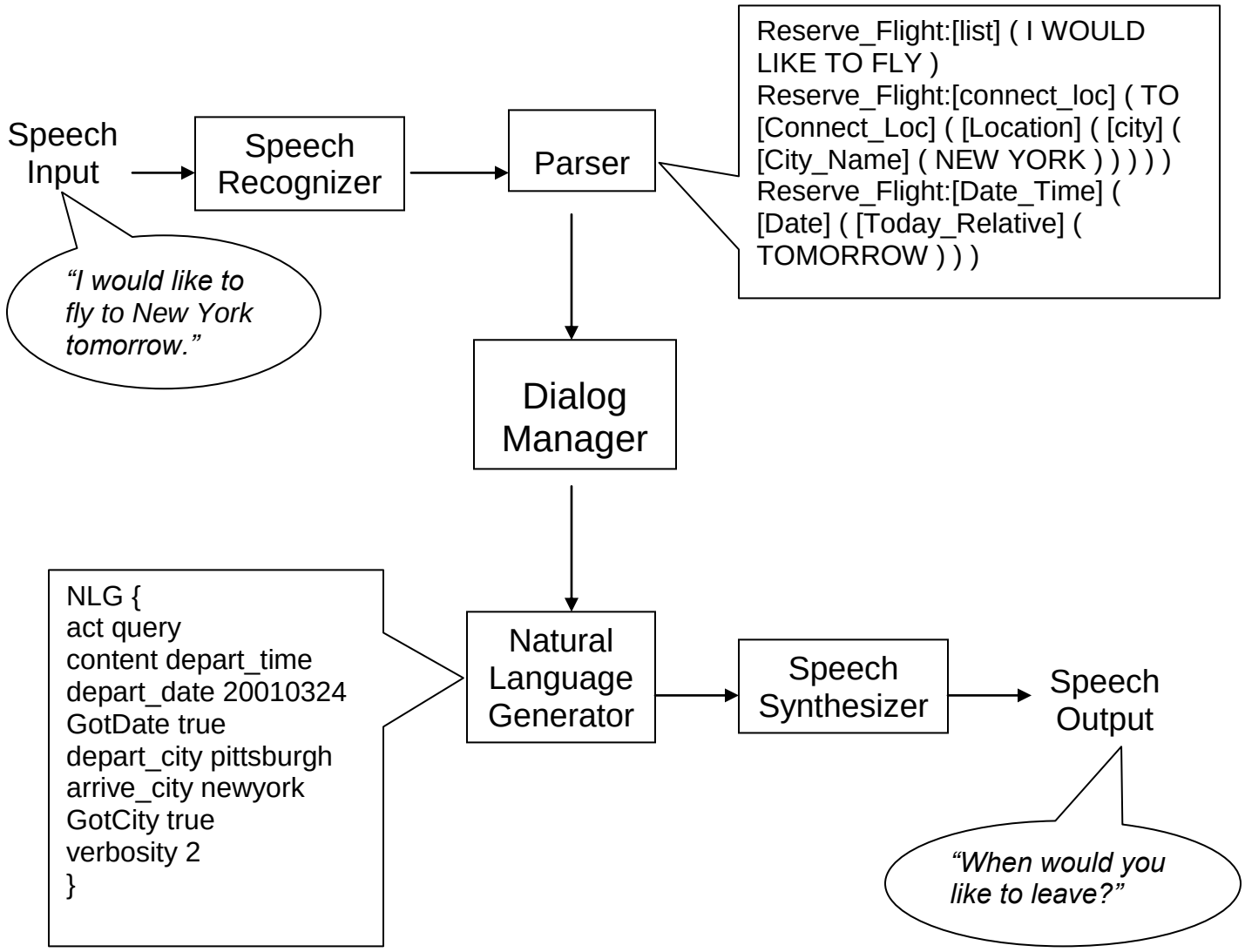

Figure 1.1: Modules and data representations in the Communicator system

1.2 Sphinx-II speech recognizer

The speech recognizer that was used in the Communicator system during the time that our training and test data were collected was the Sphinx-II recognizer [Huang *et al.*, 1993]. Sphinx-II is a large vocabulary, speaker-independent recognizer. The vocabulary size of the Communicator domain is 6,997 words. The acoustic model of Sphinx-II is a semi-continuous hidden markov model. For the language model, we used a class-based trigram language model trained by the CMU-Cambridge language model toolkit [Clarkson and Rosenfeld, 1997] with manually defined classes. In the Communicator domain, there are 20 predefined classes, for example, *city*, *airline company* and *month*. Sphinx-II is capable of generating an N-best hypothesis list as an output; however, the current Communicator system simply uses the 1-best.

1.3 The Phoenix parser

In the Communicator system, a semantic parser, Phoenix [Ward, 1991], is used to interpret the meaning of user utterances. The grammar was derived from the CMU ATIS system [Ward, 1995] and the transcriptions of the data collected using the prototype Communicator system. Given a predefined domain grammar, an output string from a speech recognizer is parsed into a sequence of *semantic frames*. Each semantic frame corresponds to a fragment of the recognizer output that governed by one grammar rule. A semantic frame contains a parse tree of the fragment and is referred to as a *slot*. Each sub-tree of the parse tree is referred to as a *sub-slot*. Figure 1.2 shows an output parse from Phoenix.

User utterance: "I would like to fly to New York tomorrow"
Speech recognizer output: I'd like to go fly the New York tomorrow
Output parse:

```
Reserve_Flight:[i_want] ( I'D LIKE TO )
Reserve_Flight:[Connect_Loc] ( [Location] ( THE [city] ( [City_Name] ( NEW YORK ) ) ) )
Reserve_Flight:[Date_Time] ( [Date] ( [Today_Relative] ( TOMORROW ) ) )
```

Frame label: Reserve_Flight
Slot label: [i-want], [Connect_Loc], [Date_Time]
Sub-slot label: [Location], [city], [City_Name], [Date], [Today_Relative]
Slot value: "I'D LIKE TO", "THE", "NEW YORK", "TOMORROW"

Figure 1.2: Output from a semantic parser Phoenix

Each line of the output parse is a parse tree which composed of a *frame label*, a *slot label*, *sub-slot labels* and *slot values*. A *frame label* represents a topic of a semantic frame; a *slot label* and a *sub-slot label* are a slot and a sub-slot identification respectively; a *slot value* is word strings captured by a slot and/or a sub-slot. Figure 1.2 also illustrates each component of the parse trees.

Since errors from the recognizer are unavoidable, the parser was designed to be robust. Even when there are errors in an input string, Phoenix is still able to generate an output parse for the portion of the input that conforms to the grammar. In Figure 1.2 the

underlined words in speech recognizer output are the errors. Even though there are errors, the parser is still able to output correct parses. As a result, the correct interpretation of user travel location and time are passed to the dialog manager.

1.4 Proposed solution

In order to make the best hypothesis risen to the first rank in an N-best list, we need to re-evaluate the quality of the hypotheses with additional information that has not yet been considered by a recognizer. Any module in the system, such as a parser, a dialog manager or even a speech recognizer itself, can provide useful information for the reordering task. We can integrate the additional information directly to a recognizer during a recognition phase or use it sequentially in a reordering module after obtaining the recognition output. The latter is also called a post-processing approach.

When comparing two approaches for improving the speech recognizer performance, an integrated approach, which works on a speech recognizer directly, and a post-processing approach, which works on the result of a recognizer, both of them have different advantages. The first approach considers all information at the same time during a recognition phase; therefore, no information is lost as it might happen in a post-processing approach. In a post-processing approach, information in the input is used sequentially by a recognizer first and then by a post-processing module. Information from a recognition phase available during a post-processing phase is limited to only what a recognizer provides, usually an N-best list or a word lattice. However, a post-processing approach is considered less complex than an integrated approach, in which a speech recognizer may need to be modified in order to incorporate additional information. A post-processing technique also benefits from features extracted from completed hypotheses such as syntactic features.

In this research, we chose a post-processing technique called N-best hypotheses reranking to improve the recognizer accuracy. Information or features from different parts of a spoken dialog system, such as a recognizer, a parser and a dialog manager, are extracted from a recognizer N-best hypothesis list and an associated system utterance. The hypotheses in an N-best list are reordered according to this information and the new first rank hypothesis is output as a reranking result. We chose a post-processing technique because it is easier to incorporate many kinds of features into a separate reranking module than to integrate them directly into a speech recognizer. A post-processing technique allows us to experiment on various reranking features and their combinations without making much intervention on a recognizer. Moreover, a post-processing approach can benefit from features derived from completed hypotheses.

In order to benefit from various types of information extracted from an N-best list, instead of just using the best one, we used a linear regression model to optimally combine different features together. A linear regression model is used to predict the correctness score of each hypothesis given a list of features.

However, most of the utterances are already correct and don't have to be reranked. In the training set, about 91.0% of the utterances have their most correct hypotheses on the first rank. To limit the effort put into a reranking task and to avoid creating errors on utterances that are already correct, we limit the reranking scope to only dubious utterances. Based on this idea, an utterance selection module is added to a reranking process prior to a reranking module. The task of an utterance selection module

is to determine which utterance needs to be reranked and then pass it to a reranking module. For an utterance that doesn't have to be reranked, the original first rank hypothesis is output. In this research, many criteria were examined in order to find an appropriate selection criterion.

The performance of a reranking module is evaluated from the correctness of the reranked first rank hypothesis. Since we are interested in reranking accuracy in the context of a dialog system, we also consider concept accuracy as an alternative performance metric, in addition to conventional word accuracy. Concept error rate is relevant since it, rather than word accuracy, governs the behavior of the system as a whole.

1.5 Thesis organization

This thesis is organized as follows. Chapter 2 is a literature review of related research regarding the N-best hypotheses reranking problem. Chapter 3 describes features that are considered useful in a reranking task and techniques are used to extract these features. Chapter 4 presents a feature combining method, a linear regression model, and its optimization techniques along with feature representation techniques. Chapter 5 explains an utterance selection module and possible selection criteria. Chapter 6 describes the definition of concept types and concept error rate and the implementation of concept error rate. The experimental procedure and all results and discussion are given in Chapter 7. Finally, Chapter 8 concludes this thesis.

Chapter 2
Literature Review

In this chapter, we discuss the research works that are related to the N-best hypotheses reranking problem. Many researchers have been working on improving the accuracy of the speech recognizer by utilizing the information in an N-best list or a word lattice (the latter is considered to contain more information). The research in this area can be categorized into two broad approaches based on the time when information from an N-best list or a word lattice is considered: during a recognition phase or after a recognition phase. We discuss the first approach, which we call the integrated approach, in section 2.1. Then we review the works that employ the second approach, the post-processing approach, in the section that follows. Section 2.3 discusses an experiment on N-best hypotheses reranking by humans. Finally, section 2.4 discusses related but not identical works on speech recognition accuracy that were done on the same Communicator system.

2.1 Integration approach

The integration approach uses additional information from an N-best list or a word lattice to adjust the hypothesis score given by a speech recognizer during the recognition phase. Therefore, a recognizer may need to be modified to incorporate the additional information. In [Mou and Zue, 2000], a word lattice was rescored during a recognizer search phase using a technique called dynamic reliability. The idea was based on the assumption that some acoustic units were modeled and recognized more reliably than others. Reliability was measured by the likelihood of choosing a hypothesized acoustic unit to extend the current partial search path as opposed to using other units. For each acoustic unit, 2 Gaussians were trained, one for the correct acoustic unit and another one for the incorrect acoustic units. The reliability score was then used to adjust the score of partial search path. This approach achieved 24.0% relative word error rate reduction on JUPITER, an English weather information system, which had 12.1% initial word error rate, and it achieved 28.6% relative word error rate reduction on PANDA, a Chinese weather information system, which had 11.9% initial word error rate.

2.2 Post-processing approach

The post-processing approach views a speech recognizer as a black box; no modification is made directly to a recognizer itself. A recognizer is asked to provide a list of hypothesized sentences or a word lattice. Then the post-processing module is applied to the N-best list or the word lattice to improve the accuracy of the recognizer output. Word error rate is commonly used to evaluate the performance of a recognizer; however, the maximum likelihood algorithm in a recognizer minimizes sentence error rate which may not be correlated to word error rate in the case that the error rate is high. The explicit word error minimization approach [Stolcke *et al*., 1997] was introduced to address this problem. No additional information was used in this method; only the criterion for selecting the most correct hypothesis was changed. The expected word error rate of each hypothesis was calculated from the sum of the word error rated of that hypothesis using other hypotheses in an N-best list as references. The expected word error rate was weighted by the posterior probability of the hypothesis that was used as the reference.

The posterior probability was given by the recognizer. This technique improved the word error rate by 1.0% relatively on the Switchboard corpus with the initial word error rate of 52.7 %, and 0.9% relatively on the CallHome corpus with the initial word error rate of 68.4%. This improvement is quite small compared to other approaches. Furthermore, no improvement was found on the test set with lower initial word error rate.

In [Mangu *et al*., 1999], explicit word error minimization was done on a word lattice instead of an N-best list. A technique for word lattice alignment was proposed in order to efficiently compute the expected word error between two hypotheses. This model achieved 3.6% relative improvement from the baseline word error rate of 38.5% on the Switchboard corpus compared to 1.6% relative improvement gained by using explicit word error minimization on the N-best list of the same corpus. The higher performance on a word lattice came from a better posterior estimation and a larger candidate set for hypothesis selection.

In [Rayner *et al*., 1994] and [Moore *et al*., 1995], additional information that had not been used by the recognizer was introduced and applied to rerank an N-best hypothesis list. In [Rayner *et al*., 1994], additional knowledge sources from different levels of language understanding, such as surface knowledge (word class N-gram), syntactic knowledge (grammar rules) and semantic (words and grammatical relation), were combined with acoustic knowledge from the recognizer score to reorder an N-best hypothesis list. Information from these knowledge sources was used in the form of discriminate scores. The discriminate score reflected the usefulness of each feature in discriminating between correct and incorrect hypotheses. Various knowledge sources were combined together using a near-optimal weight sum. The best combination achieved 12.2% relative word error rate improvement on the ATIS (Air Travel Information System) 1993 test set which had the initial word error rate of 7.4 %. The analysis of the result revealed that syntactic knowledge was a good feature in the sense that it picked harmless syntactic variants of the correct hypotheses. The result also showed that the N-gram feature was able to capture surface uniformities between speakers.

A similar technique on the post-processing approach was found in [Moore *et al*., 1995]. The multi-level N-gram model was used to rescore an N-best hypotheses list based on the assumption that the closer integration of linguistic and statistical factors could improve a language model. The multi-level N-gram model was composed of tri-grams of fragment types, four-grams of words and word-classes and the within class probability. The fragment types, e.g. a sentence, a nominal phrase and filler, were additional information from a parser. No acoustic information was used. The rescoring model improved the word error rate by 14.6% relatively on the test set with 2.5% initial word error rate, and 7.3% relatively on a higher, 13.7%, initial word error rate test set in the ATIS1994 domain. The model also selected low rank hypotheses such the 22nd rank and the 63rd rank.

The approaches in [Ringger and Allen, 1996(1)], [Ringger and Allen, 1996(2)] and [Ringger and Allen, 1997] were also considered a post-processing approach. However, the algorithms were not restricted to choose the most correct hypothesis out of an N-best list or a word lattice. On the other hand, arbitrary editing was allowed to modify the first rank hypothesis. A noisy channel model was used to model the relationship between a correct user utterance and the output of the speech recognizer. The model consisted of 2 parts, a language model and a channel model. A language model

accounted for the likelihood of an intended user utterance. The channel model accounted for the likelihood of errors made by a speech recognizer when it substituted a user word with another word, deleted a user word, or inserted an extra word. Error patterns were collected and then modeled to recover the correct hypothesis back from the corrupted hypothesis. This approach achieved 14.9% relative word error rate reduction on TRAINS-95, a train travel domain corpus, which had 41.3% initial word error rate when the speech recognizer was trained on ATIS, an air travel domain corpus. It also achieved 24.5% relative word error rate reduction when the speech recognizer performance was improved to 32.0% initial word error rate by training the recognizer on the data from the same domain as the test set. However, for the case that the recognizer performance is quite good and error patterns are sparse, this approach may not work well.

The performances, measured by word error rate, of all discussed approaches are given in the table below.

Approach	Corpus	Baseline WER	New WER	Relative Improvement
Reliability scoring [Mou and Zue, 2000]	JUPITER	12.1%	9.2%	24.0%
	PANDA	11.9%	8.5%	28.6%
Explicit word error minimization (N-best hypotheses) [Stolcke *et al.*, 1997]	Switchboard	52.7%	52.2%	1.0%
	CallHome	68.4%	67.8%	0.9%
Explicit word error minimization (N-best hypotheses) [Mangu *et al.*, 1999]	Switchboard	38.5%	37.9%	1.6%
Explicit word error minimization (word lattice) [Mangu *et al.*, 1999]	Switchboard	38.5%	37.1%	3.6%
Combining knowledge sources [Rayner *et al.*, 1994]	ATIS1993	7.4%	6.5%	12.2%
Multi-level N-gram [Moore *et al.*, 1995]	ATIS1994	2.5%	2.1%	14.6%
	ATIS1994	13.7%	12.7%	7.3%
Post-processing error correction [Ringger and Allen, 1996(2)]	ATIS/TRAINS-95[1]	41.3%	35.5%	14.9%
	ATIS+TRAINS-95/ TRAINS-95	32.0%	24.5%	24.0%

Table 2.1: The performances of different approaches in improving speech recognizer performance

2.3 Human experiments

In the approaches that use additional information to improve the recognizer performance, such as [Rayner *et al.*, 1994] and [Moore *et al.*, 1995], the first question that arises is what is the useful additional information that we should employ. A human experiment was conducted in [Brill *et al.*, 1998] to investigate the features that humans used in order to select the most correct hypothesis out of the list of N-best hypotheses and

[1] The first corpus was used to train a speech recognizer while the second corpus was used as a test set.

how good they were at the task. In the experiment, the subjects were asked to choose the most correct hypothesis out of a 10-best list and they were also allowed arbitrary editing of the hypotheses. The experiment was done on 3 different corpora: Switchboard, Broadcast News and Wall Street Journal. The result of the experiment is summarized in Table 2.2. The result showed that humans could reduce the recognizer errors via a post-processing technique. The word error rate reduction on human editing was higher than that of human selecting. On the Wall Street Journal corpus, where humans achieved the highest error reduction rate, human selecting gained 23.5% relative improvement while human editing gained 30.3% relative improvement on the test set with 13.2% initial word error rate. The word error rate reduction also depended on the amount of information provided by N-best hypothesis lists which correlated to the recognizer accuracy and the length of utterances. The most common feature that human used was *closed class word choice*. The features used frequently by humans usually helped in reducing the word error rate. Even though some of the features used by human in selecting the most correct hypothesis out of an N-best list are difficult to implement in a computer system, there are, nevertheless, some possible solutions.

Approach	Corpus	Baseline WER	New WER	Relative Improvement
Human selecting	Switchboard	43.9%	42.0%	4.3%
	Broadcast News	27.2%	25.9%	4.8%
	Wall Street Journal	13.2%	10.1%	23.5%
Human editing	Switchboard	43.9%	41.0%	6.6%
	Broadcast News	27.2%	25.2%	7.4%
	Wall Street Journal	13.2%	9.2%	30.3%

Table 2.2: The performance of human subjects on a hypotheses reranking task

2.4 CMU Communicator related research

In this section, we discuss two related research topics, word level confidence annotation and utterance level confidence annotation which were also done on the CMU Communicator system. The reason that we include these research topics in the literature review even though they are less relevant than other works is that they were implemented on the exact same system as the one we were working on. We could adopt useful ideas easily or even use their work as part of our proposed model without much modification. Furthermore, the result of our proposed technique can be compared to their results directly since we worked on the data from the same system.

[Zhang and Rudnicky, 2001] described the technique used in word level confidence annotation. Each word in the speech recognizer output string was tagged as *correct* or *incorrect* according to the annotator. From this binary tag, confidence annotation can be regarded as a classification task. The features that were used to discriminate between correct and incorrect words came from 2 sources, a recognizer and a parser. Recognizer-based features were categorized into 4 categories: acoustic features, language model features, word lattice features and N-Best list features. The recognizer-based features are, *the percentage of frames match the phone-only decoding*, *the percentage of phones match the phone-only decoding*, *normalized acoustic score*,

language model back off mode, *log posterior word probability by acoustic score*, *log posterior word probability by language model score*, *log posterior word probability by acoustic score and language model score*, *N-best homogeneity and N-best word rate*. Among nine them that were investigated, a word lattice feature, *log posterior word probability by acoustic score and language model score*, had the lowest annotation error rate of 20.8%. Two parser-based features were examined: *a parsing mode* and *a slot back off mode*. Their performances were comparable to the top recognizer-based features. Three classification techniques: decision tree, neural network and support vector machine (SVM), were used to combine the features in order to predict annotation tags. For an SVM, 5 kernel functions were examined. The SVM with the ANOVA kernel function was the best classifier with 18.2% annotation error rate.

[Carpenter *et al*., 2001] addressed a similar problem, but the confidence annotation was done at the utterance level. The confidence annotator assigned each recognizer output utterance either an *OK* or a *Bad* label. Again, this can be considered as a classification task. Three groups of features: recognizer features, parser features and dialog features, were used in the classification. Recognizer features are *word numbers* and *confidence percentage*. Parser features are *uncovered percentage*, *fragment transitions*, *gap number*, *slot number*, *slot bigram* and *garble*. Dialog features are *dialog state*, *state duration*, *turn number* and *expected slot*. Among the twelve features, uncovered percentage was the best feature with classification error rate of 19.9%. Five classification techniques were used to combine the features: Bayesian network, AdaBoost, decision tree, neural network and support vector machine. All classifiers except Naive Bayes had comparable error rates of around 18%.

Chapter 3
Feature Selection

As mentioned in the first chapter, additional information or feature that can help in reevaluating the quality of hypotheses is needed in order to promote the best hypothesis up to the first rank. Different types of information were investigated based on the analysis of the domain and the knowledge sources reported to be used by human beings in the same reordering task. This human experiment will be discussed in Chapter 7. The features come from three components in a dialog system: a speech recognizer, a parser and a dialog manager. Therefore, we can categorize the features based on their information sources into 3 groups: speech recognizer features, parser features and dialog manager features.

3.1 Speech recognizer features

Speech recognizer features are the features obtained from a recognizer during a recognition phase. The speech recognizer features are useful because they reflect the similarity between the hypothesis and the input signal. Even though they have already been used to generate a recognizer output, some of them are not used directly when a recognizer selects its most probable hypothesis. Moreover, A user doesn't always response according to the system expectation, and sometimes syntactic and semantic information are not sufficient to distinguish between competing hypotheses. The *yes/no* answer is a good example of the necessity of acoustic features. There are two types of recognizer features that we investigated, a hypothesis score and an N-best list score.

3.1.1 Hypothesis scores

Hypothesis scores are the scores that associate directly with each hypothesis. In most speech recognizers, hypotheses are ranked according to their posterior probability calculated by the following equation.

$$P(W \mid X) = P(X \mid W) * P(W) \tag{3.1}$$

where X is acoustic signal and W is a string of hypothesized words.

All three terms in the equation are used as features as described below:

1) *Speech recognizer score*, $P(W|X)$, is the likelihood of the hypothesis given the input speech signal. It is also a combination of acoustic model score and language model score.
2) *Acoustic model score*, $P(X|W)$, is a score given by the Hidden Markov Model. Acoustic model score captures the likelihood that the input signal comes from a specific sequence of phones.
3) *Language model score*, $P(W)$, is a probability estimated by a class-based trigram language model. Language model score captures the likelihood that a specific sequence of words occurs in the language.

Speech recognizer score is the feature that the recognizer used to rank the hypotheses in an N-best hypothesis list initially. Therefore, this feature also serves as the baseline of the reranking task.

3.1.2 N-best list scores

An N-best list score contains the information extracted from a set of hypotheses in an N-best list as opposed to the information obtained from an individual hypothesis in a hypothesis score. The similarity between hypotheses in an N-best list can indicate the confidence of a recognizer in recognizing the input speech. We considered two features from an N-best list, *N-best word rate* and *N-best homogeneity*, as described below.

1) N-best word rate

we believe that if a particular word occurs in many hypotheses of an N-Best list, we should be confident that the word is correctly recognized. This feature was also used in a word level confidence annotation task [Zhang and Rudnicky, 2001], In order to determine that words in two hypothesis strings are the hypothesized words of the same input word, we need to consider the positions of the words in the hypothesis strings as well as their contents. We use start and end frames to represent the position of a word in a hypothesis string. Specifically, consider a word w_i, which starts at a frame s_i and ends at a frame e_i in a hypothesis h_1, and likewise a word w_j, which starts at a frame s_j and ends at a frame e_j in a hypothesis h_2. To compensate for the uncertainty in a word boundary, we extend the boundary to the left by 1 frame and to the right by 4 frames. So the extended word boundary (s'_i, e'_i) is equal to (s_i –1, e_i +4). This idea was adopted from the idea of *slop* described in [Bansal and Ravishankar, 1998]. Word w_i and word w_j are in the same position if one of the following conditions is true:

i) The extended word boundary of w_i contains the extended boundary of w_j or vise versa,

ii) More than 75% of extended boundaries of both words are overlapped. The overlapped percentage is calculated by the following equation.

$$\text{Overlapped percentage} = \frac{\#overlapped\ frames}{\max(e'_i - s'_i, e'_j - s'_j)} \tag{3.2}$$

where (e'_i - s'_i) is the extended word length.

Word w_i and word w_j are considered the same word if both their contents and positions in the hypothesis strings are the same. N-best word rate of a word w_i is calculated from the number of hypotheses that contain the same word w_i divided by the total number of hypotheses in an N-best list. Specifically, N-best word rate of a word w_i is defined by,

$$\text{N-best word rate } (w_i) = \frac{Number\ of\ hypotheses\ containing\ w_i}{Number\ of\ hypotheses\ in\ an\ Nbest\ list} \tag{3.3}$$

N-best word rate is a confidence score that associates with each word. In a hypothesis reranking problem, we need an overall N-best word rate score for each hypothesis. We examined two variations in the computation of an overall N-best word rate score.

- *Average raw score*

  An overall N-best word rate is calculated from the average of the N-best word rate score of every word in a hypothesis.

- *Average confidence score*

  If there is a deletion error in one of the hypotheses in the N-best list, the correct word doesn't appear in every hypothesis. For that reason, if a particular word occurs frequently enough in the N-best list, but not necessarily in every hypothesis, it should be considered as a confident word. Thus, we threshold an N-best word rate score of each word into a confidence score of 0 or 1. If the N-best word rate is greater than or equal to 0.5, the confidence score is 1, otherwise the confidence score is 0. The overall N-best word rate is the average of the confidence score of every word in a hypothesis.

2) N-best homogeneity

In an N-best hypothesis list, the hypothesis in a higher rank is considered more accurate, according to a recognizer, than the one in a lower rank. Therefore, instead of using a word count, which gives an equal weight to words form any hypothesis, to determine the degree of homogeneity of an N-best list, we weigh each word by the hypothesis score of the hypothesis that it belongs to. N-best homogeneity is calculated in the same manner as N-best word rate except that the hypothesis count is replaced by the sum of the hypothesis scores instead. Specifically, N-best homogeneity is defined as,

$$\text{N-best homogeneity } (w_i) = \frac{\textit{Sum of scores of hypotheses containing } w_i}{\textit{Sum of scores of hypotheses in an Nbest list}} \quad (3.4)$$

The scores used in equation (3.4) can be any of the hypothesis scores in section 3.1.1, a speech recognizer score, an acoustic model score or a language model score.

An overall N-best homogeneity score is the average of the N-best homogeneity score of every word in a hypothesis.

3.2 Parser features

3.2.1 Parse quality

In the Communicator system, we expect user utterances in an air travel domain. We use a pre-defined grammar to describe the domain language. Thus, the conformance of a hypothesis string with respect to the grammar can be used to select the hypothesis that is more relevant to the domain. This idea was also exploited in [Rayner *et al*., 1994] and [Moore *et al*., 1995]. The quality of a hypothesis parse is a good indicator of how well a hypothesis string matches the domain grammar. We believe that a more correct hypothesis has a higher parse quality while a hypothesis with more errors has a lower quality parse. To quantify the quality of a hypothesis parse, we can either give penalties for the parts of a hypothesis that are not covered in the parse or give rewards for the parts of a hypothesis that are in the parse. For *unparsed penalty*, we can use the following characteristics of a hypothesis parse to represent the parse quality.

1) *Uncovered words*, is the number of words in a hypothesis that are not covered in the hypothesis parse.
2) *Gap number*, is the number of unparsed fragments, or gap slots, in the hypothesis parse.
3) *Fragmentation transitions*, is the number of changes from a parsed fragment or a semantic slot to an unparsed fragment or a gap slot and vice versa in the hypothesis parse. This feature indicates the fragmentation degree of the parse. For the hypothesis that contains only a gap slot, a fragmentation transitions feature is set to a predefined maximum value, which is 100 in our experiment.

Unparsed penalty features represent the quality of the parse at different granularities. Uncovered words captures the parse quality at a word level while gap number captures the quality at a slot level. Fragmentation transitions looks at an even broader level, the continuity of a sequence of slots.

Gap number and fragmentation transitions are correlated. The value of the fragmentation transitions feature is twice as much as the value of the gap number feature unless a gap slot occurs at the beginning or the end of a hypothesis. Nevertheless, we introduce both of them since it is not clear if one is a better predictor than the other. Moreover, a linear regression model, which is discussed in Chapter 4, will choose the one that is better.

On the other hand, *under-covered reward* features represent the quality of a hypothesis parse by giving rewards to the parts of the hypothesis that are covered in the parse. Three under-covered reward features are described below. They are counterparts of unparsed penalty features.

4) *Coverage*, is the number of words in a hypothesis that are covered by the hypothesis parse.
5) *Semantic slot number*, is the number of semantic slots in the hypothesis parse.
6) *Slot continuity*, is the number of times the succeeding slot is the same slot type (semantic or gap) as the current slot. For the hypothesis that has only a gap slot, a slot continuity feature is set to zero.

System: "What time do you want to travel?"
User: "I'd like to go around noon please."
Hypothesis: I'd like to go hung around noon please.
Hypothesis parse:
Reserve_Flight:[i_want] (I'D LIKE TO)
Gap:[Gap] (GO HUNG)
Reserve_Flight:[Time_Range] ([time_spec] ([_aprx_time] (AROUND [Time] ([_noon] (NOON)))))
Reserve_Flight:[Polite] (PLEASE)

Figure 3.1: A sample dialog with a speech recognizer hypothesis and its corresponding parse

We can normalize parse quality features by the hypothesis length. For unparsed words and coverage features, we can do the normalization by dividing the features with the total number of words in the hypothesis. Gap number and semantic slot number features are normalized by the total number of slots including gap slots. For fragmentation transitions and slot continuity features, we use the total number of transition between slots, which equals to the total number of slots including gap slots minus one, as a normalization factor.

From a sample dialog in Figure 3.1, the recognizer inserted an extra word '*hung*' in its hypothesis. In the hypothesis parse, the second fragment, which is denoted by *Gap:[Gap](GO HUNG)*, indicates a gap slot. A gap slot is inserted into the original hypothesis parse at the point where the hypothesis string is not covered by the parse. Even though there is only one error in the recognizer hypothesis, there are two words '*go*' and '*hung*' that are not conversed in the parse. The values of parse quality futures of the recognizer hypothesis in Figure 3.1, both unparsed penalty features and under-covered reward features, are given in Table 3.1. The normalized values of these features are also given.

Feature	Value	Normalized value
Unparsed penalty		
Uncovered words	2	2/8
Gap number	1	1/4
Fragmentation transitions	2	2/3
Under-covered reward		
Coverage	6	6/8
Semantic slot number	3	3/4
Slot continuity	1	1/3

Table 3.1: The values of parse quality features of the hypothesis in Figure 3.1

The previous set of features prefers the hypothesis that has fewer uncovered parts or more covered parts in the parse tree. However, when there is a recognition error in a hypothesis, it may result in parse errors not only at the error word but also at the parts of the hypothesis nearby. There is a simpler parse quality feature that gives a preference to only the hypothesis that can be parsed entirely. This feature is called highest-in-coverage in [Rayner et al., 1994].

7) Highest-in-coverage, is a binary feature which equal to one if a hypothesis can be parse entirely, and zero otherwise. When uses a highest-in-coverage feature, a reranking module outputs, among the set of hypotheses that are fully parsed, the one that has the highest speech recognizer score.

3.2.2 Slot N-gram model

Apart from the quality of the hypothesis parse, a sequence of slots can also be used to measure the quality of each hypothesis. Each slot represents a meaningful unit in a hypothesis; therefore, we believe that some sequences of them are more probable than the others. We used a slot N-gram model to capture the relationship between slots. The following aspects were considered when we trained a slot N-gram model.

1) Slot representation

From section 1.3, each slot, or parse tree, is composed of a frame label, a slot label, sub-slot labels and slot values. A slot label is a unique representation of each slot. We can use a slot label or a slot label together with a frame label as a slot representation. However, using both a frame label and a slot label to represent a user utterance seems to be redundant. For example, *Reserve_Flight:[i_want]* and *Reserve_Hotel:[i_want]* are similar user expressions. This example also indicates that some slots are members of more than one semantic frame. This can cause an ambiguity to a parser when it determines the semantic frame label, if there is no further information. A parser does sometimes assign an incorrect frame label to a parse tree, even though all other parts, such as a slot label, are correct. Furthermore, a slot label has a stronger association with a hypothesis string than a frame label. A slot label associates with a grammar rule, which specifies patterns of word strings that can be parsed, while a frame label is determined later on from a set of slots. Therefore, using only a slot label to represent a slot seems to be a better choice. The results of slot N-gram models with both slot representations, a slot label and a slot label together with a frame label, are given in section 7.4.3 of Chapter 7.

2) Modeling a gap slot

We also modeled unparsed fragments of a user utterance, or gap slots, in a slot N-gram model. However, there are quite a number of gap slots even when parsing directly from the transcript; namely 7.68% of slots in the transcript are gaps. Gap slots in the transcript come from two possible sources, an out of domain part of an utterance and an in domain but unexpected part of an utterance. The later case indicates the lack of coverage of the grammar. If we represent a gap slot by its slot label like other semantic slots, each gap slot will share the probability mass of all the gaps and become more likely in a slot N-gram model. However, a gap slot has some characteristics that are different from the characteristics of other semantic slots. For semantic slots, the portions of utterances represented by the same slot label have similar meaning. But for gap slots, the fragments of utterances under them are varied since gap slots cover every string that is not in the grammar. Hence, we differentiated gap slots by the word strings inside the gaps, or *gap string*. Each gap slot is then represented by a gap string in a slot N-gram model. From the example in Figure 3.1, the gap slot "*Gap:[Gap](GO HUNG)*" has "*[Gap]*" as a slot label and "*GO HUNG*" as a gap string. It is represented in a slot N-gram model as "*[Gap](GO HUNG)*" instead of just "*[Gap]*".

3) Vocabulary and vocabulary type

When we represent a gap slot with a gaps string, the vocabulary type of a slot-gram model changed from a closes vocabulary set of predefined slots to an open vocabulary set since a set of gap strings is an infinite set. Gap slots in hypotheses are quite different from those in the transcript since many gaps in hypotheses are caused by recognition errors. Since we are interested more in the recognition errors, we model the gaps in the transcript in a slot N-gram model, but not the gaps in the hypotheses. We do this by using all possible semantic slots and all seen gap slots in the reference as a vocabulary set. Therefore, there is no

unknown in the training data, which is the transcript. As a consequence, a gap string in a hypothesis that wasn't seen before in the transcript, which is likely to be a recognition error, is considered an unknown slot and receives low probability from a slot N-gram model.

4) Discounting Strategies

Discounting strategy is the technique that redistributes the probability mass from the more commonly observed events to the less frequent and unseen events. We experimented on 4 different discounting strategies available in the CMU-Cambridge language model toolkit: linear discounting, absolute discounting, Good-Turing discounting and Witten-Bell discounting, [Clarkson and Rosenfeld, 1997]. The perplexity of the slot N-gram models and their hypothesis reranking performances will be reported in section 7.4.3 of Chapter 7.

3.3 Dialog manager features

In a goal-oriented conversation, e.g. negotiating a travel plan, two participants who engage in the conversation cooperate on the task and organize their conversation so that their goal is successfully achieved. For example, an agent tries to get as much information as possible from a client, in order to accommodate his/her request. On the other side, a client cooperates by providing information about his/her preference or restriction and answering a client's questions accordingly. This characteristic of a goal-oriented dialog also holds for a human-machine conversation.

In the Communicator system, a computer plays an agent role while a human a user plays a client role. On the machine side, the dialog manager is the main component that controls a conversation. As a conversation proceeds, the dialog manager will be in different *dialog states* depending on which part of the travel plan is being discussed. They may first discuss about a departure flight then move on to a return flight and then talk about a hotel and a car rental after that. The current Communicator system has 18 dialog states, for instance, *query_depart_date*, *query_arrive_city* and *query_confirm*.

Since both participants cooperate in a conversation, we can assume a correlation between system utterances and user utterances. Therefore, the information about the prior system utterances can help predicting the upcoming user utterance. We represent a system utterance by its corresponding dialog state, to make the problem tractable. For a user utterance, we use represents it by a sequence of slots which are the parse trees for that utterance. The representation of each slot is a slot label similar to the representation of a slot in the slot N-gram model discussed in section 3.2.2. Three features that make use of information from dialog states are discussed below.

3.3.1 Expected slots

When we examined logged conversations, we found that most of the system utterances are questions either for obtaining more information from a user or making a clarification. In unproblematic conversation, we expect a user to cooperate with a system and respond with an expression that conforms to the previous system utterance. At each state of the system, a dialog manager expects particular slots in a user utterance. For example, in the state *query_arrive_city*, a dialog manager expects slots *[city_name]* or *[location]*. In the utterance level confidence annotation task described in [Carpenter *et al*., 2001], the expected slots feature's performance is among the top features.

For each dialog state, we list all the slots that are expected by a dialog manager manually. We investigated two different techniques for calculating an expected slots feature as described below.

1) Binary representation

The value of an expected slots feature is either 0 or 1. If the considered hypothesis contains at least one slot that is expected by the dialog state of the previous system utterance, the expected slots feature has a value of one; otherwise the value is zero.

2) Proportional representation

The value of an expected slots feature is a continuous value from 0 to 1. The value is the ratio between the number of slots that are expected by the system and the total number of slots that are in the considered hypothesis.

Further work on the utterance level confidence annotation task suggests that considering *acceptable slots* in addition to expected slots helps to improve the performance of this feature. An acceptable slot is a slot that is not expected by any particular state, but is acceptable as a reasonable response to all the states. Examples of acceptable slots are *[Help]* and *[Repeat]*. Acceptable slots are added to the list of expected slots of every state.

From the sample dialog in Figure 3.1, the dialog state of the given system utterance is *query_depart_time* and the hypothesis slots are *[i_want]*, *[Gap](GO HUNG)*, *[Time_Range]* and *[Polite]*. Only the *[Time_Range]* slot is expected by the dialog manager at this state. Therefore, the binary expected slots feature has a value of one since the hypothesis slots contain an expected slot, while the value of the proportional expected slots feature is 1/4 since only one slot in four hypothesis slots is expected. If we also consider acceptable slots, the value of the proportional expected slots feature changes from 1/4 to 2/4 since *[Polite]* is an acceptable slot. However, the value of binary expected slots feature still remains the same.

3.3.2 Conditional slot model

As mentioned earlier, in a human-computer conversation, a user utterance is likely to be influenced by the previous system utterance, especially if the previous system utterance is a question. Given the previous system utterance, some user responses are more likely that the others. For example, if the previous prompt is "*What time do you want to travel?*" as in Figure 3.1, a user response that follows that prompt is more likely to contain an expression about time rather than a "*yes*/*no*" answer. We model the relationship between a system utterance and a user utterance by the conditional probability of a user utterance given a system utterance or a *conditional slot* feature which can be expressed as follows:

$$P(u_{user} \mid u_{system}) = \prod_{i}^{N} P(topic\_slot_i \mid dialog\_state) \tag{3.5}$$

where *U* is an utterance and *N* is the number of slots in the user utterance.

We also consider gap slots in a conditional slot model. Each gap slot is represented by a word string inside the gap as in the slot N-gram model.

From the sample dialog in Figure 3.1, the conditional slot feature can be calculated as the following.

P("I'd like to go hung around noon please" | "What time do you want to travel")

= P(*[i_want]* | *query_depart_time*) * P(*[Gap](GO HUNG)* | *query_depart_time*) * P(*[Time_Range]* | *query_depart_time*) * P(*[Polite]* | *query_depart_time*)

= (-1.3993) + (-6.0766) + (-0.4817) + (-1.9179) = -9.8755[1]

Both the conditional slot feature and the expected slots feature described in section 3.1.1 capture the relationship between a system utterance and a user utterance; however, unlike the expected slots feature, which models the relationship from the dialog manager's point of view (namely, which slots it expects to get from a user for each state), the conditional slot feature models the relationship from the user's point of view. In other words, what are the actual slots that a user uses to respond to the system prompt at each state. One of the differences is that with the conditional slot feature, every slot is allowed for every state, but the likelihood is determined by the training data. Whereas with the expected slots feature only expected slots are allowed. Since a user may not always respond in the way that the system expects, it may be better to use a data-driven approach, as in a conditional slot model, to capture the relationship between a system utterance and a user utterance.

3.3.3 State-conditional slot N-gram model

Since the previous system utterance has influence on a user utterance, it can also have influence on a sequence of slots, not only an individual slot. In [Hacioglu and Ward, 2001], a sequence of concepts was conditioned on the *dialog context* in the dialog dependent concept model for N-best hypotheses rescoring. The notion of their concept is similar to our slot, while the dialog context is similar to our dialog state. In order to generate a state-conditional slot N-gram model, we condition the slot N-gram model described in section 3.2.2 with the corresponding dialog states. We adopted two conditioning techniques that were used in [Hacioglu and Ward, 2001]. Both techniques are described below.

1) State context-cue model

For each slot sequence, a normal context cue, the beginning of sentence symbol "<*s*>", is replaced by a dialog state that associates with that slot sequence. In this model, a dialog state has influence only on the first N-1 tokens in the slot sequence.

2) State-specific model

A separate slot N-gram model was trained for each dialog state using only user utterances corresponding to that state. In this model, every slot N-gram sequence is influenced by a dialog state.

[1] To avoid the magnitude problem, we work on the log-base instead. Therefore, product becomes addition.

Chapter 4
Feature Combination: Linear Regression Model

In the previous chapter, we discussed features that might be useful in the hypotheses reordering task. Unfortunately, none of the individual features could improve the word error rate beyond the speech recognizer baseline, and only some features, such as parse quality scores and N-best homogeneity with recognizer score, could improve the concept error rate[1] but not significantly as shown in section 7.4.6.

The baseline speech recognizer score is a weighted combination of acoustic model score and language model score. The weights are usually determined from the experiment and adjustable in some speech recognizers. From table 7.9, which illustrates the performance of each individual feature, the performance of language model score ranks in the middle while the performance of acoustic model score is the worst. Since there are many features that performed better than acoustic model score and language model score individually, we expected that combining these features together in the reranking module would improve the reranked word error rate and concept error rate beyond the baseline. In order to determine the combining weights systematically, we use a linear regression model to estimate the optimal weights given a set of training data. The detail of a linear regression model is given below.

4.1 Linear regression model

The goal of N-best hypotheses reranking is to reorder the hypotheses by the number of errors each hypothesis contains, so that we can output the hypothesis that has the least number of errors. In Chapter 3, we studied a set of features that might be able to predict the number of errors in the hypothesis or on the contrary predict the goodness of the hypothesis. We chose linear regression as a method for combining features together and then predicting the quality of hypotheses. The linear regression model can be described as the following: let X represent a set of features and Y represent the number of errors (or hypothesis quality); then the training data is a set of tuples (X_1, Y_1), (X_2, Y_2),…, (X_n, Y_n) where n is the number of hypotheses in the training set. We would like to find the relationship between X and Y, so that we can predict a new *response* Y (number of errors or hypothesis quality) from a given new X (a set of features).

We focused on linear regression rather than other types of regression since the linear regression is considered simple but still powerful enough in many applications. The linear combination is also similar to the way that the speech recognizer combines acoustic model score and language model score together when it calculates the score of each hypothesis. Linear regression models the relationship between X and Y as the following:

$$Y_i = w_0 + w_1 x_1 + w_2 x_2 + \ldots + w_m x_m \tag{4.1}$$

where $w_0, w_1, \ldots, w_n$ are the weights, $x_1, \ldots, x_n$ are values of each feature and m is the number of features.

[1] Chapter 6 will discuss the definition of concept error rate and how to calculate it.

The weights are estimated by the least squares estimation that minimizes the prediction errors in the training data. More technical detail about linear regression and other types of regression can be found in many statistics books such as [DeGroot and Schervish, 1989] and [Venables and Ripley, 1998].

4.2 Feature representation

When we use only a single feature to rerank the hypotheses, the representation of the score, e.g. a linear scale or a log scale, doesn't make any difference to the reranking module since it simply selects the hypothesis that has the best feature score. However, when we combine many features together in a linear regression model, the difference in the order of magnitudes among combined feature scores might affect the trained weights if the difference is a great deal. We investigated some feasible feature representations as discussed below.

4.2.1 Raw scores

We represent each feature by the original representation that was used when the feature was extracted. The table below lists all the features with their score types and ranges.

Features/responses	Type of score	Min	Max
Speech recognizer score	Converted log base	-1082203967	-6801276
Acoustic model score	Converted log base	-1080249454	-6276909
Language model score	Converted log base	-10140338	-162770
N-best word rate	Number of hypotheses ratio	0	1
N-best homogeneity (SR)	Hypothesis scores ratio	-690809984	0
N-best homogeneity (AM)	Hypothesis scores ratio	-690809984	0
N-best homogeneity (LM)	Hypothesis scores ratio	-690809984	0
Coverage ratio	Minus number of errors	-100	0
Gap ratio	Minus number of errors	-100	0
Fragmentation transition	Minus number of errors	-100	0
Highest-in-coverage	Binary	0	1
Slot bigram	Probability in log base	-54.65	-0.81
Conditional slot	Probability in log base	-63.24	-0.09
Expected slots	Binary	0	1
Conditional slot bigram	Probability in log base	-63.05	-0.08
Word error rate	Percentage	0	1300
Concept error rate	Percentage	0	500

Table 4.1: Types and ranges of scores for each individual feature and response

It can be seen easily from Table 4.1 that the magnitudes of feature scores are very different. The values of features that derived from hypothesis scores (speech recognizer score, acoustic model score and language model score) are usually large negative integers since the Sphinx recognizer converts the log base probability score into an integer to speed up the calculation. When features are combined together in a linear regression model, the estimated weights of these features are much lower than the weights of other

features due to their large magnitudes. This problem cause the combined features model to perform poorly as shown in section 7.5.

Table 4.1 also shows types and ranges of linear regression model responses, word error rate and its alternative concept error rate. Both of them also have a wide range of values. The very high error rates are caused by the feedback noise of system prompts.

4.2.2 Linear scaling

In [Mendelsohn, 1993], a linear scaling method is used to normalize the data into a desired range before using it as an input of a neural network. Besides the need for comparable feature ranges to avoid the problem from order of magnitude differences, rescaling the data is justified for the following reason. In the context of hypothesis reordering, we would like to select the hypothesis that has the lowest relative error rate among the set of N-best hypotheses of a particular utterance without considering the absolute error rate values.

Reranked order	**1**	**2**	**3**	**4**	**5**
Utterance #1	20%	25%	25%	30%	50%
Utterance #2	5%	10%	10%	20%	20%

Table 4.2: Sample word error rates of two 5-best hypotheses sets

For example, consider two sets of hypotheses in Table 4.2. The best hypothesis for utterance #1 is the one that has the word error rate of 20%. However, in utterance #2, the hypothesis with the word error rate of 20% is not the one that we would like to select. Therefore, in the reranking task, relative goodness is more important than absolute. This assertion also applies to the value of each feature.

We choose the rescaling range of 0 to 1. The best feature value and the lowest word error rate are scaled to 1 while the lowest feature value and the highest word error rate are scaled to 0. With this rescaling scheme, the best values for both the features and the response in each set of N-best hypotheses are always 1 while the worst values are always 0. The transformation is done by the following equation.

$$D_{new} = R_{\min} + (R_{\max} - R_{\min}) * \frac{(D - D_{\min})}{(D_{\max} - D_{\min})} \tag{4.2}$$

where D_{new} is a rescaled value
R_{min} and R_{max} is the rescaling range, which is 0 and 1 respectively
D is the raw score
D_{min} and D_{max} are the minimum value and the maximum value in an N-best hypotheses set respectively

We rescale all the features whose original ranges are not [0,1] and also rescale the regression model responses, word error rate and concept error rate. The linear regression model is trained on the rescaled features to predict the rescaled response.

4.2.3 Linear scaling with clipping

This normalization technique doesn't only rescale the data to a desired range, but also removes outliers and spreads out the data distribution. The method is based on a

statistical measure of central tendency and variance. Instead of using D_{min} as the minimum data value and D_{max} as the maximum data value, linear scaling with clipping determines the new data range [ND_{min}, ND_{max}] by the following equations.

$$ND_{\min} = \mu - n * std \tag{4.3}$$

$$ND_{\max} = \mu + n * std \tag{4.4}$$

where μ and *std* are the mean and the standard deviation of the values in an N-best hypotheses set respectively; *n* is a small integer.

The outliers are clipped as follows: the data values that are less than ND_{min} are set to ND_{min}, similarly, the data values that are greater than ND_{max} are set to ND_{max}. We chose *n*=2 for our data. So, the outliers are the data values outside the range of the mean +/- two standard deviations. By replacing the original range [D_{min}, D_{max}] with the new range [ND_{min}, ND_{max}] in equation 4.2, the transformation is performed similarly to the standard linear rescaling.

4.3 Optimizing the feature combination model

When we combined all features together using a linear regression model, we gained only marginal improvement over the baseline as shown in section 7.5.1. Furthermore, some of feature weights are negative which is counter-intuitive. These outcomes suggest that some of the features may be redundant. For this reason, combining only a subset of features may lead to a more efficient model. In this section, we describe 3 techniques: stepwise regression, greedy search and brute force search that we used to discover the optimal model in model space.

4.3.1 Stepwise regression

Both statistical packages S-Plus and R provide the model searching strategy called *stepwise regression*. Stepwise regression uses a greedy search to find the best model in given model space. At each iteration, stepwise regression adds or drops one feature in the model depending on the search direction, *forward* or *backward*. For example if we start from a null model or a uniform model, where the response does not depend on any feature, with the forward direction, the stepwise regression conducts the search by adding one feature at each iteration. In the first iteration, every feature is examined and the best single feature model is selected. In the next iteration, the feature that makes the best combination with the selected model is chosen. The search continues until the best model in the current iteration is not better than the best model in the previous iteration.

The goodness of each model is measured by *penalized likelihood* which balances the likelihood of the model with respect to the data with the model complexity (number of parameters). Penalized likelihood has the following form:

$$Score = -2 * \log(likelihood) + k * npar \tag{4.5}$$

where *npar* is the number of parameters in a model.

When *k* is set to 2, the score is called *AIC* (Akaike Information Criterion). When k is set to *log(n)*, with n equal to the number of hypotheses in our problem, the score is

called *BIC* or *SBC* (Schwarz's Bayesian criterion). The stepwise regression use a greedy search strategy to find the model with the lowest AIC or BIC value.

We investigated stepwise regression in both directions.

1) *Forward direction*

   We start the search from a null model or a uniform model, where the response does not depend on any feature. Then we iteratively add one feature in each step until the local minimum AIC or BIC is found or the all-features model is reached.

2) *Backward direction*

   In contrast, with a backward direction, we start from the model that combined all features together. We then remove one feature iteratively in each step until the local minimum AIC or BIC is found or the null model is reached.

The results of stepwise regression models are discussed in section 7.5.2.

4.3.2 Greedy search

As shown by the results in section 7.5.2, the stepwise regression, which uses penalized likelihood as a search criterion, wasn't successful in discovering the optimal model in the model space. The penalized likelihood still has a bias toward a complex model for our training data. Therefore, we changed the criterion that guided a greedy search from the regression model statistic to the error rate which makes the new greedy search directly optimizes our objective performance.

The greedy search starts by first finding a single feature model that has the lowest error rate. This model is the best model for the first iteration. In the next iteration, we add one feature to the best model from the previous iteration. All features that haven't been considered are tried. The new model that has the lowest error rate is selected as the best model for that iteration. One feature is added to the selected model at each iteration until no more improvement on the error rate can be achieved. The result of the greedy search is discussed in section 7.5.2.

4.3.3 Brute force search

The solution from a greedy search is a local optimal model not the global one due to the nature of the search. In order to find the best combination of the features, we conducted a brute force search over the entire model space. This method requires a lot of computation, but it is guaranteed to reach the optimal model. The objective function that we tried to minimize in the brute force search is the error rate as in the greedy search. We discuss the result of the brute force search in section 7.5.2.

Chapter 5
Utterance Selection

In the training data, for 91.0% of the utterances, the most correct hypotheses are already in the first ranks. Therefore, only 9.0% of the utterances need to be reranked. Since the reranking model is still not perfect, reranking only utterances that are likely to benefit from hypotheses reordering can reduce a lot of computational cost and may also improve the accuracy. Based on this idea, an utterance selection module was added to the reranking process prior to the reranking module that was discussed in Chapter 4. The reranking process with an utterance selection module, or a classifier, is illustrated in Figure 5.1.

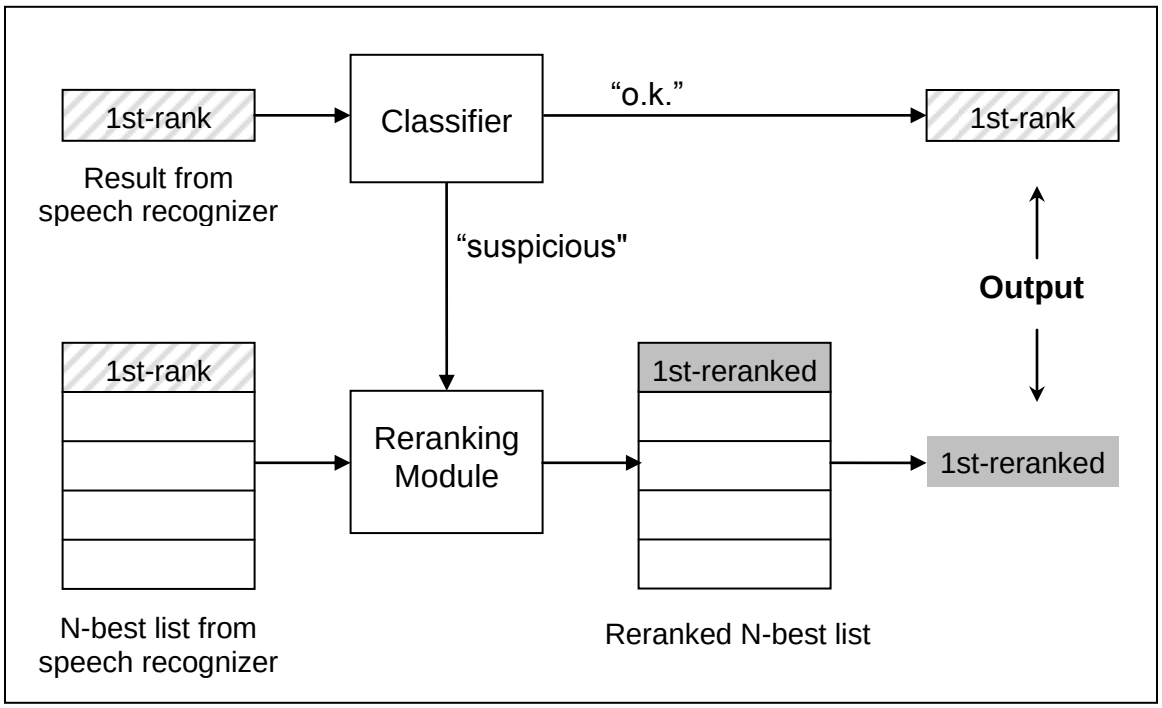

Figure 5.1: Reranking process diagram

The reranking process begins with a selection module. A selection module determines, based on the information in a speech recognizer's first rank hypothesis, whether that utterance needs to be reranked or not. If the classifier classifies an utterance as an '*o.k.*' utterance, no reranking needs to be done. The reranking process simply outputs the recognizer's original first rank hypothesis. On the other hand, if the classifier classifies an utterance as a '*suspicious*' utterance, that utterance is sent to a reranking module. A reranking module will rerank the N-best hypothesis list and then output the first rank of the reranked N-best list.

In this chapter, we will focus on utterance classification techniques. As shown in Figure 5.1, we need to classify each utterance as an 'o.k.' or 'suspicious' utterance. Ideally, a suspicious utterance is an utterance where the first rank hypothesis is not the most correct one. In Chapter 3, we investigated 15 features that have potential to predict

hypothesis quality. We can use these features to classify an utterance as ‘o.k.’ or ‘suspicious.’ Two utterance selection techniques, single feature classification and classification tree, are considered.

5.1 Single feature classifier

For simplicity, we started by using each feature individually as a classifier. The classifier is a simple rule of the form below.

```
if feature value < threshold  then
    classify as ‘suspicious’
else
    classify as ‘o.k.’
endif
```

Figure 5.2: Single feature classifier

Two types of feature values can be used as classification features, *first rank score* and *difference score*.

5.1.1 First rank score

One source of information that we can use to classify the first rank hypothesis as ‘o.k.’ or ‘suspicious’ is the feature values of the first rank hypothesis itself. If the first rank hypothesis has relatively low values for some of the features, it is suspicious that the first rank hypothesis is the most correct hypothesis. For example, if a first rank hypothesis that has a low language model score is usually not the most correct one, we can use the following classification rule in an utterance selection module.

```
if language_model_score < 0.5  then
    classify as ‘suspicious’
else
    classify as ‘o.k.’
endif
```

Figure 5.3: Single feature classifier that has language model score as a criterion

We examined 10 threshold values for each feature. The thresholds are chosen by dividing the range of each feature value into 10 equal intervals. For most of the features, we used {0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0} as a set of thresholds. For the speech recognizer score feature, a different set of thresholds, which is {0.55, 0.60, 0.65, 0.70, 0.75, 0.80, 0.85, 0.90, 0.95, 1.00}, was used since the range of the score is smaller. For the binary features, expected slots and highest-in-coverage, only one threshold value is needed.

We can evaluate an utterance selection module separately from a reranking module by evaluating the performance of a classifier. In the classification task, the following three metrics: *precision*, *recall* and *F-1*, are commonly used to measure the performance of the classifier.

1) *Precision* measures correctness of a selection.

$$precision = \frac{Number\ of\ suspicious\ utterances\ classified\ as\ suspicious}{Total\ number\ of\ utterances\ classified\ as\ suspicious} \quad (5.1)$$

2) *Recall* measures completeness of a selection.

$$recall = \frac{Number\ of\ suspicious\ utterances\ classified\ as\ suspicious}{Total\ number\ of\ suspicious\ utterances} \quad (5.2)$$

3) *F-1* is a harmonic mean of precision and recall.

5.1.2 Difference score

Besides the feature value of the first rank hypothesis, the difference between the feature value of the first rank hypothesis and the feature value of the second rank hypothesis (*difference score*) can provide some useful information. The range of the difference score is [–1, 1]. If the difference is small or the feature value of the second rank hypothesis is even better, this might be a good indicator that we should rerank this utterance. In contrast, if the difference is large, namely the score of the first rank hypothesis is much better, this may indicate that the first rank hypothesis is already good. The classifier is a rule of the following pattern.

```
if (1st rank feature value – 2nd rank feature value) < threshold then
    classify as 'suspicious'
else
    classify as 'o.k.'
endif
```

Figure 5.4: Single feature classifier that uses a difference score as a criterion

The result of an utterance selection module that uses a single feature classifier, both with first rank score and difference score, is given in section 7.6.1.

5.2 Classification tree

From the results presented in section 7.6.1, we found that reranking only suspicious utterances selected by a single feature classifier didn't result in better word error rate or concept error rate than reranking every utterance. Moreover, using a predefined threshold as a selection criterion is quite subjective. We had experience from the reranking task that the performance of combined features is better than the performance of those features individually. This may also apply for the utterance selection task.

One algorithm that is well known for solving the classification problem is a *Classification* or *Decision Tree*. In [Carpenter *et al*., 2001], a classification tree is the second most efficient algorithm for the utterance level confidence annotation task. The classification tree that we used to classify each utterance as 'o.k.' or 'suspicious' is a common tree method called *Classification and Regression Tree* (CART) that comes with the statistical package, R. A simplified version of a classification tree uses a binary tree to classify the utterances as follows. At each step, the utterances at each leaf node of the tree are divided into two groups by a rule of the following pattern: *feature value* < *threshold*. For each utterance, if the criterion is true, the utterance is classified into the first group; otherwise, it is classified in to the second group. The classification of each group is assigned such that the misclassification rate is minimized. This is also the same as a majority vote among the members of the group. The pair of feature and threshold that is selected for each step is the pair that also optimizes the misclassification rate. The algorithm is repeated until the stopping criterion is met. One stopping criterion that is commonly used is the size of the leaf node. For example, we won't consider splitting the leaf node if it contains less than 5 utterances. Technical details of a classification tree in general can be found in many machine learning books such as [Mitchell 1997].

The rule that is used to split each node is similar to the rule that is used in the single feature classifier. However, in the single feature classifier, we make an assumption that an utterance with a higher feature value is better and should be classified as 'o.k.' Whereas in a classification tree, a classification of each group is data-driven.

We avoid the 'over-fitting' problem by pruning the classification tree. With the k-fold cross validation technique, the data are divided into a training set and a validation set. The initial tree is trained from the training set. Then a separate validation set is used to prune the tree by removing node splitting the initial tree if it can improve the misclassification rate on the validation set. Tree pruning results in a smaller, less complex tree that also performs better on a separate validation set. The result of a classification tree on an utterance selection task is shown in section 7.6.2.

Chapter 6
Concept Error Rate

Word error rate, which is a commonly used metric for evaluating the performance of a hypothesis reordering module, is the metric that measures the accuracy of the output word strings. However, in a dialog system, the semantic representation of a user utterance generated by a semantic parser is used instead of the surface form. A semantic parser is able to map the surface variations of the same semantic concept into the same semantic representation, at the same time ignoring unimportant words (or non-concept words) in the domain, for example, social pleasantries. As a result, surface word variations of the same concept such as 'yes' and 'yeah', and recognition errors in non-concept words, do not affect the performance of a dialog system.

To measure the performance of a reranking module with a metric that better correlates with the performance of a dialog system, the concept error rate was considered as an alternative evaluation metric. Concept error rate compares the output parses of the hypothesized utterance to the output parses of the reference instead of comparing the surface words. In a hypotheses reranking task, the hypothesized utterance is the output of the reranking module. The output parse is a series of semantic frames; each frame is composed of a slot, sub-slots and their values as described in section 1.3[1]. However, not every component in the parse is considered by a dialog manager. We categorized slots into three categories according to the amount of information in the slots that a dialog manager actually uses.

1) *Non-concept slot* is a slot that contains information that, while captured by the grammar, is not considered relevant for selecting a system action. Politeness expressions, such as 'please', are examples.
2) *Value-insensitive slot* is a slot whose identity, rather than specific value, is sufficient to drive a system action. An example would be *[_yes]*.
3) *Value-sensitive slot* is a slot for which both the occurrence and the value of the slot are important, for example *[City_Name]*.

For different types of slots, different pieces of information are converted into concepts. The entire non-concept slot is discarded. For a value-insensitive slot, only its label is converted to a concept, while for a value-sensitive slot, both its label and value are converted to a concept. Since the definition of the *concept* is not as well-defined as the definition of the *word* in word error rate, we can think of the definition of the concept in different levels of granularity. One may think of a concept as a big piece of information that can drive a complete system action. On the other hand, it is also possible to think of a concept in a finer grain. In this case, a concept is equivalent to a piece of information that a dialog manager considers at a time, but the information may not be completed enough to drive a unique system action. We discuss two different concept definitions, *frame-level concept* and *path-level* concept, and give some concrete examples below.

[1] In this chapter, we don't differentiate between a slot and a sub-slot since both of them contain similar information for the concepts. We use the term 'slot' to refer to both the slot and the sub-slot.

6.1 Frame-level concept

In frame-level concepts, each semantic frame in the output parse is considered as one concept. A semantic frame contains all information required for driving a system action. For each semantic frame, information in the slots are extracted according to their types and then merged together into a concept. This concept definition is similar to the concept mentioned in [Chotimongkol and Rudnicky, 2001]. A frame-level concept ranges from a simple concept that containing only one slot label to a complex concept contains several slot labels and their values. The examples of frame-level concepts in an air travel domain are given in Figure 6.1.

"*No*"
Output parse: Respond:[_no] (NO)
Concept: [_no]

"*Leave from Pittsburgh*"
Output parse:
Reserve_Flight:[departloc] (LEAVE FROM [Depart_Loc] ([Location] ([city] ([City_Name] (PITTSBURGH)))))
Concept: [departloc][City_Name](PITTSBURGH)

"*Tomorrow around seven a.m.*"
Output parse:
Reserve_Flight:[Date_Time] ([Date] ([Today_Relative] (TOMORROW)) [Time_Range] ([time_spec] ([_aprx_time] (AROUND [Time] ([Hour] (SEVEN) [periodofday] ([Period_Of_Day] ([_am] (A M))))))))
Concept: [Date_Time][Today_Relative](TOMORROW)[Time_Range] [_aprx_time][Hour](SEVEN)[Period_Of_Day][_am]

Figure 6.1: Examples of frame-level concepts in an air travel domain

A frame-level concept for a 'no' response is very simple, containing only one slot label *'[_no]'*, while a frame-level concept for a date-time expression is very complex, containing seven slot labels and two slot values. In the first example, the only slot in the semantic frame is a value-insensitive slot; therefore, only its label is used to represent the concept. In the second example, *[departloc]* is a value-insensitive slot while *[City_Name]* is a value-sensitive slot. The rest of the slots are non-concept slots. Thus, only two slot labels and one slot value are used to represent the concept. In the last example, five slots in the parse, *[Date_Time]*, *[Time_Range]*, *[_aprx_time]*, *[Period_Of_Day]* and *[_am]*, are value-insensitive slots while two slots, *[Today_Relative]* and *[Hour]*, are value-sensitive slots. As a result, the concept for this parse contains seven slot labels and two slot values. We note that many slots in the

middle of the parse are categorized as non-concept slots since these slots are not critical to a dialog manager and used solely in a parser.

From the examples of frame-level concepts in an air travel domain, we found that some semantic frames, such as the one about date and time, contain many pieces of information that are significant to a dialog system. Consequently, a concept generated from an entire semantic frame can be overly complex. If there is an error in just one piece of information in the semantic frame, the entire concept is incorrect, even if all other pieces of information are correct. To illustrate this, let us consider the last concept in Figure 6.1 as a reference concept. Suppose that a reordering module selects the hypothesis '*Tomorrow seven a.m.*" as the best reordered hypothesis. The frame-level concept for this hypothesis is "*[Date_Time][Today_Relative](TOMORROW) [Time_Range][Hour](SEVEN)[Period_Of_Day][_am]*." The reordered hypothesis concept is considered wrong even though only the part about approximated time is missing.

6.2 Path-level concept

Path-level concept defines a concept in a finer level of granularity than frame-level concept. In order for a dialog manager to determine its appropriate action, sometimes many pieces of information from a user are required. For instance, a date-time expression is composed of the following slots: *[Day_number]*, *[Month_Name]*, *[Hour]*, *[Minute]*, etc. Errors in some of the components may cause the perceived expression to be inaccurate. However, the concept is still perceived as a date-time expression with some correct information. Therefore, it should get some credit for being partially correct.

Some information in the parse is additive, namely being captured together in the same frame or being captured separately don't change the meaning of each individual bit of information. Consider the frame-level concept "*[Date_Time][Today_Relative]-(TOMORROW)[Time_Range][Hour](SEVEN)[Period_Of_Day][_am]*", these three pieces of information: *[Today_Relative](TOMORROW)*, *[Hour](SEVEN)* and *[_am]* are additive. On the other hand, some components in the parse are required to occur together in order to determine a correct system action. Being captured separately could change the meaning of each piece of information. For instance, *[departloc]* and *[City_Name]-(PITTSBURGH)* have to occur together in order to determine the departure city. We call a set of parse components that are required simultaneously a *conjunction rule*.

We define conjunction rules according to a dialog manager, how it uses the information from the parse. Normally, a conjunction rule is equivalent to a path in a parse tree. Therefore, for this reason, we call this a path-level concept. The information piece that are additive are separated into different conjunction rules. Each conjunction rule is converted into a concept. If a semantic frame contains additive information, more than one conjunction rule is matched. As a consequence, that semantic frame is converted into multiple path-level concepts. Figure 6.2 gives examples of path-level concepts for the same date-time expression as the last example in Figure 6.1.

[Today_Relative](…), *[_aprx_time]*, *[Hour](…)* and *[Period_Of_Day][_am]* are additive information. Thus, there is a separate conjunction rule for each piece of information. An output parse matches four conjunction rules in the list of conjunction rules. Therefore, it generates four concepts instead of one big concept as in the frame-level concept.

"*Tomorrow around seven a.m.*"
Output parse:
Reserve_Flight:[Date_Time] ([Date] ([Today_Relative] (
TOMORROW)) [Time_Range] ([time_spec] ([_aprx_time] (
AROUND [Time] ([Hour] (SEVEN) [periodofday] (
[Period_Of_Day] ([_am] (A M))))))))
Conjunction rules:
[Date_Time][Today_Relative](…)
[Date_Time][_aprx_time]
[Date_Time][Hour](…)
[Date_Time][Period_Of_Day][_am]
…
Concept:
[Date_Time][Today_Relative](TOMORROW)
[Date_Time][_aprx_time]
[Date_Time][Hour](SEVEN)
[Date_Time][Period_Of_Day][_am]

Figure 6.2: Examples of path-level concepts for a date-time expression

We also made the following changes to make the concept error rate metric better correlated with the performance of a dialog system.

- Replace concepts that cause the same action in a dialog manager with the same concept labels.

  For example, [arriveloc][Airport_Name](DULLES) and [connect_loc][Airport_Name](DULLES) are considered the same by a dialog manager. Hence, we replaced [arriveloc][Airport_Name] and [connect_loc][Airport_Name] with the same new label [All_Airport_Name][Airport_Name]. As a result, these two concepts are considered the same concept.
- Collapse consecutive concepts that are identical.

  The amount of times that each concept occurs doesn't affect the system action. However, the order of concepts is still important since the concept that comes after may overwrite the concept that occurs before. Therefore, we just collapsed the repeated concepts while still preserving their order.

The calculation of concept error rate for both a frame-level concept and a path level concept is the same. Each concept is considered as one token equivalent to a word in the calculation of word error rate. Therefore, the algorithm for calculating word error rate can be applied to calculate concept error rate without any modification as in [Boros *et al*., 1996]. From the example in Figure 6.3, the word error rate of the reordered hypothesis is 50% (one substitution of 'yes' with 'yeah' and one deletion of 'please'). However, when we converted the output parses of both the transcript and the reordered hypothesis to concepts and calculated concept error rate, the concept error rate of the reordered hypothesis is 0%. One of the word errors occurred with the non-concept word

'please.' Another word error is a substitution of 'yeah' for 'yes'. However, 'yeah' is a variation of the word 'yes' and is converted to the same concept *[_yes]*. Therefore, both word errors don't affect the concepts of the reranked hypothesis. Since these two utterances have the same meaning, the concept error rate is a better measurement. We noted that both frame-level concepts and path-level concepts are identical for utterances in Figure 6.3.

Transcript: *"Yes to Boston please"*
Output parse:
Respond:[_yes] (YES)
Reserve_Flight:[arriveloc] (TO [Arrive_Loc] ([Location] ([city] (
[City_Name] (BOSTON)))))
Reserve_Flight:[Polite] (PLEASE)
Concept: [_yes], [arriveloc][City_Name](BOSTON)

Reordered hypothesis: Yeah to Boston
Output parse:
Respond:[_yes] (YEAH)
Reserve_Flight:[arriveloc] (TO [Arrive_Loc] ([Location] ([city] (
[City_Name] (BOSTON)))))
Concept: [_yes], [arriveloc][City_Name](BOSTON)

Figure 6.3: The transcript concepts and the reordered hypothesis concepts

The advantage of the path-level concept is that a complex semantic frame gets some credit for partially correct information rather than getting nothing as in the frame-level concept. In the first example in Figure 6.4, the concept error rate of the path-level concept is lower because the "*Time_Range*" frame is partially correct. However, if the slot label close to the root node of a parse tree is incorrect, the number of errors in the path-level concept can be increased significantly since all the paths starting from the error point are incorrect. For instance, in the second example in Figure 6.4, there is an error at the slot label of the root node of the parse. The reranked hypothesis contains the slot label *[Time_Range]* instead of *[Date_Time]*. Therefore, all the concepts for a date-time expression are incorrect, both frame-level concepts and path-level concepts. Since there are more date-time concepts in the path-level concept, its concept error rate is higher.

The concept error rate still has some disadvantages. Categorizing slots into one of the three categories, non-concept, value-insensitive and value-sensitive slots, is subjective and depends a lot on the dialog manager. The conversion from output parses to concepts is not well defined and is also dialog manager dependent. Moreover, for path-level concepts, we have to decide which piece of information is additive and what should be conjunction rules. It is quite difficult to identify the information that is really used in a dialog manager and actually has an influence on a system action. One tedious solution is to look through the entire dialog manager code and identify information that effects system actions. If there is any change in the dialog manager, we may need to revise the parse-to-concept conversion. Furthermore, since concept error rate compares the output parse of the reordered hypothesis to the output parse of the transcript, the errors from a parser can make concept error rate inaccurate.

Transcript: "*How 'bout something at around eight a.m.*"

Reordered hypothesis: How 'bout something at around a.m.

Frame-level concept
Transcript concept:
[Time_Range][_aprx_time][Hour](EIGHT)[Period_Of_Day][_am]
Reordered hypothesis concept:
[Time_Range][_aprx_time][Period_Of_Day][_am]
Concept error rate=1/1=100%

Path-level concept
Transcript concept: [Time_Range][_aprx_time]
[Time_Range][Hour](EIGHT)
[Time_Range][Period_Of_Day][_am]
Reordered hypothesis concept: [Time_Range][_aprx_time]
[Time_Range][Period_Of_Day][_am]
Concept error rate =1/3=33.3%

Transcript: "*I'd like to go to Boston tomorrow morning"*

Reordered hypothesis: I'd like to Boston file morning

Frame-level concept
Transcript concept:
[arriveloc][City_Name](BOSTON)
[Date_Time][Today_Relative](TOMORROW)[Time_Range][Period_Of_Day][_morning]

Reordered hypothesis concept:
[arriveloc][City_Name](BOSTON)
[Time_Range][Period_Of_Day][_morning]
Concept error rate =1/2=50%

Path-level concept
Transcript concept: [All_City_Name][City_Name](BOSTON)
[Date_Time][Today_Relative](TOMORROW)
[Date_Time][Period_Of_Day][_morning]
Reordered hypothesis concept: [All_City_Name][City_Name](BOSTON)
[Time_Range][Period_Of_Day][_morning]
Concept error rate =2/3=66.7%

Figure 6.4: The comparison between frame-level concepts and path-level concepts in terms of concept error rate

Chapter 7
Experiments and Discussion

In this chapter, we report the experimental results for all the reranking techniques that we have discussed so far. We also discuss some of the interesting results that we obtained.

7.1 Experimental procedure

All the experiments, except when explicitly mentioned, were conducted based on the following setting. All the training and test data are based on the utterances collected from the Communicator system. The N-best hypothesis lists were generated in batch mode from recorded speech files using the Sphinx-II speech recognizer. The language models used in the recognizer are state-dependent class based trigram models. Each language model was trained separately on the transcription of user utterances that correspond to each dialog state. The state-dependent language model is similar in principle to the state-conditional slot N-gram model discussed in section 3.3.3. The hypotheses in the N-best list were passed to the Phoenix parser and the output parses were used to generate parser features and features that make use of semantic slots. The output parses were also used in the calculation of concept error rate. The dialog state of each system utterance was extracted from a dialog manager log for calculating dialog manager features. Finally, after information from all knowledge sources was extracted, all the features that were discussed in Chapter 3 were generated for each hypothesis. In each N-best list, if two or more hypothesis strings were identical after noise and fillers were removed, only the hypothesis with the highest speech recognizer score was kept. However, we still maintained the number of times the same hypothesis string occurs in the N-best list as a hypothesis count.

Our training data contains 39,377 utterances collected from the CMU Communicator system during the period of June 1998 to May 1999. The training set was divided into two parts by assigning each dialog session randomly into one of the two sets. Utterances from the same session were in the same set. The first part, which contained 19,757 utterances, was used to train a slot N-gram model, a conditional slot model and state-conditional slot N-gram models. The second part, which contained 19,059[1] utterances, was used by the S-PLUS software version 4.5 and the R software version 1.4.1 (the free license version of S-Plus) to train the feature weights in a linear regression model. The test set is composed of 1,679 utterances from the Communicator data in June 1999.

Ten-fold cross-validation was employed in the training process in order to perform a significance test and select the optimal model among the variations of each feature or among different feature combinations. The training data was divided into ten subsets of equal size. We trained a linear regression model on nine subsets and then

[1] 561 utterances in which the reference transcriptions were null strings (silence or all noise utterances) or null concepts, i.e. out of domain utterances, were removed from the set. Those utterances accounted for 2.86% of the set.

evaluated it on the subset that was left out (the validation set). We alternated the validation set among the ten subsets. Thus, there were ten regression models trained on ten different training subsets (but not exclusively different) and evaluated on ten different validation sets. The result of the ten-fold cross-validation is usually reported in terms of an average word error rate or an average concept error rate of the ten validation sets. The concept error rate was computed based on the path-level concept definition discussed in Chapter 6. It is considered unbiased to choose the model that has the optimal performance on the cross-validation since the training subsets in the ten-fold cross-validation made no use of the data from the validation set.

Apart from using an average word error rate or an average concept error rate of the ten-fold cross-validation results to compare the performances of two reranking models, we also performed a significance test to verify that the difference between the average error rates was statistically significant. We used the Wald test described in [Wasserman, 2002] to test the different between the means of two reranking models. We chose this hypothesis testing mechanism instead of the t-test, which can also be used for the same purpose, because the Wald test makes fewer assumptions about the distribution of the data, in this case the error rate. (We did some preliminary experiments using both the Wald test and the t-test for a significance test. We found that the results from both techniques were similar.) The significance tests were conducted at 0.05 level of confidence.

We used the word error rate and concept error rate of the recognizer output, or the first rank hypothesis in an N-best list before reranking, as the baseline performance of our hypotheses reranking task. The average baseline word error rate on the ten-fold cross-validation is 11.45% while the average baseline concept error rate is 9.43%.

7.2 N-best List Size

From our preliminary experiment, we found that increasing the number of hypotheses in the N-best list, namely using the higher number of N, didn't help to improve the performance of the reranking module much, even for the oracle (selecting the hypothesis that has the lowest word error rate). The graph in Figure 7.1 shows that after N equal to 5, the oracle word error rate decreases just slightly. By increasing the value of N from 5 to 25, the oracle word error rate decreases by only 1%.

The linear regression model that was used in this experiment was trained to predict word error rate from the combination of 7 features: speech recognizer score, acoustic model score, language model score, unparsed words, gap number, slot bigram and conditional slot. All the features and the word error rate response were normalized using the linear rescaling method described in section 4.2.2. The model was trained on 5,000 utterances selected from the June98-May99 training set and evaluated on about 1,700 utterances from the June99 test set. A separated linear regression model was trained for each N-best list size (each value of N). The reported word error rate in the graph was evaluated on the test set with the same N-best list size[1]. We can see from the graph that the accuracy of the linear regression models degrades when the number of

[1] The word error rate of Jun-99 test set in the preliminary experiment is 12.46% This number is different from the number in the main experiments dues to the changes in the speech recognizer parameters, such as using newer version of language model and acoustic model. In the main experiments, the word error rate of Jun-99 test set is 11.45%.

hypotheses in the N-best list is large since the hypotheses toward the end of the list are not confidently generated by a speech recognizer and are more likely to contain noise or misrecognized words. We decided to consider only the top-5 hypotheses from the list of 25-best hypotheses in the hypotheses reranking task because at N equal to 5, the performance of the linear regression model is about the same as the optimal performance. Moreover, using a smaller value for N can reduce the computational complexity.

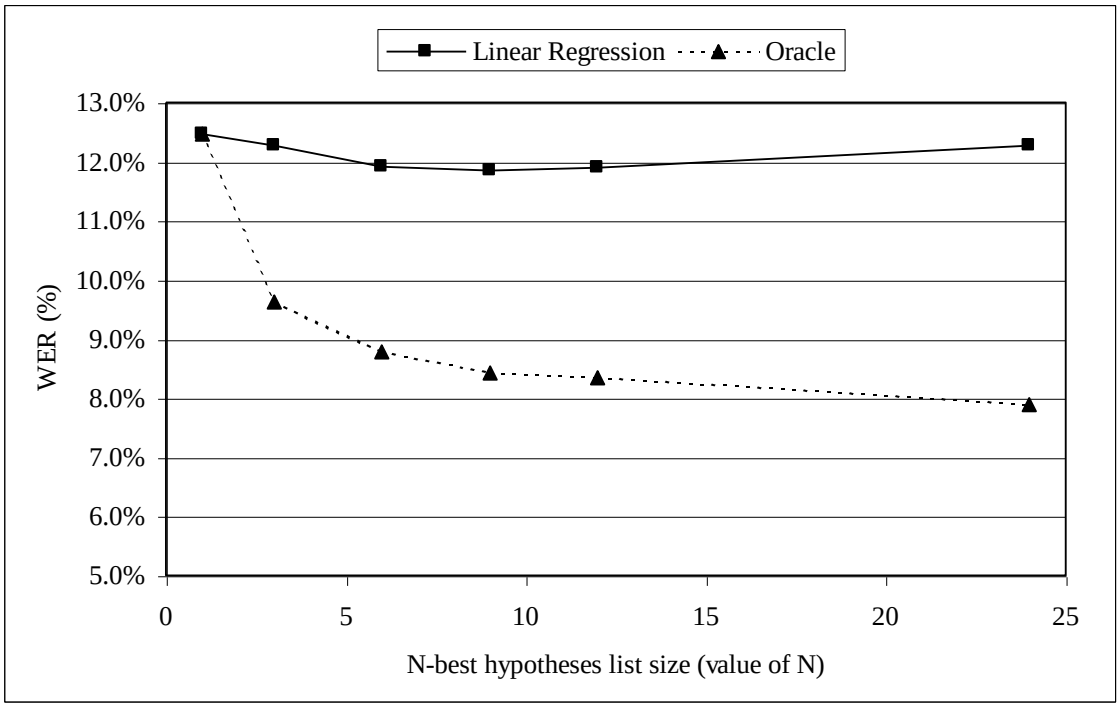

Figure 7.1: Word error rates of a linear regression model and the oracle when varied the size of an N-best list

7.3 Human subjects on hypotheses reranking task

We evaluated the performance of human subjects on the same reranking task for two purposes. The first one is to gain the insight into the type of knowledge that a human use to select the most correct hypothesis out of a set of N-best hypotheses similar to the experiment in [Brill *et al.*, 1998]. The second purpose is to compare our reranking model to human ability.

The experiment was constructed as follows. There were 18 subjects participated in the experiment; half of them (9 people) are native speakers of English and another half are non-native speakers. Each human subject was given the lists of 5-best hypotheses of 30 utterances that were randomly selected from the Jun-99 test set and had a comparable word error rate to the word error rate of the entire test set[1]. The system prompt that was spoken before each user utterance was also given. The subjects were told that hypotheses in a 5-best list were ranked in the order of speech recognizer confidence (the hypothesis with the highest confidence was in the first rank), but audio files were not provided. The

[1] The word error rate of Jun-99 test set in the preliminary experiment is 12.46% as mentioned in section 7.2.

subjects were asked to select the most appropriate hypothesis from each N-best list. They were allowed to select more than one hypothesis if they were equally appropriate. They were also allowed to arbitrary edit the hypotheses if they thought that none of the given hypotheses was correct. For each utterance, the subjects were also asked to provide information sources that they used in order to make a reranking decision. A sample question from the human experiment is shown in Figure 7.2.

System: What time do you want to travel?
User:

1. I'D LIKE TO GO HUNG AROUND NOON PLEASE
2. I'D LIKE TO GO AROUND NOON PLEASE
3. I'D LIKE TO GO HOME HUNG AROUND NOON PLEASE
4. I'D LIKE TO GO ON HUNG AROUND NOON PLEASE
5. NO I'D LIKE TO GO HUNG AROUND NOON PLEASE

__ None of them. What do you think they said? _____
__ I can't tell.

What kind of information did you use?
__ Syntax or grammar
__ Relationship with a system prompt
__ Correct/incorrect word occurs in hypothesis. Which word _____
__ Appropriateness or naturalness
__ Semantic or meaning
__ World knowledge or domain knowledge
__ Utterance was complete/incomplete
__ Other _____

Figure 7.2: A sample question from the human experiment

The Table 7.1 shows the type of knowledge that human subjects used to rerank N-best hypothesis lists. The most frequent knowledge type that human subjects reported using is syntax. Syntactic knowledge is included in the reranking model in term of parse quality features: unparsed words, gap number, fragmentation transitions and highest-in-coverage. The third most frequent knowledge type, relationship with a system prompt, is captured by features that utilize information from a dialog state: expected slots, conditional slot and conditional slot bigram. Other type of knowledge used by human subjects such as correct/incorrect word is modeled implicitly by the language model feature and also by N-best word rate and N-best homogeneity scores. However, some type of knowledge such as naturalness (which ranked second), semantic and world/domain knowledge are difficult to implement even not impossible. We might be able to say that the features that incorporate information from the semantic slots, slot bigram and all dialog manager features, implicitly include semantic information from semantic parser. Nevertheless, the level of semantic information provides in semantic parses is less sophisticated than what human subjects appear to use.

Knowledge Type	% of Utterances
Syntax	51.30%
Naturalness	30.93%
Relationship with a system prompt	29.44%
Semantic	13.52%
Correct/incorrect word	12.78%
World/domain knowledge	7.59%
Completion of utterance	5.00%
Other	2.59%

Table 7.1: The type of knowledge that human subjects used to rerank N-best hypothesis lists[1]

The linear regression model that was used in this experiment is a preliminary model similar to the one that was used in the N-best size list experiment discussed in section 7.2. The model was trained on the entire June98-May99 training set. Since the human subjects didn't edit every utterance in the set, the reranked hypotheses of human editing were composed of 1) new hypotheses (when the subject arbitrary edited the hypotheses) and 2) original hypotheses (when the subject thought that some of the hypotheses were already correct). The oracle word error rate model is the model that outputs the hypothesis that has the lowest word error rate for each utterance. Similarly, the oracle concept error rate model is the model that outputs the hypothesis that has the lowest concept error rate for each utterance. Even though the oracle word error rate model has the lowest word error rate, its concept error rate is not necessary the lowest one. This is also the case for the word error rate of the oracle concept error rate model. The graph in Figure 7.3 shows the result of the experiment. The relative improvement of each reranking approach when compares to the recognizer baseline is given in Table 7.2.

We analyzed the result by first looking at the conventional metric, word error rate. The performances of native and non-native speakers are about the same. The average word error rate of native speakers is a little higher than the average of non-native speakers, but not significantly higher. This counter-intuition result was caused by the high word error rate of native speakers who are not familiar with the task. However, when we looked at the word error rate of human editing, native speakers did better than non-native speakers. The word error rate of native speaker editing is the best among all reranking methods other than the oracle. On the other hand, non-native speakers editing is a little worse than non-native speakers selecting. We also note that our preliminary linear regression model performs better than human selection even not significantly better. This achievement may come from the regression model advantage of using acoustic features in the model while the human subjects didn't have a chance to listen to

[1] Since the subjects could report using more than one knowledge source together to determine the most appropriate hypothesis, the percentage of utterances reported in the table may not sum to 100.

recorded speech. However, when native speakers were allowed arbitrary editing they could gain higher word error rate reduction.

The word error rates on human selecting and editing are higher than expected. One possible reason is that the human subjects didn't always select the hypotheses that had the lowest word error rate, but rather selected the ones that he/she believed having the right meaning. The humans tend to look at the meaning or the concept of the utterance rather than surface words. The criterion used by the humans when they reranked the hypotheses supports the use of concept error rate as an alternative evaluation metric.

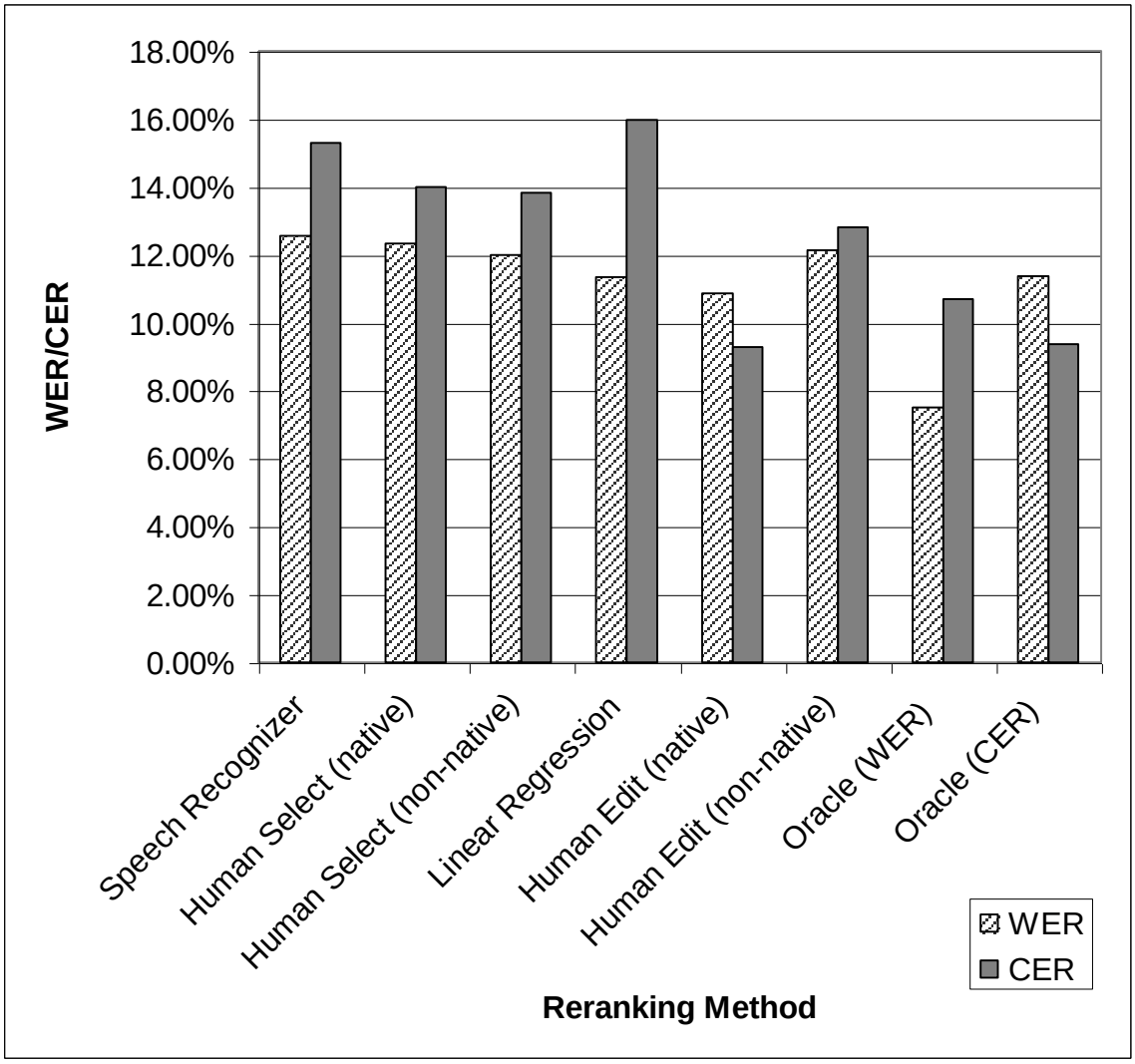

Figure 7.3: Word error rates (WER) and concept error rates (CER) of different reranking methods

When we compared different reranking methods by concept error rates, we found that human subjects, both selecting and editing, did better than a linear regression model. The concept error rate of the regression model is even higher than the baseline speech recognizer concept error rate. The explanation might be that this regression model was

trained to minimize the word error rate not the concept error rate. The difference between native and non-native speakers ability is similar to the difference when we evaluated using word error rate. Native and non-native speaker performances on human selecting are about the same, but native speakers did better on the editing task with larger difference. When both native and non-native speakers were allowed arbitrary editing the hypotheses they could gain higher concept error rate reduction than just selected the most appropriate hypotheses from the lists. However, the gain achieved by non-native speakers is less than the gain in achieved by native speakers since some non-native speakers introduced new errors when they tried to edit the hypotheses. The concept error rate of the native speaker editing is even lower than the concept error rate of the oracle concept error rate model.

Approach	WER		CER	
	Reranked WER	Relative Improvement	Reranked CER	Relative Improvement
Speech recognizer (baseline)	12.57%	-	15.29%	-
Human selecting (native)	12.33%	1.87%	14.00%	8.45%
Human selecting (non-native)	11.99%	4.58%	13.81%	9.68%
Linear regression	11.35%	9.67%	15.97%	-4.45%
Human editing (native)	10.87%	13.53%	9.28%	39.35%
Human editing (non-native)	12.12%	3.54%	12.80%	16.32%
Oracle (WER)	7.50%	40.34%	10.69%	30.10%
Oracle (CER)	11.38%	9.44%	9.37%	38.75%

Table 7.2: Relative improvement on word error rate and concept error rate of different reranking approaches

In [Ringger and Allen, 1996(2)], the post-processing method that arbitrary edited the hypotheses achieved 14.9% and 24.0% relative improvement on word error rate on two different corpora as shown in Table 2.1. The improvement is greater than the improvement from the native speakers editing in our experiment which demonstrates that the algorithm for arbitrary editing is efficient enough. However, we must note that the initial word error rates were higher in their corpora. Hence, there was more room for improvement.

7.4 The evaluation of individual features

To evaluate a performance of each feature on the N-best hypothesis reranking task, we reranked the 5-best hypothesis list according to the value of that feature and then selected the hypothesis with the highest feature value as an output of the reranking module. We chose the feature value representation so that the higher value corresponds to the better hypothesis. For example, in gap number feature, the more gaps in the hypothesis, the lower the feature value. We compared the performance of each feature to the word error rate and the concept error rate of the recognizer original output. This baseline is the same as the reranking result when use speech recognizer score to rerank the N-best list. In this section, we discuss the performances of different variations of the following features: N-best word rate, parse quality features, slot N-gram model, expected

slots and state-conditional slot N-gram model. At the end of this section, we summarize the performance of each feature on its best model.

7.4.1 N-best word rate

The performances of two variations of the N-best word rate feature, *average raw score* and *average confidence score*, measured by word error rate (WER) and concept error rate (CER) are given in Table 7.3.

Feature	WER (%)	CER (%)
N-best word rate (average raw score)	17.45	13.37
N-best word rate (average confidence score)	15.26	12.18

Table 7.3: The performances of two variations of the N-best word rate feature

From a significance test, the average confidence score variant of the N-best word rate feature is significantly better than the average raw score variant for both the average word error rate and the average concept error rate. Therefore, between these two variations, we selected the average confidence score as a method for calculating the N-best word rate feature.

When we used the average raw score to calculate the overall N-best word rate score, a short hypothesis tends to get a better score than a longer one. The reason for this is that, only the word that occurs in every hypothesis in an N-best list gets a full N-best word rate score. So, this feature variation selects a hypothesis that is (or close to) an intersection of every hypothesis, which is normally a shorter one. This can cause many deletion errors since the correct words may not appear in every hypothesis due to deletion errors in some hypotheses. On the other hand, for the average confidence score, a word that gets a full N-best word rate score is a word that occurs often enough in the N-best list (at least in 50% of the hypotheses in an N-best list). So, the average confidence score can avoid the problem of high deletion errors that occurs with the average raw score N-best word rate.

From the example in Table 7.4, the fourth hypothesis has the best raw score N-best word rate even it has two deletion errors because it is the hypothesis that close to the intersection of every hypothesis the most. For the confidence score N-best word rate, the first, second (the correct one) and fourth hypotheses have a full score since every word in theses hypotheses occurs frequently enough.

Hypothesis	N-best word rate (raw score)	N-best word rate (confidence score)
WOULD LIKE TO LEAVE ON SUNDAY	0.84	1.00
I WOULD LIKE TO LEAVE ON SUNDAY	0.81	1.00
I LIKE TO LEAVE ON MONDAY	0.77	0.83
WOULD LIKE LEAVE ON SUNDAY	0.89	1.00
I WOULD LIKE LEAVE ON MONDAY	0.80	0.83

Table 7.4: A sample N-best list and the raw score N-best word rate and the confidence score N-best word rate of each hypothesis

7.4.2 Parse quality features

We investigated four variations of three parser quality features. The variations are unparsed penalty with normalization and without normalization and under-covered reward with normalization and without normalization. The performances of all the variations are given Table 7.5.

Feature		With normalization		Without normalization	
		WER (%)	CER (%)	WER (%)	CER (%)
Unparsed penalty	Unparsed words	12.74	10.96	12.30	10.68
	Gap number	12.66	11.15	12.03	10.32
	Fragmentation transitions	12.67	11.01	12.21	10.40
Under-covered reward	Coverage	12.74	10.96	18.58	14.89
	Semantic slot number	12.66	11.15	19.39	20.26
	Fragmentation continuations	12.67	11.01	18.69	18.54

Table 7.5: The performances of parse quality feature variations

From the experiments, we found that the output of the reranking module that use normalized unparsed penalty features and the output of the reranking module that use normalized under-covered reward features are identical. Therefore, their performances measured by word error rate and concept error rate are the same. For non-normalized features, unparsed penalty features did better than under-covered reward features. The non-normalized unparsed penalty features also performed significantly better than the normalized variations when measured by word error rate. When measured by the concept error rate, they performed better but not significantly except for non-normalized gap number that did significantly better tan the normalized one. Among the parse quality features, non-normalized gap number performed the best on both word error rate and concept error rate.

Since a reranking module makes a comparison among the hypotheses of the same utterance, the lengths of the hypotheses are about the same. Therefore, the normalization may not be necessary. Sometimes an error in a hypothesis can be parsed as a partial parse. Therefore, a hypothesis with more elements covered in the parse is not necessary be a better hypothesis. On the other hand, it is less likely that the part of a hypothesis that is not covered in the parse is not an error. For that reason, unparsed penalty features can evaluate the quality of a hypothesis more accurately than under-covered reward features.

7.4.3 Slot N-gram model

In our experiments, we set N to 2 because 62.4% of the hypotheses in our training data contain only a single slot. Therefore, using just a bigram model is enough to capture the relationship between slots while also avoids the sparse data problem. The performances of different slot bigram models discussed in section 3.2.2 are given in Table 7.6.

In table 7.6, we also reported the perplexities of the slot bigram models. Since we used a slot bigram model in a hypothesis reranking task, a desired model is the one that can differentiate between the correct hypothesis and the one that contains errors.

Specifically, a desired model should assign higher probability to the correct hypothesis than to the incorrect one. Therefore, the model that has low perplexity on a set of correct hypotheses while having high perplexity on a set of incorrect hypotheses is considered a good model. We separated the hypotheses into 2 sets. The first set contains the best, lowest word error rate, hypothesis of each utterance. The rest of the hypotheses were put into the second set.

Model	Slot representation	Discounting strategy	Perplexity		WER (%)	CER (%)
			Best hypotheses	Not-best hypotheses		
1	Frame label and slot label	absolute	8.52	12.93	13.90	12.70
2	Slot label	absolute	8.39	12.77	13.87	12.70
3	Slot label and gap string	absolute	19.91	65.65	13.08	12.10
4	Slot label and gap string	linear	18.73	60.01	13.06	12.04
5	Slot label and gap string	good_turing	18.61	58.95	13.09	12.07
6	Slot label and gap string	witten_bell	18.08	54.61	13.06	12.05

Table 7.6: Perplexities and performances in term of word error rate and concept error rate of slot bigram models with different slot representations and discounting strategies

Comparing two slot representations, the one that uses both a frame label and a slot label (model1) and the one that use only a slot label (model2), the latter is better when measure by the word error rate of the reranking module output. In term of perplexity, the perplexities of model2 are lower than those of model1 on both sets of hypotheses. It is quite difficult to determine which model has higher discriminative power in distinguishing between a correct hypothesis and an incorrect one. However, since the performance of model2 is slightly better than model1 and from the discussion in section 3.2.2, we decided to represent a slot using only its slot label.

Model3 represents a semantic slot by a slot label and represents a gap slot by a gap string. The perplexities of model3 on both sets of hypotheses are higher than those of model2. However, the difference between the perplexities of both sets was increased significantly in model3. So, it is possible to say that model3 has higher discriminative power than model2. Moreover, the improvement in term of word error rate and concept error rate is also significant.

Model3 through model6 use the same slot representation. However, each model deploys different discounting strategy. Both the discriminative powers and the reranking performances of these models are not significantly different. Since model4, which uses linear discounting strategy, has the lowest word error rate and concept error rate, we decided to use this model to generate a slot bigram feature.

7.4.4 Expected slots

Table 7.6 presents the performances of different variations of expected slots feature. The binary representation of expected slots feature performed significantly better than the proportional representation. This is also the case when we also considered acceptable slots together with expected slots. This result implies that the number of expected slots in a hypothesis is less important than their existence. We found the following explanation when we analyzed the result. A user sometimes provides more

information than what the system expected. For instance, in *query_arrive_city* state, a user may voluntarily provide additional information about the date and time of arrival. The additional information increases the total number of slots in a hypothesis, but doesn't increase the number of expected slots. As a result, the proportion of expected slots in a hypothesis is decreased. Sometimes the errors are parsed as expected slots, e.g. the insertion of a word 'no'. This problem makes a hypothesis that contains such an error has a high proportional expected slots value.

Feature	Value representation	WER (%)	CER (%)
Expected slots	Binary	11.79	10.19
Expected & acceptable slots	Binary	11.73	10.09
Expected slots	Proportion	13.14	12.23
Expected & acceptable slots	Proportion	13.18	12.36

Table 7.7: The performances of expected slots feature variations

When both expected slots and acceptable slots are considered, the performances of the expected slots feature didn't change much. The performance of the binary representation was slightly improved while the performance of the proportional representation was slightly degraded. Acceptable slots are mostly short words and are easily confused with recognition errors. Therefore, the problem of the proportional representation that caused by recognition errors was increased. Since the variation of expected slots feature that uses a binary representation and considers both expected slots and acceptable slots is the one that had the lowest word error rate and concept error rate, we chose this variation as one of the reranking features.

7.4.5 State-conditional slot N-gram model

Similar to a slot N-gram model discussed in section 7.4.3, we set N to 2 in the state-conditional slot N-gram model. The performances of two conditioning techniques are shown in Table 7.8.

Conditioning technique	WER (%)	CER (%)
State context-cue model	12.89	11.61
State-specific model	12.88	11.49

Table 7.8: The performances of two conditioning techniques in state-conditional slot bigram models

The state-specific model is slightly better than the state context-cue model. However, it isn't significantly better by a significance test. In Table 7.9, we broke down the performances of four reranking features: speech recognizer score, slot bigram, state-conditional slot bigram using context-cue model and state-conditional slot bigram using state-specific model. The numbers in bold are the best number among the three models, excluding the speech recognizer score baseline. The underlined numbers indicate the numbers that are better than the baseline.

We found that both of the state-conditional slot bigram features are better than the unconditional one when the number of utterances in the state is large enough. The state-specific model is better at more states than the state context-cue model and also better on

the average performance. For the state-specific model, we also tried interpolating the specific bigram model of each state with the general bigram model (the one discussed in section 3.2.2). The interpolation usually helps when the state-specific model doesn't have enough training data. However, the result showed a little improvement in word error rate, but some degradation in concept error rate. There is no specific pattern on when the interpolated model did better or did worse than the non-interpolated one. Therefore, we chose the state-specific model without the interpolation as a modeling technique for a state-conditional slot bigram feature.

State name	#Utts.	WER (%)				CER (%)			
		Speech recog-nizer	**Slot bigram**	**Context cue**	**State specific**	**Speech recog-nizer**	**Slot bigram**	**Context cue**	**State specific**
hotel_hotel_info	745	12.87	**13.94**	14.02	14.02	7.46	9.30	**9.19**	10.05
hotel_need_car	596	7.05	9.77	9.84	**9.21**	3.87	7.89	8.61	**7.46**
hotel_need_hotel	721	6.80	8.65	8.61	**8.27**	3.86	**5.94**	**5.94**	**5.94**
hotel_where	523	11.68	**13.13**	**13.13**	13.37	8.15	**9.59**	9.73	**9.59**
inform_epilogue	100	16.95	**20.58**	**20.58**	22.52	13.48	**16.31**	**16.31**	19.86
query_arrive_city	2,523	10.05	11.07	10.84	**10.70**	8.60	10.47	10.23	**9.97**
query_confirm	641	7.52	**9.74**	9.86	10.10	2.79	**5.17**	5.31	5.70
query_confirm_flight	3,729	9.73	12.50	**12.36**	12.53	7.72	12.08	**11.64**	11.69
query_depart_date	3,320	11.13	12.35	12.33	**12.27**	9.09	10.58	10.52	**10.27**
query_go_on	237	7.43	9.90	10.40	**9.03**	6.48	9.07	9.33	**6.99**
query_name	2,000	34.67	34.28	**33.89**	34.00	35.07	36.23	35.31	**34.73**
query_pay_by_card	312	10.49	**10.35**	10.49	11.04	3.37	**4.91**	5.21	**4.29**
query_preferred_airport	535	11.19	13.39	13.02	**12.41**	8.70	13.77	12.47	**12.08**
query_return	1,152	7.54	9.39	**8.95**	**8.95**	5.62	7.22	**6.70**	6.83
query_summary	203	7.79	**7.54**	**7.54**	**7.54**	3.64	4.09	4.09	**3.64**
Remaining	23	4.82	**6.02**	8.43	**6.02**	0.00	**2.86**	**2.86**	**2.86**
unhappy_excuse_me	129	5.98	**9.78**	9.96	10.14	5.26	**12.63**	13.16	13.16
query_depart_time	1,570	9.12	10.69	**10.15**	10.28	7.59	11.81	**9.80**	10.08
Average		11.45	13.06	12.89	12.88	9.43	12.04	11.61	11.49

Table 7.9: The performances of speech recognizer score feature, slot bigram feature and state-conditional slot bigram features (with two conditioning techniques, state context-cue model and state-specific model) for each dialog state

We note that none of the slot bigram or conditional slot bigram features preformed better than the baseline on average. However, there are some states, such as *query_name*, *query_pay_by_card* and *query_summary*, that these features are slightly better either at word error rate or concept error rate than the baseline. The reason why none of the features performs better than the baseline will be discussed in the following section.

7.4.6 Performance of individual features in summary

Table 7.10 presents word error rate and concept error rate of all features discussed in Chapter 3. For the feature that more than one variation was investigated, the performance of the best variation was reported. The number for the training set is the average number of the 10-fold cross validation. For the test set, we trained the reranking

model on the entire training data then evaluated the model on the Jun-99 test set. The values in bold are the one that are better than the baseline which is the speech recognizer score (in italic). The rank associated with each evaluation metric indicates the relative performance when the performances of all features are ranked by that metric. For example, N-best homogeneity with speech recognizer score ranks second on the training set word error rate, which means that it is the second best feature when we evaluate all the features based on the training set word error rate. The list of features in the table was ordered by the average between word error rate rank and concept error rate rank of the training set.

Feature	Training Set (Average)				Test Set			
	WER (%)	**Rank**	**CER (%)**	**Rank**	**WER (%)**	**Rank**	**CER (%)**	**Rank**
N-best homogeneity (Speech Recognizer score)	11.47	2	**9.37**	1	**11.39**	4	**9.68**	1
Speech recognizer score (baseline)	*11.45*	*1*	*9.43*	*2*	*11.41*	*6*	*9.80*	*2*
Expected slots	11.73	3	10.09	3	11.61	7	9.99	3
Highest-in-coverage	11.90	4	10.14	4	**11.24**	1	10.22	4
Gap number	12.03	5	10.32	5	**11.31**	2	10.33	6
Fragmentation transitions	12.21	6	10.40	6	**11.39**	5	10.26	5
Unparsed words	12.30	7	10.68	7	**11.32**	3	10.49	7
Language model score	13.02	9	11.23	8	12.15	10	10.56	10
Conditional slot bigram	12.88	8	11.49	10	11.71	8	10.53	9
N-best homogeneity (Language Model score)	13.39	11	11.41	9	12.46	12	10.49	7
Slot bigram	13.06	10	12.04	11	12.13	9	11.37	11
Conditional slot	13.46	12	12.52	13	12.30	11	12.68	12
N-best word rate	15.26	13	12.18	12	15.57	13	12.75	13
N-best homogeneity (Acoustic Model score)	18.05	14	15.50	14	20.19	14	18.67	14
Acoustic model score	19.51	15	16.52	15	21.30	15	19.55	15

Table 7.10: The performance of each individual feature in term of word error rate and concept error rate on both the training set and the test set

On the training set, the only feature that performed better than the baseline was N-best homogeneity with speech recognizer score. Its performance was better than the baseline when we evaluated by concept error rate. However, the improvement was not statistically significant. On the test set, N-best homogeneity with speech recognizer score was also the only feature that performed better than the baseline in term of concept error rate. But in term of word error rate, there were five features that performed slightly better than the baseline, highest-in-coverage, gap number, unparsed word, fragmentation transitions and N-best homogeneity with speech recognizer score.

Since most of the features couldn't improve the word error rate and concept error rate beyond the baseline as we expected, we seek the explanation by analyzing the reranked result of each feature and comparing it to the original result of the recognizer. The discussion for each feature is given below.

Hypothesis score

The baseline speech recognizer score is a combination of acoustic model score and language model score. Therefore, using acoustic score or language model score alone couldn't do better than their combination. A language model score feature seemed to favor a short hypothesis. As a consequence, it decreased insertion errors while increasing deletion errors. A language model score feature also preferred a hypothesis with high-frequency words. An acoustic model score feature had the highest error rates. It caused many insertion errors, especially the insertion of short words. Even though, using acoustic model score or language model score alone couldn't improve the result of the reordering module beyond the baseline, we hoped that by combining them with other features, we would be able to improve the reranking result beyond the baseline, which is the combination of just two features. The results of the combined models will be discussed in section 7.5.

N-best list scores

As discussed in section 7.4.1, an N-best word rate feature favored a short hypothesis, thus it made more deletion errors than the baseline speech recognizer score. This trend was also true for all the N-best homogeneity features (N-best homogeneity with speech recognizer score, N-best homogeneity with acoustic model score and N-best homogeneity with language model score). N-best homogeneity scores caused higher deletion errors but less insertion errors than their counterpart original scores.

In an N-best word rate feature, words from every hypothesis are weighed equally; however, in fact, the hypotheses are not equally good. When we considered a hypothesis score as a weight in an N-best homogeneity feature, we could do better if the score well represents the quality of the hypothesis such as a speech recognizer score. N-best homogeneity features usually did better than their counterpart original score features. Therefore, when we reranked the hypotheses using N-best homogeneity with speech recognizer score, we could improve the performance beyond the baseline speech recognizer score. However, N-best homogeneity with language model score did a little poorer than the language model score itself. The reason might be that N-best homogeneity feature already had a high number of deletion errors. When N-best homogeneity feature used language model score (which also favored a short hypothesis) as a weight, the number of deletion errors increased more than the number of the insertion errors that are decreased.

Parse quality features

The performances of parse quality features are about the same. Even thought parse quality features performed better than the baseline on the test set word error rate, on average they didn't do better than the baseline. One reason might be that we still have gaps in the correct user parses since some user utterances or parts of the utterances are not covered by the grammar.

A highest-in-coverage feature performed better than other parse quality features. As mentioned before, some errors can also be parsed. Therefore the fractions of the hypothesis that are covered or not covered in the hypothesis parse may not accurately reflect the quality of the hypothesis if the hypothesis contains an error. In this case, the more conservative feature like a highest-in-coverage feature, which chooses only the hypothesis that is fully parsed, can perform better.

Expected slots

An expected slots feature makes a mistake when a user responses with an unexpected utterance such as making a correction on the previous utterance, changing the focus to another topic or providing more information than what the system expected at that point. An expected slots feature also suffers from parser errors because this feature is extracted from parser output.

Slot-based features

The performances of slot-based features: slot bigram, conditional slot and conditional slot bigram, are about the same. Since the slots are taken from the output parses, these features also suffer from parser errors similar to parse quality features.

One might think that a single slot hypothesis can cause a problem to a slot bigram feature since it has no information, the sequence of slots, that the slot bigram captures. However, when we broke down the error rate into two numbers, one for the utterances that contain only one slot and another one for the utterances that contain at least two slots, there was no different in the performance. A slot bigram feature chose the hypothesis that contains a more common slot sequence. However, a user doesn't always use the sequence that is more common.

A conditional slot bigram feature can capture the relationship between slots better than a slot bigram feature since it also takes into account the influence of dialog states on user slot sequences. Another source of difficulty for both features might be that both of them model the correlation in user utterances at the slot level. In some cases, the slot labels are related but not the actual words inside the slots.

For a conditional slot feature, we found the same problem as in an expected slots feature when a user responds with the slots that are not regular for that particular state. Furthermore, a conditional slot feature can't distinguish between '*yes/no*' responses or similar cases, in which all of the hypotheses are semantically correct. We need acoustic information to help distinguishing in those cases.

In general, each feature has ability to promote some of the best hypotheses up to the first rank, but not all of them. Each feature only monitors the quality of the hypotheses from one aspect, e.g. grammar conformation or dialog manager expectation. Therefore, it can solve the problem only if the problem is related to its expertise. For example, a parse quality feature can move the best hypothesis that has the best parse quality up to the first rank. However, if the best hypothesis doesn't have the best parse quality, but it is better on another aspect, such as better suits the system expectation, a parse quality feature will fail to promote that hypothesis up to the first rank. Therefore, it is better to combine features together, so that we can monitor the quality of the hypotheses from various aspects at the same time. Chapter 4 contains the detail about

feature combination for N-best hypotheses reranking and the result is presented in the next section.

We also compared the results of some features in the hypotheses reranking task to the results of the same features on the related tasks that also conducted the evaluation on the Communicator data. These related tasks are word level confidence annotation [Zhang and Rudnicky, 2001] and utterance level confidence annotation [Carpenter *et al*., 2001], which were discussed in Chapter 2. All the three evaluations ware based on the data collected from the Communicator system. However, the subset of data that was used might be varied. The comparison is given in Table 7.11.

Feature	**Hypotheses Reranking Word Error Rate (%)**	**Confidence Annotation Error Rate (%)**	
		Word Level	**Utterance Level**
Baseline[1]	*11.45*	*30.2*	*32.84*
N-best homogeneity (SR)	11.47	26.8	-
Expected slots	11.73	-	20.97
Gap number	12.03	-	23.01
Fragmentation transitions	12.21	-	32.73
Unparsed words	12.30	23.8	19.93
Slot bigram	13.06	-	23.14
N-best word rate	15.26	30.2	-

Table 7.11: The performances of the features on hypotheses reranking task, word level confidence annotation task and utterance level confidence annotation task

For the confidence annotation tasks, both word level and utterance level, all the features performed better than the baselines, or at least equally good. However, for the hypotheses reranking task, these features couldn't do better than the baseline. We note that the confidence annotation tasks have higher baseline error rates, which means that there is more room for improvement. It might also be the case that the hypotheses reranking task is more difficult than the confidence annotation task because in the reranking task, we have to actually find the correct hypothesis, not only annotate that which one is right or wrong. We also note that the relative performance of each feature is different, for example, fragmentation transition performed better than slot bigram in the hypotheses rerankning task, but it is opposite in the utterance level confidence task.

7.5 The evaluation of feature combination

In this section, we present the results of linear regression models that were trained using the techniques described in Chapter 4. Each linear regression model was trained to predict the number of errors (or error rate) in each hypothesis. The hypothesis with the least number of predicted errors was chosen. All linear regression models used the hypothesis count as a weight for the number of times each hypothesis presented in the training set. The weight was applied to prediction errors, which the linear regression

[1] The baseline for the hypotheses reraking task is the speech recognizer word error rate. For the confidence annotation tasks, the baseline is the annotation error rate when we assume that every utterance is *correct* or *o.k.*

model tried to minimize when it estimated feature weights. The baselines for a feature combination model are word error rate and concept error rate of the speech recognizer output.

Technically, we can train a linear regression model to predict either word error rate or concept error rate. However, the result of a regression model that used concept error rate as a response was not so good. The reasons that a linear regression model performed poorly when predicted the concept error rate might be that the distribution of concept error rates are more skewed than the distribution of word error rates. The number of concepts per utterance is quite small. So, there are just a few possible values for the concept error rate which make it difficult to train an efficient model for predicting it. Furthermore, the features that we used as predictors seem to correlate more to the hypothesized words rather than the concepts. Therefore, we trained a linear regression model to predict only the word error rate. However, we still report the performance of the model using both metrics. The result of the experiments on feature representation and feature combination optimization are presented and discussed below.

7.5.1 Feature representation

We evaluated the effect of 3 different feature representations: raw score, linear scaling and linear scaling with clipping, in linear regression models. For both linear scaling techniques, we rescaled all the features that their original value ranges are not [0,1] (every features except N-best word rate, expected slots and highest-in-coverage). We also rescaled the responses word error rate. The linear regression models were trained on the rescaled features to predict the rescaled response. All 15 features in Chapter 3 (speech recognizer score, acoustic model score, language model score, N-best word rate, N-best homogeneity with speech recognizer score, N-best homogeneity with acoustic model score, N-best homogeneity with language model score, unparsed words, gap number, fragmentation transitions, slot bigram, conditional slot, expected slots, conditional slot bigram and highest-in-coverage) were used as predictors. The results in term of word error rate and concept error rate are given in Table 7.12. The estimated feature weighs for each feature representation are shown in Table 7.13.

Feature representation	WER (%)	CER (%)
Raw scores	14.81	11.59
Linear scaling	11.31	9.33
Linear scaling with clipping	11.29	9.33

Table 7.12: The performances of linear combination models with different feature representations

The model that trained on raw scores suffered from the difference in the order of magnitude of feature values. The values of most of speech recognizer features except N-best word rate are large in magnitude. Therefore, they got much lower weights than other features as shown in Table 7.13. The word error rate and concept error rate of the reranked result are much higher than the baseline, which is 11.45% and 9.43% respectively. When we rescaled the data into [0, 1] range with a linear rescaling technique, the estimated weight of each feature is not much different from each other. We also achieved significant improvement over the raw score representation on both word

error rate and concept error rate. The linear scaling with clipping that removes outliers and spreads out the distribution gained a little more improvement over the standard scaling method. Therefore, the linear scaling with clipping technique was chosen to pre-process feature values in all the linear regression models.

Feature	Feature representation		
	Raw Score	**Linear scaling**	**Linear scaling with clipping**
Speech recognizer score	-1.7390E-06	0.2779	0.1377
Acoustic model score	1.7500E-06	0.1818	0.2411
Language model score	- [1]	0.2959	0.3071
N-best word rate	-57.6200	0.0324	-0.0066
N-best homogeneity (SR)	-1.2510E-04	0.1101	0.1497
N-best homogeneity (AM)	1.2520E-04	-0.0318	-0.0322
N-best homogeneity (LM)	-8.4850E-09	0.0940	0.1343
Unparsed words	-10.5200	0.0854	0.0541
Gap number	-5.2720	0.0048	0.0461
Fragmentation transitions	15.6300	-0.0395	-0.0585
Slot bigram	4.0410	-0.0343	-0.0236
Conditional slot	1.2030	-0.0189	-0.0230
Expected slots	-10.3700	-0.0499	-0.0383
Conditional slot bigram	-5.3580	0.1011	0.1121
Highest-in-Coverage	-10.0600	-0.0845	-0.0382

Table 7.13: Feature weights estimated by linear regression models that used different feature representations[2]

However, the improvements on both linear scaling methods are not significant when compare to the speech recognizer baseline. From Table 7.13, we noticed that some feature weights are negative numbers, which are counter-intuitive. In linear scaling representation, the higher the number the better the feature is. In the response case, the higher the number the lower the estimated error rate is. Therefore, we expected all the features to get positive weights with a higher weight for the more important feature. The negative weights on some of the features and the marginal improvement over the baseline suggest that some of the features are redundant. Some features may capture overlapped information in the hypotheses. Thus, combining just a subset of the features may be more efficient. We discussed the detail on how to select the optimal combining model in section 4.3.

7.5.2 Optimal regression model

In this section, we use abbreviations given in Table 7.14 when we refer to feature names.

[1] A weight for language model score was not defined because of singularities.
[2] All the weights are excerpted from the last fold of 10-fold cross validation

Feature name	Abbreviation
Speech recognizer score	SR_score
Acoustic model score	AM_score
Language model score	LM_score
N-best word rate	Word_rate
N-best homogeneity (SR)	SR_homogeneity
N-best homogeneity (AM)	AM_homogeneity
N-best homogeneity (LM)	LM_homogeneity
Unparsed words	Unparsed
Gap number	Gap
Fragmentation transitions	Fragment
Slot bigram	Bigram
Conditional slot	Cond_slot
Expected slots	Exp_slot
Conditional slot bigram	Cond_bigram
Highest-in-Coverage	Higest_Coverage

Table 7.14: The abbreviations of feature names

Both forward stepwise regression and backward stepwise regression were evaluated along with both variations of the goodness score, *AIC* and *BIC*. Table 7.15 shows the performance of each stepwise regression variation.

Goodness Score	Direction	WER (%)	CER (%)
AIC	Forward	11.28	9.33
AIC	Backward	11.28	9.33
BIC	Forward	11.28	9.30
BIC	Backward	11.29	9.32

Table 7.15: Performances of linear regression models chosen by stepwise regression using different goodness scores and search directions

When AIC was used, both a forward search and a backward search selected the all-features model in every cross-validation fold. However, when BIC was used, less complicated regression models were selected. The selected models are slightly different among the cross-validation folds. With a forward search, the selected models combined between 12 to 14 features together while in a backward search, the selected models combined between 11 to 14 features. The word error rate and concept error rate in Table 7.16 are the average performance of those models.

AIC gives less penalty to complex models than BIC as we can see from equation 4.5. As a result, the stepwise regression that uses AIC chose a model that is more complex. Nevertheless, the performances of all stepwise regression variations are about the same, and also similar to the performance of the regression model that combines all

features together. Hence, stepwise regression wasn't successful in discovering the optimal model in the model space.

Table 7.16 shows the result of various methods that were used to find the optimal features set for a linear regression model, as described in section 4.3.

Optimal model	Training set		Test set	
	WER (%)	CER (%)	WER (%)	CER (%)
Baseline	11.45	9.43	11.41	9.80
Oracle (WER)	8.23	7.7	8.15	7.76
Oracle (CER)	10.68	6.51	10.56	6.07
AM_score + LM_score	11.35	9.56	10.78	9.41
Combine all 15 features	11.28	9.33	10.94	9.64
Stepwise regression	11.28	9.30	10.92	9.68
Greedy search	11.14	9.19	10.89	9.68
Brute force Search	11.15	9.22	10.86	9.64

Table 7.16: Performances of different optimization strategies on feature selection

The baseline model is the result from a speech recognizer. The oracle models are the models that select the hypothesis that has the lowest word error rate or concept error rate. The performance of the oracle model that optimizes the word error rate is the lower bound, the best possible number that we can achieve, of the reranked word error rate. Similarly, the performance of the oracle model that optimizes the concept error rate is the lower bound of the reranked concept error rate.

The speech recognizer score, which is the score the speech recognizer uses as its reranking criterion, is the sum of acoustic model score and language model score. When we combined these two scores with the weights estimated by a linear regression model, we achieved a small improvement on the word error rate of the training set. More improvement was gained over the test set.

For the stepwise regression model, the performance shown in Table 7.16 is the performance of the stepwise regression in the forward direction that used BIC as a goodness score. This stepwise regression model is a little better than other variations of the stepwise regression models as shown in Table 7.15. When the stepwise regression was trained on the entire training data it chose 13 features out of 15 features, excluding only N-best word rate and conditional slot. The order that each feature was added to the model and the word error rates and concept error rates of the intermediate models are shown in Figure 7.4. The model that trained on the entire training data was the model that we used to rerank the test set.

Both word error rate and concept error rate did not always decrease when a new feature was added to the model since the stepwise regression uses penalized likelihood as a search criterion not the word error rate or the concept error rate, as described in section 4.3.1. The intermediate linear regression model seemed to reach its local optimal point for the error rates around the 8^{th} and 9^{th} iterations. After that, adding more feature slightly increased word error rate and concept error rate.

Both a greedy search and a brute force search used the average word error rate of the n-fold cross-validation to guide the search. For a brute force search, we only used 4-fold cross validation instead of 10-fold cross-validation, which we used in other

experiments, to reduce the computational complexity. We also applied a pruning technique in a brute force search. For a considered model in the search space, if the word error rate of any fold of the cross-validation is worse than the baseline word error rate of that fold, the model is immediately discarded. It may seem odd that the performance of the model selected by a brute force search is a little worse than the performance of the model selected by a greedy search. The reason for this is that the search spaces are slightly different since a greedy search was performed on the result of 10-fold cross-validation while a brute force search was performed on the result of 4-fold cross-validation. However, we found that the optimal model selected by a greedy search is the second best model in a brute force search.

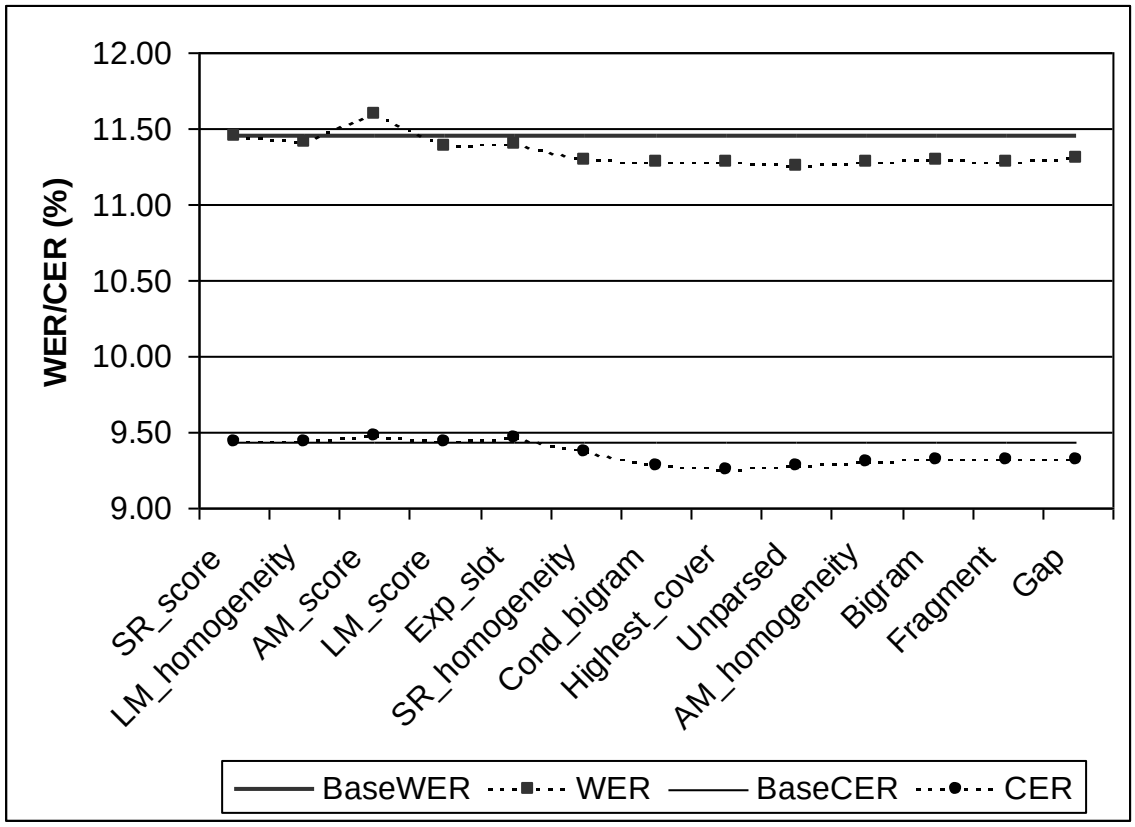

Figure 7.4: Word error rate and concept error rate of each stepwise regression iteration

Word error rate and concept error rate of each greedy search iteration is shown in Figure 7.5. The optimal model selected by a greedy search used 6 features: speech recognizer score, acoustic model score, language model score, N-best word rate, N-best homogeneity with speech recognizer score, and slot bigram. The word error rate was monotonically decreasing due to the search criterion. However, the concept error rate, which was not the search criterion, also decreased monotonically.

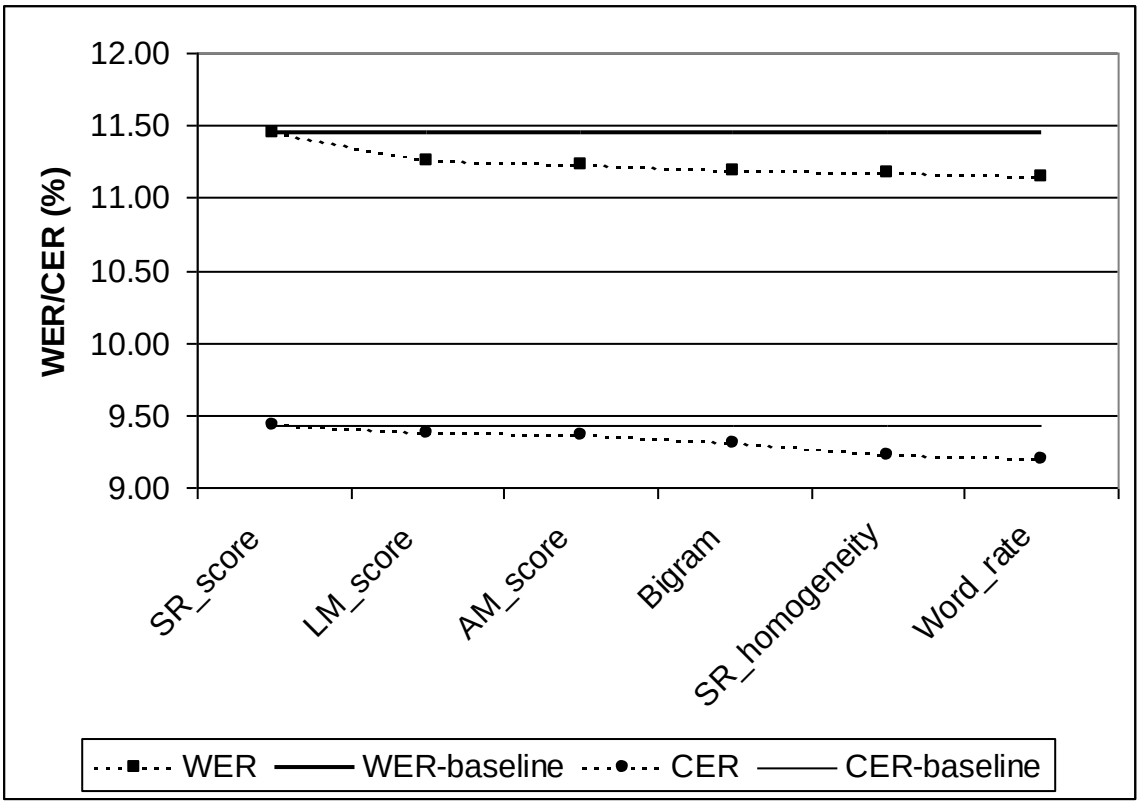

Figure 7.5: Word error rate and concept error rate of each greedy search iteration

The optimal model obtained by the brute force search used 7 features: speech recognizer score, acoustic model score, language model score, N-best word rate, N-best homogeneity with speech recognizer score, unparsed words and slot bigram. This model is very similar to the one selected by a greedy search except that it contains and an extra feature, unparsed word. When compared the performance of the model selected by a greedy search to the performance of the model selected by a brute force search on the 4-fold cross validation, they are quite similar. The smaller model chosen by a greedy search has word error rate of 11.15% and concept error rate of 9.21%. A brute force search didn't achieve much improvement over a greedy search. Therefore, a greedy search is considered better since it requires much less computation than a brute force search. Moreover, word error rate reduction on the model chosen by a greedy search is statistically significant.

We observed that a set of features that was selected by any of the features selection methods (a stepwise regression, a greedy search and a brute force search) and the order that each feature was selected didn't correlate with the performance of the features individually. Some features for which the individual performance were quite poor, such as acoustic model score and language model score, were selected in the early iterations while some features that performed well individually such as expected slots, highest-in-coverage and gap ratio were not selected or were selected on later iterations.

From the result on the test set, the model that is optimal on the training set, the model selected by a greedy search, is not optimal on the test set. The result in Table 7.16 shows that the model that combined acoustic model score and language model score

together has a better performance on the test set. The likely reason for is that the model that is optimal on the training data is over-fitting to the training data. However, it still achieved improvement over the baseline on the test set and even more than the improvement achieved on the training set. We also note that, there is not much improvement on concept error rate on both training set and test set. This might due to the fact that all the linear regression models were not trained to directly optimize the concept error rate.

7.6 The evaluation of utterance selection

In this section, we present the result of the hypotheses reranking process that reranked only suspicious utterances selected by an utterance selection module. Two techniques were used to select suspicious utterances to be reranked, single feature classifier and classification tree. Utterance selection techniques were described in Chapter 5.

The utterance classifiers were trained to classify each utterance as an 'o.k.' utterance or a 'suspicious' utterance. The results presented in this section were also obtained from the 10-fold cross-validation. However, when the data were divided into a training set and a test set, only a subset of the training data that was classified as 'suspicious' was used to train a linear regression model. Similarly, only a subset of the test data that was classified as 'suspicious' was reranked.

We evaluated the reranking process that incorporates an utterance selection module in two aspects, selection performance and reranking performance. The selection performance measures how good the selection module is at a classification task. Three metrics: precision, recall and F-1, described in section 5.1 were used. The reranking performance was reported in term of word error rate and concept error rate as usual. We would like to note that the word error rate and concept error rate reported in this section were combined error rates, namely, the error rates of reranked hypotheses were used when utterances were passed to a reranking module while the error rates of the original first rank hypotheses were used when utterances were classified as 'o.k.'.

7.6.1 Single feature classifier

Table 7.17 presents the performances of single feature classifiers in term of selection performances and reranking performances. The results of single feature classifiers that used first rank score criteria and difference score criteria (the difference between the first rank hypothesis feature value and the second rank hypothesis feature value) were compared to the results of the speech recognizer baseline, the utterance selection baseline and the oracle selection. Table 7.17 shows the results on the training set while table 7.18 shows result on the test set. For single feature classifiers, only the selection criteria that yielded the best performance in one of the evaluation metrics are presented and the best value is highlighted in bold. For example, "language model score < 0.4" is the criterion that yielded the highest precision with the value of 0.50 for a single feature classifier that used the first rank feature value as a classification feature. The underlined number indicates reranking performance that is better than the selection baseline.

The selection baseline is the model that classifies all utterances as 'suspicious'; therefore, all of them are reranked. Since every utterance will be reranked in the selection

baseline, its reranking performance is the same as the performance of the reranking model that that it uses which is the reranking model that was optimized by a greedy search (as described in the previous section). In the oracle selection, we rerank the hypotheses only when the first rank hypothesis is not the most correct one. As a result, we could improve the word error rate significantly over the selection baseline. When compared the oracle selection to a speech recognition baseline, both word error rate and concept error rate were reduced significantly. The oracle (mismatch) model had a mismatch condition in the reranking model that was used on the training set and the test set, namely, the reranking model was trained on the entire training set but was used to rerank only the selected set of the test data. We suffered some degradation in the reranking performance from the mismatch. This result confirms that it is better to train the reranking model on the selected set.

Selection criterion	**Reg. model**	**% selected**	**Selection performance**			**Reranking performance**	
			Precision	**Recall**	**F-1**	**WER (%)**	**CER (%)**
Speech recognition baseline	-	-	-	-	-	11.45	9.43
Selection baseline	1[1]	100.00	0.09	1.00	0.17	11.14	9.19
Oracle	1	9.02	1.00	1.00	1.00	10.25	8.85
Oracle (mismatch)	1	9.02	1.00	1.00	1.00	10.86	9.05
Oracle	2[2]	9.02	1.00	1.00	1.00	10.20	8.81
Single feature classifier (first rank score)							
Language model score < 0.4	1	3.11	**0.50**	0.17	0.26	11.28	9.43
N-best homogeneity (LM) < 1	1	93.46	0.10	**1.00**	0.18	11.15	9.20
Language model score < 0.7	1	12.82	0.39	0.55	**0.46**	11.16	9.28
Language model score < 0.6	1	8.37	0.45	0.42	0.43	**11.14**	9.29
Single feature classifier (difference score)							
Language model score < -0.6	1	0.45%	**0.55**	0.03	0.05	11.40	9.43
N-best homogeneity (LM) < 1	1	93.59%	0.10	**1.00**	0.18	**11.14**	9.19
N-best homogeneity (SR) < 1	1	93.59%	0.10	**1.00**	0.18	**11.14**	9.19
Language model score < 0	1	11.45%	0.34	0.43	**0.38**	11.27	9.29

Table 7.17: Selection performances and reranking performances of single feature classifiers on the training data

Since the selected training data is different from the entire training data, we can further optimize the reranking model by performing a greedy search on the regression model that trained on the selected training set. The new regression model (model2) used 8 features: language model score, N-best homogeneity with speech recognition score, slot bigram, N-best homogeneity with language model score, conditional slot bigram, fragmentation transition, highest-in-coverage and N-best word rate, instead of 6 features:

[1] The regression model1 is the one optimized by a greedy search on the entire training data (without selection).

[2] The regression model2 was optimized by a greedy search when the linear regression model was trained only on the data selected by the oracle criterion.

speech recognizer score, language model score, acoustic model score, slot bigram, N-best homogeneity with speech recognition score and N-best word rate, as obtained when optimizing on the entire training set (model1). The result of the oracle selection with the regression model2 shows some improvement on both types of error rates over the oracle selection with the regression model1.

From the lists of features used by model1 and model2, we observed that features in model1 were mostly speech recognizer features while features in model2 came from every information sources: a speech recognizer, a parser and a dialog manager. When the recognizer output is already correct, relying on the features that the recognizer uses to determine its output (speech recognizer score and its components: acoustic model score and language model score) won't hurt a reranking result. However, this conservative weight assignment, as occurred in model1 since most of the training utterances were already correct, couldn't improve much beyond the speech recognizer baseline. On the contrary, when most of the training utterances were incorrect as in model2, the reranking relied less on the hypothesis scores and utilized more variety of features. Therefore, it had more chance to improve the reranking result beyond the recognition baseline. The result of the oracle selection reveals that if we rerank only utterances that need to be reranked, we can achieve a significant improvement on the reranking performance and also reduce computation cost to less than 10% (only 9.02% of the utterances need to be reranked).

When we classified the utterances with a single feature classifier that used the first rank score a selection criterion, the best reranking performance that we achieved is comparable to the selection baseline by reranked just 8.73% of the utterances. However, choosing the selection criterion based on its reranking performance is easily to over-fit the training data. Therefore, it might be better to choose the selection criterion based on the selection performance instead. Among three selection performance metrics, we found that F-1 is the most appropriate metric. The criterion that achieved high precision selected just a few utterances to rerank. Therefore, we couldn't get much improvement on the reranking performance over the speech recognition baseline as we can see from the performance of the criterion that has the best precision. On the other hand, the criterion that achieved high recall usually chose to rerank almost every utterance, which made the reranking performance not much different from the selection baseline that reranked every utterance. The criterion that yielded the best F-1; however, selected to rerank only 12.82% of the utterances, but still achieved the same level of the reranking performance as the selection criterion that achieved the best reranking performance.

For a single feature classifier that used a difference score as a selection criterion, we found that the criterion that had the best, 100%, recall also yielded the lowest word error rate. This error rate is equal to the word error rate of the selection baseline. Besides the two criteria presented in Table 7.17, other 24 criteria also leaded to 100% recall with comparable word error rate and concept error rate. However, all of them chose to rerank more than 90% of the utterances. The criterion that yielded the highest F-1 had a lower value of F-1 than that of the first rank score criterion. As a result, the reranking performance is lower.

It is quite difficult to justify which criterion is better between the first rank score criterion and the difference score criterion. There are more difference score criteria that yielded good reranking performance. However, these criteria selected more than 90% of

the utterances. The first rank score criteria that yielded good reranking performance, even though there are a few of them, chose to rerank just about 10% of the utterances.

Unfortunately, none of the selection criteria could improve the reranking performance beyond the selection baseline. The criterion "first rank language model score < 0.6" is the one that yielded the same word error rate as the selection baseline while choosing only 8.37% of the utterances to rerank.

Selection criterion	Reg. model	% selected	Selection performance			Reranking performance	
			Precision	Recall	F-1	WER (%)	CER (%)
Speech recognition baseline	-	-	-	-	-	11.41	9.80
Selection baseline	1	100.00	0.09	1.00	0.17	10.89	9.68
Oracle	1	9.23	1.00	1.00	1.00	9.91	8.72
Oracle	2	9.23	1.00	1.00	1.00	9.76	8.68
Single feature classifier (first rank score)							
Language model score < 0.4 (best precision)	1	3.63	0.54	0.21	0.31	11.00	9.91
N-best homogeneity (LM) < 1 (best recall)	1	90.77	0.10	1.00	0.18	10.90	9.68
Language model score < 0.7 (best F-1)	1	12.03	0.47	0.61	0.53	10.89	9.64
Language model score < 0.6 (best word error rate)	1	8.22	0.53	0.47	0.50	10.77	9.53
Single feature classifier (difference score)							
Language model score < -0.6 (best precision)	1	0.71	0.33	0.03	0.05	11.27	9.76
N-best homogeneity (LM) < 1 (best recall and WER)	1	91.07	0.10	1.00	0.18	10.90	9.68
N-best homogeneity (SR) < 1 (best recall and WER)	1	91.07	0.10	1.00	0.18	10.90	9.68
Language model score < 0 (best F-1)	1	10.18	0.44	0.48	0.46	11.10	9.57

Table 7.18: Selection performances and reranking performances of single feature classifiers on the test data

The results on the test set have similar trend as the training set except that the difference score criterion "language model score < -0.6" has the best precision only on the training set, but not on the test set. Some selection criteria can improve the reranking performance slightly beyond the selection baseline on the test set, for example, the first rank score criterion "language model score < 0.6".

7.6.2 Classification tree

We used 10-fold cross-validation to train and then prune a classification tree. A classification tree that used the first rank scores as classification features is shown in Figure 7.6 while the classification tree that used the difference scores as classification

features is shown in Figure 7.7. The percentages of ‘o.k.’ and ‘suspicious’ utterances at each node are given along with the rule that used to split the node.

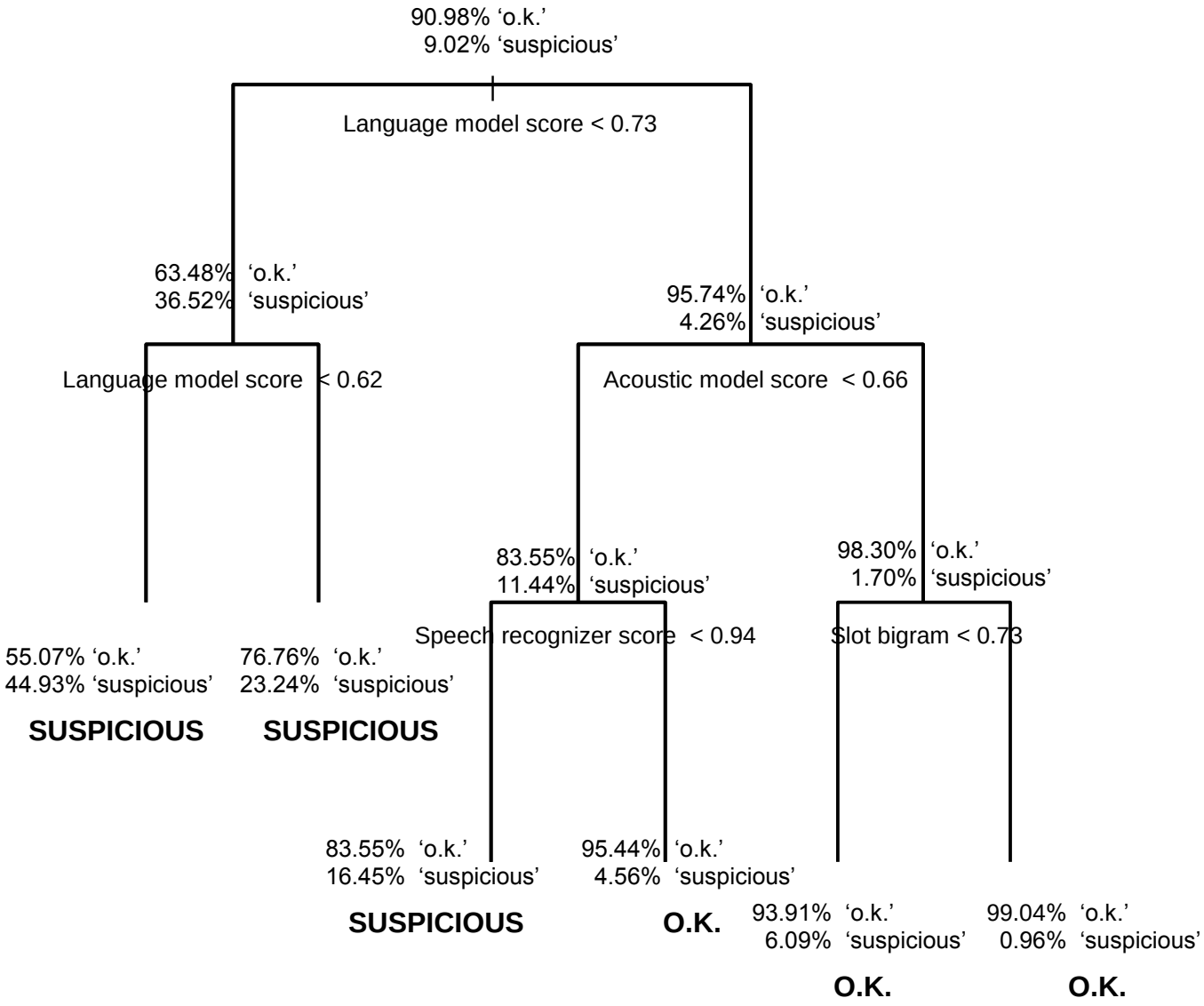

Figure 7.6: A classification tree which used the first rank score features

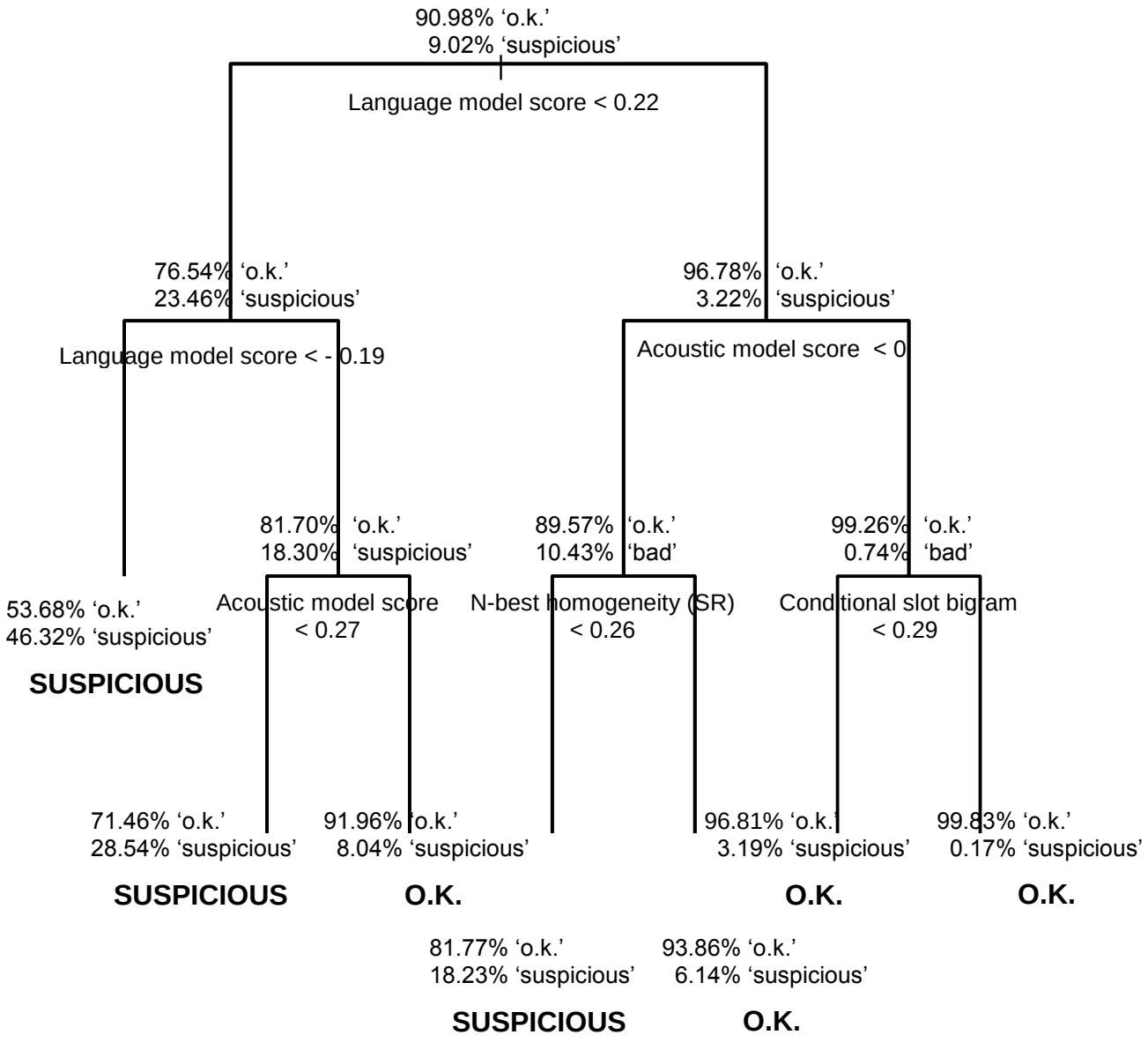

Figure 7.7 A classification tree which used the difference score features

From the classification trees, we noticed that the same classification feature was allowed at more than one node. For example, language model score was used as a node splitting criterion in two nodes for both classification trees. Since the number of good utterances is much higher than the number of suspicious utterances, the algorithm originally classified every leaf node as 'o.k.' which is not helpful for the reranking task. Therefore, we modified the way that a classification label is assigned to a leaf node as follows. If the percentage of suspicious utterances at a leaf node is higher than the percentage of suspicious utterances at a root node, that leaf node is classified as 'suspicious'; otherwise, it is classified as 'o.k.'. The modified classification labels are shown instead of the original labels in Figure 7.6 and Figure 7.7 for all the leaf nodes. With the modification of leaf node classification, the modified classification tree is no longer minimized the misclassification rate, since the modified classification makes the misclassification rate increased.

Table 7.19 shows the results of classification trees in term of selection performance and reranking performance. The underlined number indicates reranking performance that is better than the selection baseline.

Feature	**Reg. model**	**% selected**	**Selection performance**			**Reranking performance**	
			Precision	**Recall**	**F-1**	**WER (%)**	**CER (%)**
Training set							
Speech recognition baseline	-	-	-	-	-	11.45	9.43
Selection baseline	1	100.00	0.09	1.00	0.17	11.14	9.19
First rank scores	1	27.73	0.27	0.83	0.41	11.16	9.25
Difference scores	1	23.46	0.30	0.77	0.43	11.18	9.25
Test set							
Speech recognition baseline	-	-	-	-	-	11.41	9.80
Selection baseline	1	100.00	0.09	1.00	0.17	10.89	9.68
First rank scores	1	27.28	0.30	0.90	0.45	<u>10.80</u>	<u>9.60</u>
Difference scores	1	22.57	0.33	0.81	0.47	<u>10.75</u>	9.72

Table 7.19: Selection performances and reranking performances of the classification trees

The classification tree that trained on difference scores classified less number of utterances as 'suspicious' than the tree that trained on first rank scores. On the training set, the first rank score tree performed better in term of reranking performance while the difference score tree performed better in term of selection performance. However, on the test set, the classification tree that trained on difference scores is better in term of selection performance and word error rate, but not concept error rate.

Both of the trees can improve the word error rate slightly beyond the selection baseline on the test set. The tree that trained on the first rank scores can also improve the concept error rate beyond the selection baseline. Nevertheless, from the values of performances metrics, it is difficult to determine that which feature, first rank score or difference score, is better since the difference is insignificant. When compared the

performance of the trees to the performance of single feature classifiers, there is no significant different in both selection performance and reranking performance.

We found that it is quite difficult to find the criterion that can distinguish suspicious utterances from good utterances. This might due to the fact that the number of good utterances is much greater than the number of suspicious utterances (90.98% of the utterances, the first rank hypothesis is already the most correct hypothesis). If we just simply classify every utterance as ‘o.k.’, we will achieve the classification accuracy of 90.98%. It is quite difficult to find the classifier that is powerful enough to achieve the classification accuracy beyond this baseline. Moreover, the optimal goal of an utterance selection module is the reranking performance. We found that a criterion that yield high classification accuracy doesn’t always leads to high reranking performance. We found that the selection performance metric F-1 is a good candidate metric, however, it is not strongly correlated to the reranking performance. It is not clear on how to optimize the selection criteria directly on the reranking performance without sacrificing the large amount of computational cost.

Chapter 8
Conclusion

This thesis investigates N-best hypotheses reranking techniques for improving speech recognition accuracy. We focused on improving the accuracy of a speech recognizer used in a dialog system. We chose a post-processing technique that reorders an N-best hypothesis list after the recognition process is complete as our reranking approach for the following reasons: 1) It retains the modularity between a recognition process and a reranking process which makes it easier to try on various features and combination techniques without making much intervention on the recognizer. 2) It also allows us to use global hypothesis features such as a syntactic feature.

We investigated 15 different features based on the analysis of the domain and the knowledge sources reported to be used by human beings in the same reordering task. These features are sampled from 3 components of a dialog system: decoder, parser and dialog manager. We used 7 features from a decoder, 3 of them, which are *speech recognizer score*, *acoustic model score* and *language model*, associate directly with each hypothesis while the other 4, *N-best word rate*, *N-best homogeneity with speech recognizer score*, *N-best homogeneity with language model score*, and *N-best homogeneity with acoustic model score*, involve a set of hypotheses in an N-best list. We believe that speech recognizer features are still useful, even though they were already used during a recognition process, because they reflect the similarity between the hypothesis and the input signal and can help distinguish between competing hypotheses when syntactic and semantic information are not sufficient. Five features are from a semantic parser: *unparsed words*, *gap number*, *fragmentation transitions*, *highest-in-coverage* and *slot bigram*. The first 4 represent the quality of a user utterance in term of parse quality, but at different granularities, while the last one captures regularity in the slot sequence. A dialog manger provides 3 features: *conditional slot*, *expected slots* and *conditional slot bigram*. These features model the relationship between a user utterance and the previous system prompt, but from different perspective.

We chose to work on the 5-best hypothesis list since using a larger list not only increases the cost of computation but may also degrades the performance due to noise and misrecognized words in hypotheses toward the end of the list. Besides the conventional performance metric, the word error rate, we also purposed the *concept error rate* as an alternative metric. Concept error rate correlates with the performance of a dialog system better since the concept captures the piece of information that actually affects the system action. An experiment with human subjects also revealed that concept error rate is the metric that better conforms to the criteria used by humans when they evaluated hypotheses quality. Two definitions of the concepts were proposed, *frame-level* concept and *path-level concept* and the path-level concept, which defines the concept in finer granularity, was chosen.

From the evaluation on the Communicator data, none of the individual features could improve both word error rate and concept error rate significantly beyond the speech recognizer baseline. The explanation might be that each feature monitors the quality of the hypotheses from only one aspect, e.g. grammar conformation or dialog manager expectation. Therefore, we decided to combine the features together using a linear

regression model. A linear regression model was trained to predict the word error rate of each hypothesis from hypothesis features, and then outputs the one that has the lowest (recomputed) word error rate. There was an attempt to trained to model that could predict the concept error rate. However, it couldn't achieve a satisfied result due to the skewed distribution of the concept and the fact that the features are more correlated to the hypothesized words rather than the concepts. We also used a *linear rescaling with clipping* technique to normalize feature values to deal with differences in order of magnitude.

A searching strategy was used to discover the optimal feature set for reordering; three search algorithms were examined: *stepwise regression*, *greedy search* and *brute force search*. The stepwise regression performs a greedy search based on the regression statistic, *penalized likelihood*, while the greedy search performs a search that based directly on the reranking performance metric, word error rate. The brute force search on the other hand, explores the entire search space to find the optimal model. The reranking model, that performed the best, combined 6 features together to predict error rate. These 6 features are *speech recognizer score*, *language model score*, *acoustic model score*, *slot bigram*, *N-best homogeneity with speech recognizer score* and *N-best word rate*. This optimal set of features was obtained using greedy search. The brute force search was also able to obtain the optimal set of features, but it requires much more computation. This model can improve the word error rate significantly beyond the speech recognizer baseline. The reranked word error rate is 11.14%, which is a 2.71% relative improvement from the baseline. The reranked concept error rate is 9.68%, which is a 1.22% relative improvement from the baseline. We realized that we couldn't improve much on the concept error rate, which considered more correlated to the dialog system performance, since the linear regression model didn't directly optimize the concept error rate.

To improve reranking accuracy and reduce computation we examined techniques for selecting utterances likely to benefit from reranking then applying reranking only to utterances so identified. Two classification techniques, *single feature classifier* and *classification tree*, were used to classify the utterances using two types of classification features: *first rank score* and *difference score*. A single feature classifier classifies the utterances using only a single feature with a manually defined threshold as a criterion while the classification tree combined more than one feature and use data-driven thresholds. With the oracle selection, we could improve the word error rate beyond the selection baseline, which reranked all the utterances. We also found that, the selection criteria in the training set and the test set must be matched in order to achieve the best performance. Furthermore, with the greedy search optimization on the selected set, we could gain slightly more error rates reduction. We evaluated the selection performance using three metrics: *precision*, *recall* and *F-1* and found that F-1 was the most appropriate evaluation metric. Adding an utterance selection module into a reranking process did not improve the reranking performance beyond the number achieved by reranking every utterance. However, some selection criteria achieved the same overall error rate by reranking just a small number, 8.37%, of the utterances

When comparing the performance of the proposed reranking technique to the performance of a human on the same reranking task, the proposed method did as well as a native speaker, suggesting that an automatic reordering process is quite competitive.

The accuracy improvements reported in this work fall short of those reported for similar work done using the ATIS corpus ([Rayner *et al*., 1994] and [Moore *et al*., 1995]). However, we note that the portion of the corpus used for those experiments appear to include utterances that are on the average significantly longer than those in our corpus (3.5 words). Short utterances may have insufficient structure to effectively engage the parse-level features in our model. Some reordering features, such as highest-in-coverage, which gained 6.8% relative improvement on word error rate in [Rayner *et al*., 1994], gained only 1.5% relative improvement on our test data.

The current hypotheses reranking model still has room to improve.

- New features that have high discriminative power in discriminating between good and bad hypotheses such as confidence scores in both the word level [Zhang and Rudnicky, 2001] and utterance level [Carpenter *et al*., 2001] may worth considering. It would also be beneficial to find additional features that capture the hypothesis quality at the concept level, so that it would be more possible to improve the concept error rate beyond the baseline.
- Other feature combining methods such as other types of regression model and neural network can also be used to combine the features together and predict the error rate or the quality of the hypotheses.
- Similarly, more powerful classifiers can help discovering the selection criterion that is near the oracle, which will make the utterance selection module, not only reduces the computational cost but also improves the reranking performance.
- The utterance selection module can also be improved by adding new features that aim particularly for an utterance selection task. One potential classification feature might be the confidence annotation at the utterance level.
- The results in Table 7.9 shows that by using only an individual feature, we can improve the error rates beyond the baseline for some dialog states. This suggests that, utterances in different dialog states have different characteristics. Therefore, having a separate regression model for each dialog state may improve the reranking performance.
- It would also be interesting to evaluate the reranking model on different test sets. We will have more room to improve on the test set that has higher error rate. However, we would like to note that the different characteristics between the training data and the test data might be a problem.

References

[Bansal and Ravishankar, 1998] Bansal, D. and Ravishankar, M., "New Features for Confidence Annotation", In *Proceedings of ICSLP*'98, Sydney, Australia, 1998.

[Boros *et al*., 1996] Boros, M., Eckert, W., Gallwitz, F., Hanrieder, G., Gorz, G. and Niemann, H., "Towards understanding spontaneous speech: Word accuracy vs. concept accuracy", In *Proceedings of ICSLP'96*, Philadelphia, PA, 1996.

[Brill *et al*., 1998] Brill, E., Florian, R., Henderson, J. C. and Mangu, L., "Beyond N-Grams: Can Linguistic Sophistication Improve Language Modeling?", In *Proceedings of COLING-ACL'98*, Montreal, Canada, 1998.

[Carpenter *et al*., 2001] Carpenter, P., Jin, C., Wilson, D., Zhang, R., Bohus, D. and Rudnicky, A. I., "Is This Conversation on Track?", In *Proceedings of EUROSPEECH'01*, Aalborg, Denmark, 2001.

[Chotimongkol and Rudnicky, 2001] Chotimongkol, A. and Rudnicky, A. I., "N-best Speech Hypotheses Reordering Using Linear Regression", In *Proceedings of EUROSPPECH'01*, Aalborg, Denmark, 2001.

[Clarkson and Rosenfeld, 1997] Clarkson, P. R. and Rosenfeld, R., "Statistical Language Modeling Using the CMU-Cambridge Toolkit", In *Proceedings of EUROSPEECH'97*, Rhodes, Greece, 1997.

[DeGroot and Schervish, 1989] DeGroot, M. H. and Schervish, M. J., *Probability and Statistics*, Addison-Wesley, Reading, Mass., second edition, 1989.

[Hacioglu and Ward, 2001] Hacioglu, K. and Ward, W., "Dialog-Context Dependent Language Modeling Using N-Grams and Stochastic Context-Free Grammars", In *Proceedings of ICASSP'01*, Salt Lake City, May 2001.

[Huang *et al*., 1993] Huang, X., Alleva, F., Hon, H. W., Hwang, M. Y., Lee, K. F. and Rosenfeld, R., "The SPHINX-II Speech Recognition System: An Overview", *Computer, Speech and Language*, vol. 2, pp. 137-148, 1993.

[Mangu *et al*., 1999] Mangu, L., Brill, E. and Stolcke, A., "Finding Consensus Among Words: Lattice-Based Word Error Minimization", In *Proceedings of EUROSPEECH'99*, Budapest, Hungary, 1999.

[Mendelsohn 1993] Mendelsohn, L., "Preprocessing Data For Neural Networks", *Technical Analysis of Stocks & Commodities magazine*, Vol. 11, 1993, p 416-420.

[Mitchell 1997] Mitchell, T. M., *Machine Learning*, McGraw-Hill, USA, 1997.

[Moore *et al*., 1995] Moore, R., Appelt, D., Dowding, J., Gawron, J. M. and Moran, D., "Combining linguistic and statistical knowledge sources in natural-language processing for ATIS"., In *Proceedings ARPA Spoken Language Systems Technology Workshop*, Austin, Texas, 1995.

[Mou and Zue, 2000] Mou, X. and Zue, V., "The Use of Dynamic Reliability Scoring in Speech Recognition", In *Proceedings of ICSLP'00*, Beijing, China, 2000.

[Rayner *et al*., 1994] Rayner, M., Carter, D. M., Digalakis, V. and Price P., "Combining Knowledge Sources to Reorder N-best Speech Hypothesis Lists", In *Proceedings of the ARPA HLT Meeting*, Princeton, New Jersey 1994.

[Ringger and Allen, 1996(1)] Ringger E. K. and Allen, J. F., "Error Correction via a Post-Processor for Continuous Speech Recognition", In *Proceedings of ICASSP'96*, Atlanta, Georgia, 1996.

[Ringger and Allen, 1996(2)] Ringger E. K. and Allen, J. F., "A Fertility Channel Model for Post-Correction of Continuous Speech Recognition", In *Proceedings of ICSLP'96*, Philadelphia, Pennsylvania, 1996.

[Ringger and Allen, 1997] Ringger E. K. and Allen, J. F., "Robust Error Correction of Continuous Speech Recognition", In *Proceedings of the ESCA-NATO Workshop on Robust Speech Recognition for Unknown Communication Channels*, Pont-a-Mousson, France, 1997.

[Rudnicky *et al*., 1999] Rudnicky, A. I., Thayer, E., Constantinides, P., Tchou, C., Shern, R., Lenzo, K., Xu, W., Oh, A., "Creating natural dialogs in the Carnegie Mellon Communicator system", In *Proceedings of EUROSPEECH'99*, Budapest, Hungary, 1999.

[Rudnicky, 2000] Rudnicky, A. I., "CMU Communicator". In *DARPA Communicator PI Meeting*, Philadelphia, Pennsylvania, 2000.

[Stolcke *et al*., 1997] Stolcke, A., Konig, Y. and Weintraub, M., "Explicit Word Error Minimization in N-best List Rescoring", In *Proceedings of EUROSPEECH'97*, Rhodes, Greece, 1997.

[Venables and Ripley, 1998] Venables, W. N. and Ripley, B. D., Modern Applied Statistics with S-Plus, Springer-Verlag, New York, second edition, 1998.

[Ward, 1991] Ward, W., "Understanding Spontaneous Speech: The Phoenix System", In *Proceedings of ICASSP'91*, Toronto, Canada, 1991.

[Ward, 1995] Ward, W. and Issar, S., "The CMU ATIS System", In *Proceedings of ARPA Workshop on Spoken Language Technology*, Austin, Texas, 1995.

[Wasserman, 2002] Wasserman, L., "Hypothesis Testing and p-value", In *36-326/727 Probability and Statistics II lecture note*, http://www.stat.cmu.edu/~larry/=stat326.02/

[Zhang and Rudnicky, 2001] Zhang, R. and Rudnicky, A. I., "Word Level Confidence Annotation using Combinations of Features", In *Proceedings of EUROSPEECH'01*, Aalborg, Denmark, 2001.