

FEATURE SELECTION FOR ERROR DETECTION
AND RECOVERY IN SPOKEN DIALOGUE SYSTEMS

by

TIZIANA LIGORIO

A dissertation submitted to the Graduate Faculty in Computer Science in partial
fulfillment of the requirements for the degree of Doctor of Philosophy
The City University of New York

2011

© 2011

TIZIANA LIGORIO

All Rights Reserved

This manuscript has been read and accepted for the Graduate Faculty in
Computer Science in satisfaction of the dissertation requirement for the
degree of Doctor of Philosophy.

Professor Susan L. Epstein

Date

Chair of Examining Committee

Professor Theodore Brown

Date

Executive Officer

Professor Matt Huenerfauth

Professor Rebecca J. Passonneau

Professor Elizabeth Sklar

Professor Virginia Teller

Supervisory Committee

The City University of New York

Abstract

FEATURE SELECTION FOR ERROR DETECTION AND RECOVERY IN SPOKEN DIALOGUE SYSTEMS

by

Tiziana Ligorio

Adviser: Professor Susan L. Epstein

Spoken dialogue between people and machines is increasingly common, but not as flexible and complex as that between people. Spoken dialogue is susceptible to error – human speech is often loosely structured, people change their mind at mid-sentence, repeat themselves, pause, and produce non-speech sounds. A spoken dialogue system expected to handle complex and flexible dialogue like that between people must be robust to error, and employ strategies for error detection and recovery.

People recover from understanding error during spoken dialogue with little effort, and without disruption of the dialogue flow. To study how people would handle the same kind of errors that a spoken dialogue system must contend with, the work recounted here *embedded* a person within a spoken dialogue system with noisy speech recognition, that is, the person and the system processed information concurrently. The person was also *ablated*, that is, had input and output restricted

to that of a spoken dialogue system. Run-time system features were collected to train data-driven models of human error detection and recovery. Our results indicate that people have successful dialogue interactions even with very high levels of noise, and they rely heavily on context and questions to disambiguate noisy speech recognition.

Empirical results also demonstrate that data-driven models of human error detection and recovery strategies can be obtained from system features alone. These models can draw from a rich and varied collection of system features across all levels of spoken language processing, and careful selection from among them is essential for effective modeling. To this end, this research developed two novel feature selection algorithms, one general and one knowledge-guided. They effectively and efficiently identify features for a particular machine learning algorithm and data set, and, guided by domain knowledge, they support adequate learning of error detection and recovery models.

The resulting models of human error detection and recovery were then implemented within a spoken dialogue system. Empirical results indicate these strategies support dialogue management to address speech recognition noise, and improve overall task success. Our results demonstrate that successful dialogue with a spoken dialogue system is possible despite high levels of understanding error.

Acknowledgements

I would like to thank my adviser, Susan L. Epstein, for her invaluable help, guidance and support. Without her this work would have not been possible. I would like to thank the outside member of my examining committee, Rebecca J. Passonneau, for her essential and continued guidance throughout this research, as well as the other members of my examining committee, Matt Huenerfauth, Elizabeth Sklar and Virginia Teller, for their important and insightful comments.

I would also like to thank Liana Epstein, Dara Greenwood, Xingjian Li, and Smiljana Petrovic, who have also provided insightful suggestions and contributed in various ways.

Finally, my deepest gratitude goes to my husband, Marco, for supporting me at all times and in all ways, and to whom I dedicate this work.

Table of Contents

Chapter 1 Introduction	1
1.1 The problem.....	3
1.2 Goals, approach, and contributions	9
1.3 A baseline spoken dialogue system	13
Chapter 2 Related work.....	24
2.1 Wizard of Oz studies for spoken dialogue systems	26
2.2 Data-driven error detection and recovery in spoken dialogue systems	28
Chapter 3 Data collection: a wizard of Oz study	32
3.1 Preliminary title experiment	36
3.1.1 Experimental design	37
3.1.2 Evaluation of wizards and callers	40
3.1.3 Models of wizard behavior	43
3.2 Full dialogue experiment	47
3.2.1 Experimental design	47
3.2.2 The corpus: a wizard analysis	56
3.2.3 System features.....	63
Chapter 4 Feature selection for error detection and recovery in spoken dialogue systems	75
4.1 Training instances and data pre-processing	76
4.2 Related work in feature selection	85
4.2.1 Filter and wrapper approaches	87
4.2.2 Randomization and weighting	89
4.3 XFF: an experimental framework for feature selection.....	93
4.4 Randomized feature weighting (RFW)	98
4.4.1 Experimental design for RFW.....	101
4.4.2 Experimental results with RFW.....	104
4.4.3 Discussion of the results with RFW.....	118
4.5 Randomized Feature Weighting with knowledge (RFW+).....	121
4.5.1 Experiments and results with RFW+.....	124
4.5.2 Discussion of the results with RFW+.....	129
4.6 Summary	138
Chapter 5 Improving the baseline system.....	140
5.1 Wizard error detection and recovery models	140
5.2 Modifications to CheckItOut.....	146
5.3 Evaluation.....	153
5.3.1 Data collection with CheckItOut and CheckItOut+	153
5.3.2 Comparative results.....	153
5.3.3 A unified evaluation	165
Chapter 6 Conclusion.....	170
6.1 Summary of contributions and future work.....	171
6.1.1 Rapid tailored feature selection	172

6.1.2 Wizard-derived expertise improves task success, task correctness, and throughput	176
6.2 Concluding remarks	179
Appendix A Development	182
Appendix B CheckItOut’s task tree	183
Appendix C Prompts in the full dialogue experiment	184
Appendix D Datasets from the UCI repository	186
Appendix E Experimental results with RFW	187
Appendix F Experimental results with RFW+	196
Glossary	200
Bibliography	207

List of Figures

1.1	Levels of SLU processing and their features.....	4
1.2	Book titles and their noisy ASR output.....	5
1.3	Understanding error at different SLU levels.....	6
1.4	Recovery actions for different sources of error.....	7
1.5	A sample dialogue with CheckItOut.....	14
1.6	Data flow in CheckItOut.....	15
1.7	A portion of CheckItOut's task tree.....	21
3.1	An embedded wizard in CheckItOut.....	32
3.2	Caller GUI for the title experiment.....	37
3.3	Wizard GUI for the title experiment.....	38
3.4	Distribution of actions selected by each wizard in the title experiment.....	40
3.5	Distribution of actions elicited by each caller in the title experiment.....	42
3.6	Annotated screenshot for the login GUI in the full dialogue experiment.....	49
3.7	Annotated screenshot for the book-request GUI in the full dialogue experiment.....	50
3.8	Annotated dialogue excerpt from the full dialogue experiment.....	52
3.9	Excerpt from a dialogue with wizard WA that illustrates her strategy to lead the caller to change book request type.....	59
3.10	Excerpt from a dialogue with wizard WA that illustrates a successful book identification with very noisy ASR transcriptions.....	59
3.11	Levels of SLU processing and their features (reproduced from Figure 1.1)...	63
3.12	Number of features in each feature group.....	64
3.13	Number of features in each feature meta-group.....	65
3.14	Number of features in each feature sub-group.....	66
3.15	A training instance that describes an adjacency pair.....	66
3.16	A hierarchical illustration of feature meta-groups, groups, and sub-groups...	67
4.1	Levels of SLU processing and their features (reproduced from Figure 1.1)...	76
4.2	Three models of wizards' decision making.....	78
4.3	Training instances for three models of wizards' decision making.....	79
4.4	Examples of multiple ASR strings in a single adjacency pair.....	82
4.5	XFF, an experimental framework for feature selection.....	94
4.6	Pseudocode for XFF.....	95
4.7	Pseudocode for a general weighted wrapper in XFF.....	96
4.8	Pseudocode for RFW.....	100
4.9	Results for RFW with C.4.5 on the DIALOGUE ALL data set.....	105
4.10	Results for RFW with C.4.5 on the DIALOGUE CORRELATED data set.....	106
4.11	Results for RFW with C.4.5 on the DIALOGUE UNSEARCHED data set.....	107
4.12	Results for RFW with C.4.5 on the DIALOGUE SEARCHED data set.....	108

4.13	Results for RFW with C.4.5 on the CARDIOTOGRAPHY data set.....	109
4.14	Results for RFW with C.4.5 on the ROBOT NAVIGATION data set.....	110
4.15	Results for RFW with C.4.5 on the STEEL PLATES data set.....	111
4.16	Results for RFW with C.4.5 on the BREAST CANCER data set.....	112
4.17	Results for RFW with C.4.5 on the MUSK data set.....	113
4.18	Results for RFW with C.4.5 on the full COVERTYPE data set.....	114
4.19	Results for RFW with C.4.5 on the reduced COVERTYPE data set.....	115
4.20	Results for RFW with C.4.5 on the full ISOLET data set.....	116
4.21	Results for RFW with C.4.5 on the reduced ISOLET data set.....	117
4.22	Pseudocode for RFW+.....	123
4.23	Results for RFW and RFW+ with logistic regression on the DIALOGUE ALL data set.....	126
4.24	Results for RFW and RFW+ with logistic regression on the DIALOGUE UNSEARCHED data set.....	127
4.25	Results for RFW and RFW+ with logistic regression on the DIALOGUE SEARCHED data set.....	128
4.26	Improvement of RFW+ over the initial feature subset on the DIALOGUE ALL data set with meta-groups, groups, and sub-groups.....	133
4.27	Improvement of RFW+ over the initial feature subset on the DIALOGUE UNSEARCHED data set with meta-groups, groups, and sub-groups.....	134
4.28	Improvement of RFW+ over the initial feature subset on the DIALOGUE SEARCHED data set with meta-groups, groups, and sub-groups.....	135
4.29	Performance comparison of RFW, RFW+ and the baselines on the dialogue data sets.....	136
4.30	Execution time comparison of RFW, RFW+ and the baselines on the dialogue data sets.....	137
5.1	Three models of wizards' decision making (reproduced from Figure 4.2)..	141
5.2	Distribution of features across groups and sub-groups for three wizard models of error detection and recovery.....	145
5.3	The dialogue managers in CheckItOut and CheckItOut+.....	147
5.4	Pseudocode for the wizard module in CheckItOut+.....	148
5.5	Two sub-dialogues with CheckItOut+ that identify a book from a wizard module voice search offer.....	150
5.6	Two sub-dialogues with CheckItOut+ that identify a book after the wizard module elicited a change in book request type.....	151
5.7	Percent of calls in which CheckItOut and CheckItOut+ ordered zero to four books.....	154
5.8	Percent of calls in which CheckItOut and CheckItOut+ ordered zero to four correct books.....	154
5.9	Average number of correct books and ordered books per call with.....	155

5.10	Average number of incorrect books and caller-repair attempts per call.....	155
5.11	Percent of calls in which the wizard module in CheckItOut+ triggered 0 to its maximum of 26 times.....	156
5.12	A simplified execution flow of CheckItOut+'s dialogue manager up until a book is ordered.....	157
5.13	Average number of caller turns and system turns per call, and number of seconds per call.....	158
5.14	Average number of caller turns in calls with zero to four correct books.....	159
5.15	Average number of system turns in calls with zero to four correct books....	159
5.16	Total number of book requests per call, and their distribution per request type.....	160
5.17	Average number of book requests in calls with zero to four correct book...	161
5.18	Average number of book offers by author in call with zero to four correct books.....	162
5.19	Satisfaction metrics from answer to survey questions.....	164
6.1	A sub-dialogue with CheckItOut+ with an unsuccessful offer from the voice search results.....	178

List of Tables

1.1	Adaptations to Olympus contributed by this research.....	16
3.1	Wizard performance in the title experiment.....	41
3.2	Distribution of correct actions in the title experiment.....	42
3.3	Decision trees for individual wizard models.....	45
3.4	Wizard behavior and performance in the full dialogue experiment.....	57
3.5	Caller behavior and performance in the full dialogue experiment.....	61
3.6	Adjacency-pair history features.....	68
3.7	Adjacency pair features.....	69
3.8	Recognition history features.....	70
3.9	Recognition features.....	70
3.10	Parse history features.....	71
3.11	Parse features.....	71
3.12	Query history features.....	72
3.13	Query features.....	72
3.14	Comprehensive-confidence history features.....	73
3.15	Comprehensive confidence features.....	73
4.1	Wizards' confidence in the quality of the ASR and in database matches, with their corresponding actions.....	77
4.2	Distribution of database queries over adjacency pairs in the book-request portion of dialogues from the full dialogue experiment.....	81
4.3	Distribution of caller utterances over adjacency pairs in the book-request portion of dialogues from the full dialogue experiment.....	81
4.4	Selectors currently implemented in XFF.....	97
4.5	The weight-initialization and weight-update methods currently implemented in XFF for weighted and weighted-heuristic selection.....	98
4.6	Data sets for comparison of RFW to feature selection baselines.....	102
4.7	Results for RFW+ on DIALOGUE ALL with logistic regression.....	130
4.8	Results for RFW+ on DIALOGUE UNSEARCHED with logistic regression.....	131
4.9	Results for RFW+ on DIALOGUE SEARCHED with logistic regression.....	131
5.1	Performance of three logistic regression models of wizard error detection and recovery trained on features selected by RFW+.....	142
5.2	Features selected by RFW+ for three models of wizard error detection and recovery.....	143
5.3	Survey questions to elicit caller satisfaction measures.....	163
5.4	Task-success and dialogue-cost measures in dialogues with CheckItOut and CheckItOut+.....	166
5.5	Performance equations on cost and success measures.....	167
5.6	Performance equations on experience and emotion measures.....	168
E.1	Results for RFW with a C4.5 learner on data from the full dialogue experiment.....	188
E.2	Comparative results for RFW with a C4.5 learner on seven UCI data sets.....	189

E.3	Comparative results for RFW on the full COVERTYPE and ISOLET data sets with a C4.5 learner.....	190
E.4	Comparative results for RFW with logistic regression on data from the full dialogue experiment.....	191
E.5	Comparative results for RFW with logistic regression on seven UCI data sets.....	192
E.6	Comparative results for RFW with SVM on data from the full dialogue experiment.....	193
E.7	Comparative results for RFW with SVM on seven UCI data sets.....	194
E.8	Comparative results for RFW on the full COVERTYPE and ISOLET data sets with SVM.....	195
F.1	Comparative results for RFW and RFW+ with logistic regression on DIALOGUE ALL	197
F.2	Comparative results for RFW and RFW+ with logistic regression on DIALOGUE UNSEARCHED.....	198
F.3	Comparative results for RFW and RFW+ with logistic regression on DIALOGUE SEARCHED.....	199

Chapter 1

Introduction

Spoken dialogue between people and machines is increasingly common. A *spoken dialogue system* is intended to engage effectively in dialogue with people. The system may fully control the path of the dialogue (*system initiative*) or it may permit callers to interrupt and volunteer information (*mixed initiative*).

Ideally, people could speak with computers the way they do with one another. During spoken dialogue, however, human speech is loosely structured (Jurafsky and Martin 2008). It may include *disfluencies* (filled pauses, repetitions, mid-word interruptions or false starts), and the speaker may change her mind or her intention at mid-sentence. Moreover, background noise may degrade speech intelligibility for the machine. As a result, both word-level and concept-level understanding errors are endemic to human-computer dialogue, even for systems designed to elicit short utterances from users. Even a spoken dialogue system with high quality *speech recognition* (the system's transcription of the speech signal into text) must contend with understanding errors, and be able to detect and recover from them gracefully (Bohus 2007, Bulyko, et al. 2005, Carlson, et al. 2005, Skantze 2005, Skantze and Edlund 2004, Torres, et al. 2005).

There are many ways to recover from understanding errors (*recovery actions*). For example, a simple recovery action asks the speaker to repeat. Repetitive understanding errors lead speakers to hyperarticulate, which further decreases the

quality of speech recognition. After a sequence of consecutive understanding errors, it is more sensible, and less frustrating to the user, to ask a related question that could help the system understand what was said. The appropriateness of a recovery action depends on the kind of error, the user, the system application, and the current *context* of the dialogue (what has been said so far, and the dialogue's current goal/sub-goal). Moreover, understanding errors can occur in different system components, and different sources of error also affect the appropriateness of a recovery action.

Given the complexity of these dependencies, a data-driven approach to error detection and recovery is a sensible design solution. *Learning algorithms* that adapt their execution to empirical data can be used for this purpose. Here, system *features* (variables whose values are available to the system at runtime) describe knowledge about the error and its source with respect to system components. The choice of these features impacts the accuracy of the learning algorithm (Guyon and Elisseeff 2003, Guyon, et al. 2006). Ideally, the selected features would boost learning performance. In most cases, however, the number of possible features is large, many features are redundant, and not all of them are always relevant to every source of error. Automated feature selection can be used in these cases.

The thesis of this work is that feature selection is essential to data-driven error detection and recovery for robust and adaptive spoken dialogue systems. In particular, this research investigates how feature selection can exploit knowledge about the provenance and interaction of system features and their relevance to different sources of error. The principal contributions of this thesis are two novel algorithms for feature selection, one general and one augmented with knowledge

about spoken dialogue systems. Additional contributions of this thesis include a rich corpus of dialogue interactions to be released to the research community, an experimental feature selection framework, and the identification of sets of features for error detection and recovery at different stages of spoken language understanding.

Chapter 2 provides background on error detection and recovery. Chapter 3 describes a data collection experiment that supported data driven error detection and recovery for this research, and produced a rich corpus for release to the research community. Chapter 4 discusses a novel, knowledge-guided approach to feature selection for data-driven error detection and recovery, and reports on its results. It also provides details on a feature selection framework that supported experiments with different feature selection approaches. Chapter 5 describes how the learned error detection and recovery models were used to improve a baseline spoken dialogue system. Finally, Chapter 6 summarizes the contributions of this dissertation and discusses future work.

In the remainder of this chapter, Section 1.1 motivates this research, and Section 1.2 outlines its research goals, approach, and contributions. Section 1.3 introduces the spoken dialogue system and application used for this research. A glossary of terminology is provided immediately following the Appendices.

1.1 The problem

In a spoken dialogue system, *spoken language understanding (SLU)* seeks to extract a meaningful internal representation of the speech signal. In a system-initiative spoken

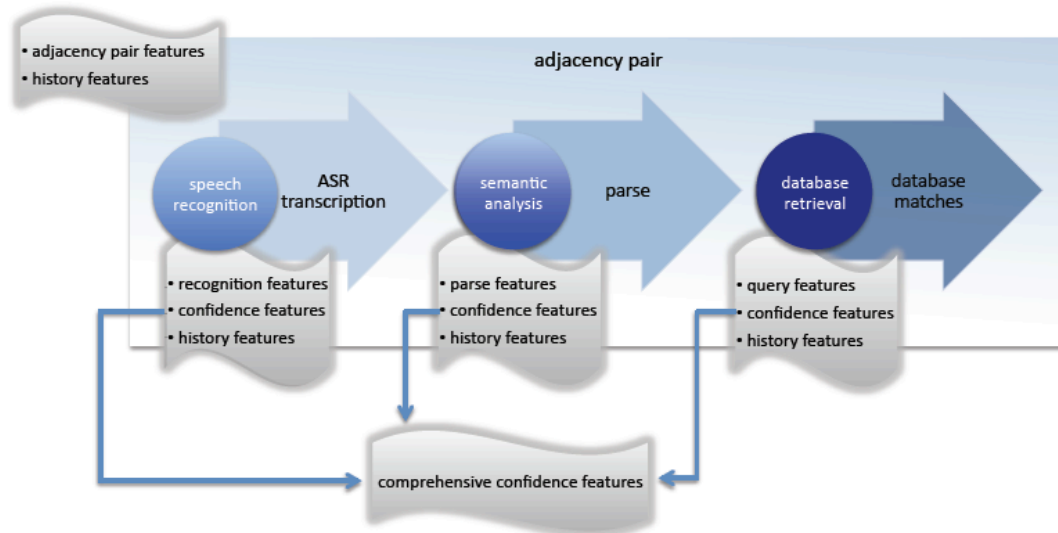


Figure 1.1: Levels of SLU processing (encompassed within an adjacency pair) and their features.

dialogue system, the immediate context of a user's utterance is an *adjacency pair*, a portion of the dialogue that begins with a system prompt that solicited the user's response and ends just before the next system prompt. An adjacency pair encompasses the SLU processing of an utterance, as in Figure 1.1.

In a traditional pipeline approach, SLU occurs at different *processing levels* (speech recognition, semantic analysis, and database retrieval) in different system components, and a host of different features is available to describe this process. First, *automated speech recognition (ASR)* transcribes the speech signal into text, and produces values for *recognition features* that describe the recognized string. Confidence features characterize the quality of the transcription, and history features describe the cumulative quality of the speech recognition from the beginning of the dialogue up to this point.

Next, *semantic analysis* determines the meaning of the recognized string, and binds it to one or more internal system concepts. This produces values for *parse*

features that describe the *parse* (the representation obtained by semantic analysis). Here, confidence features characterize the quality of the parse, and history features describe the cumulative quality of the semantic analysis up to this point in the dialogue. Comprehensive confidence measures can be derived from the combination of recognition-confidence and parse-confidence features.

When appropriate, a database query may then be used to match a parse to one or more database entries. This produces values for *query features* that describe the database retrieval. Confidence features at this SLU stage characterize the quality of the matches to database entries, and history features describe the quality of the queries thus far. Additional comprehensive confidence features can be derived at this stage by combining previously-obtained comprehensive confidence features to query-confidence features. Adjacency-pair features characterize the system prompt that initiated the current adjacency pair and other possible system internal actions. History features, here, summarize characteristics of all adjacency pairs up to this point in the dialogue. Thus, a great many features may be available during SLU.

An understanding error is either a *non-understanding* (the system fails to obtain a

-
- 1 *Into the Night*
INTO THAN 9
 - 2 *Helen and Teacher: the Story of Helen Keller and Anne Sullivan Macy*
HELEN AND TEACHER DISTORT TELL UNTIL AN AM SULLIVAN MACY
 - 3 *Map of Bones*
NAH DON'T BONES
 - 4 *I lived to Tell it All*
ELUSIVE TOTAL MAN
-

Figure 1.2: Book titles (in italics) and their noisy ASR output.

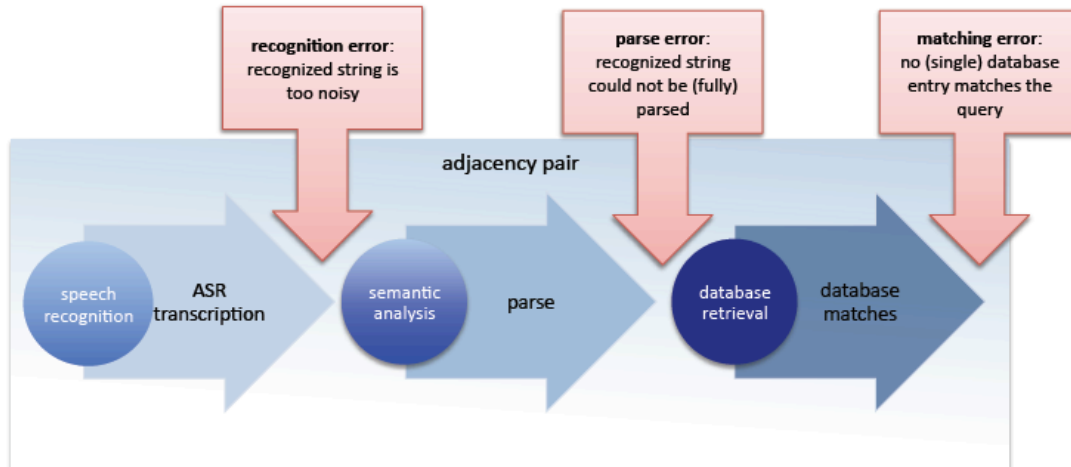


Figure 1.3: Understanding error at different SLU levels.

representation of what the speaker said), or a *misunderstanding* (the system's representation does not correspond to what the speaker said). Sometimes, even if the system fails to obtain a semantic representation of the ASR string, there is signal in the noise that, with the help of context, could be used to advance the dialogue. Figure 1.2 provides some examples of noisy ASR output for spoken book titles.

Although previous work has generally focused on misunderstandings and non-understandings alone (Bohus 2007, Bulyko, et al. 2005, Litman, et al. 2006, Skantze 2005), this work explores the use of partial content in an ASR string for disambiguation. We will refer to it as a *partial understanding*. As shown in Figure 1.3, understanding errors may occur at each SLU processing level, and characterize different degrees of understanding. For example, *recognition error* may cause non-understanding and thereby fail to support further levels of SLU processing. Recognition error may also result in partial understanding, when only some of the words are recognized confidently. Similarly, a parse error may cause non-understanding when semantic analysis fails to produce a parse, or it may produce

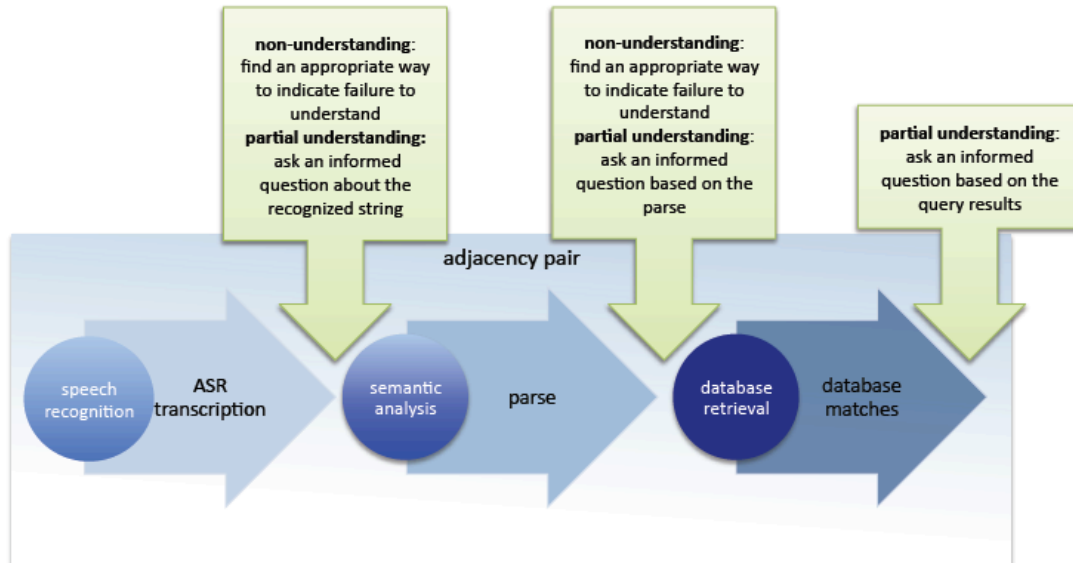


Figure 1.4: Recovery actions for different sources of error and different degrees of understanding.

partial understanding when only part of the recognized string can be parsed. A matching error can also produce partial understanding if the query cannot be matched confidently to a single database entry, but there are multiple competing matches instead. Alternatively, when the system fails to produce a parse, a matching error can produce a partial understanding if the ASR string partially matches a database entry.

As shown in Figure 1.4, different sources of error suggest different recovery actions. In particular, partial understanding at different SLU levels may suggest different kinds of informed questions the system could ask to move the dialogue forward despite the error. For example, a recognition-level partial understanding may be resolved with a disambiguation question about some word in the recognized string. As another example, a partial understanding at the database retrieval level may be resolved by disambiguation between potentially matching database entries. Moreover, in a data-driven approach to error detection, it is likely that different features will best

predict the occurrence of error at different SLU levels. Thus, this research's approach is to **decompose error detection and recovery by SLU level, and to identify the relevant features at each level** with automated feature selection.

It is limiting to assume that the only relevant features at any given error-detection level are those produced at that level; this excludes much relevant information accumulated in the SLU pipeline. For example, in recognition-error detection, recognition features alone may be sufficient to reflect that recognition is too poor to support further levels of processing. An error-detection model that also considers some parse features and context features may, however, learn a better threshold to distinguish between a non-understanding and a partial understanding. When recognition features indicate medium quality, however, a semantic analysis that could not produce a parse clearly identifies a non-understanding.

Nonetheless, all of the features available at each error-detection level need not be included at every other level; they are likely to be highly redundant and include features irrelevant to that level. For example, in parse-error detection, parse features alone are sufficient to indicate failure to parse. When different portions of the recognized string parse to different concepts, however, recognition features that indicate individual word confidence can guide parse-error recovery to ask a disambiguating question about the concept with highest word-level confidence. Similarly, when query features indicate that only one of two such concepts has a database match, query features are relevant to parse-error recovery to disambiguate only the matched concept. Finally, when query features suffice to select a well-matched entry, recognition and parse features are redundant and may introduce noise.

Some recognition and parse features could, however, select among competing database matches to formulate a question. Similarly, combined confidence features and context features may or may not be relevant at different levels of error detection.

In a spoken dialogue system, a wealth of knowledge is available about the provenance of features (i.e., the system component in which their values were generated) with respect to SLU. That **knowledge should be used to guide feature selection**. This research provides a novel feature selection approach for spoken dialogue systems that incorporates such knowledge.

1.2 Goals, approach, and contributions

The high-level goal of this research is to develop the error detection and recovery mechanisms necessary for a spoken dialogue system to contend with the understanding errors endemic to human-machine dialogue. To this end, the specific goal of this work is to **identify methods for feature selection that best support data-driven error detection and recovery**.

To achieve these goals, this research first identifies the interactions among spoken dialogue system features. To address this task, this research decomposes error detection and recovery along SLU levels to target errors at their source, as described in Section 1.1. The error-detection and recovery models are learned from a corpus of spoken dialogue interactions, whose collection is described in Chapter 3.

The main question addressed by this research is how feature selection can support error-detection and recovery given different sources of error. To address this

question, this research builds a feature selection framework, described in Chapter 4, that supports comparison experiments with multiple feature-selection approaches. To test the hypothesis that feature selection improves the accuracy of data-driven error detection and recovery, this research produced a novel feature selection algorithm that combines feature weighting with random feature sampling. To test the hypothesis that knowledge about feature relevance and interactions further supports feature selection in this domain, this research devised another novel feature selection algorithm that incorporates this knowledge as heuristic guidance. These feature selection algorithms identify features relevant to error-detection and recovery at different SLU levels. Finally, to validate the error detection and recovery models that were learned on the features identified by these feature selection algorithms, these models were implemented within a baseline system and gauged for improvement, as described in Chapter 5.

The contributions of this work include:

- An experiment that studied how people made decisions given the same input and output as a spoken dialogue system. This *Wizard of Oz* study presented a person with a speech recognition string, and gave her access to a database and the ability to select the next prompt spoken by the system. It produced a large corpus of 913 dialogues to be released to the research community. The corpus was used to produce models of error detection and recovery.
- An experimental feature selection framework that runs comparison experiments between feature selection algorithms. The experiments compare the accuracy of models learned on different sets of features to determine feature selection quality.

- Novel feature selection methods, including a randomized weight learning algorithm and a weighted-heuristic feature selection algorithm.
- The identification of sets of features for error detection and recovery at different SLU processing levels.
- Empirical results that demonstrate the feasibility of error detection and recovery strategies for successful dialogue despite noisy ASR.
- Empirical results that demonstrate how data-driven models of error detection and recovery strategies can improve overall task success.

In summary, effective and robust spoken dialogue systems must contend with the understanding errors endemic to human-machine dialogue. Thus, error detection and recovery is essential to SLU in ambitious spoken dialogue systems that expect to handle rich utterances. Because the SLU process can generate many features, the selection of appropriate features for data-driven error detection and recovery is essential (Guyon and Elisseeff 2003, Guyon, et al. 2006). Although data-driven approaches to error detection and recovery in spoken dialogue systems are increasingly popular (Bohus 2007, Bulyko, et al. 2005, Carlson, et al. 2005, Skantze 2005, Torres, et al. 2005), feature selection in this domain has not been extensively explored. This is the open question addressed by this thesis.

The feature selection framework and algorithms developed here can be applied to the identification of features to support data-driven error detection and recovery for spoken dialogue systems in other applications. The framework requires a corpus of labeled dialogue interactions, either from annotation or from a Wizard of Oz study, a data collection paradigm that is increasingly popular in the literature and discussed in

detail in Chapter 3 (Bohus and Rudnicky 2005b, Rieser, et al. 2005, Skantze 2005, Williams and Young 2004, Zollo 1999). Although data collection is expensive, it is necessary for data-driven approaches. The data collection effort, however, need not be excessive. Models learned on simulated corpora could be used to bootstrap a system that then continues to learn from its own dialogue interactions (Schatzmann, et al. 2006). Indeed, because learning on a large set of features in a complex domain can be expensive and can overfit, bootstrapping to identify the appropriate features may be essential to both accuracy and learning speed.

Furthermore, the interactions among features with respect to error detection and recovery identified in this research could also guide the development of spoken dialogue systems for other applications. Future work will validate how well the feature subsets identified here transfer directly to other applications. Finally, pipeline architectures are widely used throughout NLP (Hollingshead 2010) and employ error detection mechanisms (Xiong, et al. 2010) on potentially many features (Chiang, et al. 2009, Pado', et al. 2009). The feature selection framework and algorithms developed in this thesis could also be used to identify feature subsets for error detection and recovery in such pipelined systems, provided that the appropriate heuristic knowledge about the features generated at different processing levels can be specified.

1.3 A baseline spoken dialogue system

CheckItOut, the baseline spoken dialogue system used for this research, models book-order transactions at the Andrew Heiskell Braille and Talking Book Library, a branch of the New York Public Library and part of the National Library Service. It accesses a sanitized version of Heiskell’s database of 5,000 active patrons, and its full book database of 71,166 titles and 28,031 authors. CheckItOut receives book orders over a dedicated VOIP line. Figure 1.5 is a sample call with CheckItOut. A dialogue with CheckItOut consists of a patron login sub-dialogue (lines 1 – 12 in Figure 1.5), up to four *book request* sub-dialogues, and an order summary sub-dialogue (lines 29 – 30 in Figure 1.5).

Callers first identify themselves as patrons of the library, and then order up to four books. A book request subdialogue is one in which the system and the caller exchange information until a book is identified and ordered. For example, lines 13 to 22 in Figure 1.5 are a single book request. Callers may request books by catalogue number, title, or author. Only catalogue numbers are guaranteed to identify a book uniquely. If CheckItOut mistakenly orders the wrong book, callers may say “That’s not what I said,” and CheckItOut will retract its action, acknowledge its mistake, and re-prompt the caller for the correct book. After ordering four books, CheckItOut offers to provide a summary of the order. CheckItOut is more prone to error than most commercial dialogue systems (e.g., airline reservations), which typically handle single-word caller responses. CheckItOut must contend with a large vocabulary; its book titles and author names alone contribute 54,448 distinct words. Moreover, book

-
1. **CheckItOut:** Welcome to CheckItOut. Would you help me practice?
 2. **Caller:** SURE
 3. **CheckItOut:** Can I have your area code please?
 4. **Caller:** SEVEN ONE SIX
 5. **CheckItOut:** Did you say seven one six?
 6. **Caller:** YES
 7. **CheckItOut:** Ok, can I have your telephone number?
 8. **Caller:** FIVE THREE EIGHT ONE THREE ZERO NINE
 9. **CheckItOut:** Let's see... am I speaking with Delfina Lieberg?
 10. **Caller:** YES
 11. **CheckItOut:** and you live at 219 Broadway, New York?
 12. **Caller:** YES
 13. **CheckItOut:** Great! Let's find some books, and if I get something wrong you can say 'that's not what I said'. To request a book you can say the catalogue number, title or author.
 14. **Caller:** MY PERSONAL BEST
 15. **CheckItOut:** *Persuasion* is available. Next book?
 16. **Caller:** THAT'S NOT WHAT I SAID
 17. **CheckItOut:** Sorry, I must have misunderstood you, let's try again. Next book?
 18. **Caller:** JOHN WOODEN
 19. **CheckItOut:** Did you say JOHN WOODEN?
 20. **Caller:** YES
 21. **CheckItOut:** Would you like *My personal best life lessons from an all American journey* by John Wooden?
 22. **Caller:** YES
 23. **CheckItOut:** Next book?
 24. **Caller:** THE DESERTER
 25. **CheckItOut:** *The deserter murder at Gettysburg* is available. Next book?
 26. **Caller:** BOOK NUMBER SIX SEVEN FOUR THREE TWO
 27. **CheckItOut:** *A work in progress* is available. Next book?
 28. **Caller:** THE AUTOBIOGRAPHY OF A MONK
 29. **CheckItOut:** *The autobiography of a monk*, that's four books. Would you like a summary of your order?
 30. **Caller:** NO
 31. **CheckItOut:** Thank you for calling. Goodbye.
-

Figure 1.5: A sample dialogue with CheckItOut.

titles resemble free text much more than they resemble structured data, and there is opportunity for confusion among book titles and author names.

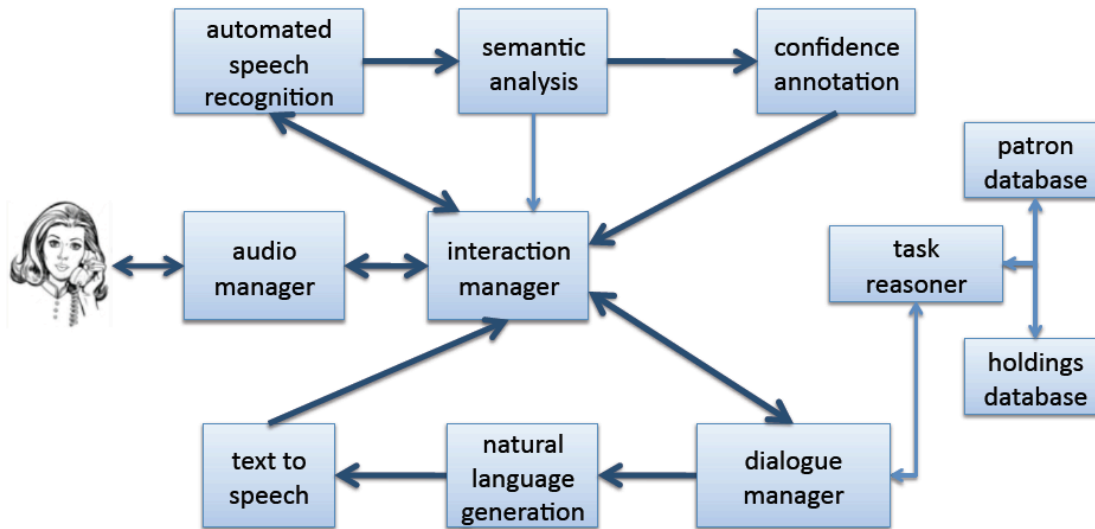


Figure 1.6: Data flow in CheckItOut. Heavy arrows emphasize the pipeline.

CheckItOut is built within the *Olympus/RavenClaw* dialogue system architecture and management framework, which has supported the implementation of more than a dozen spoken dialogue systems including Let's Go Public! (Bohus and Rudnicky 2009, Raux, et al. 2005). Although data mainly flows in a pipeline, as shown in Figure 1.6, system components communicate through the Galaxy hub (Seneff, et al. 1998).

Table 1.1 specifies the implementation contributions provided by my research to adapt Olympus for CheckItOut. It specifies the modules that were provided by Olympus and left untouched, those that were adapted for CheckItOut, and those that I implemented. The audio manager, interaction manager, and text to speech were provided within Olympus and left untouched. The databases were provided by Heiskell; I implemented the task reasoner to interface them, which was later extended by other members of the research team. This research also specifically contributed a semantic grammar, a regression model for the confidence annotator, a task

Dialogue system component	Implementation
Audio manager	Built within Olympus
Interaction manager	Built within Olympus
Automated speech recognition	Adapted: incremented acoustic model with 10 hours of spontaneous speech from our domain; generated language model
Semantic analysis	Wrote semantic grammar
Confidence annotation	Adapted: trained logistic regression model on 2,932 parses from calls with CheckItOut
Dialogue manager	Wrote task specification for the dialogue manager (C++)
Task reasoner	Wrote task reasoner to interface databases (Perl)
Patron database	Sanitized version of Heiskell's patron database
Holdings database	Full Heiskell book database
Natural language generation	Adapted: wrote templates to generate CheckItOut's prompts (Perl)
Text to speech	Built within Olympus

Table 1.1: Adaptations to Olympus contributed by this research.

specification for the dialogue manager, and templates for the natural language generation. Other members of the research team contributed adaptations to the automated speech recognition. Appendix A provides details on all development directly contributed by this research.

First, the audio manager in Figure 1.6 detects voice activity in the incoming audio signal. It extracts features from the incoming audio, segments it into small time frames (typically 10ms), and labels each frame as either speech or non-speech. Through a smoothing process, the interaction manager then refines this segmentation across the speech/non-speech regions based on a threshold for energy features (i.e., when the signal is quiet, the caller is assumed to be quiet). This approach can be a source of understanding error in the presence of background noise, because energy alone cannot discriminate between speech and noise. This process also determines the end-of-turn detection, and it is adjusted in Olympus throughout recognition,

dynamically setting its thresholds based on dialogue features (Raux and Eskenazi 2008).

Next, the automated speech recognition component in Figure 1.6 transcribes the segmented speech data. For example, in a book request, the waveform corresponding to the utterance “I would like to read The da Vinci Code” is transcribed into the string hypothesis I WOULD LIKE TO READ THE DA VINCI CODE. This transcription relies on an *acoustic model* that maps speech data to phonemes, a *lexicon* that describes the possible phoneme sequences in words, and a *language model* that indicates the probability of sequences of n words (*n-grams*). CheckItOut uses freely available acoustic models of Wall Street Journal dictation speech, adapted with about 10 hours of spontaneous speech in our book-request application. The lexicon and language models are generated from a grammar with *Logios*, a language compilation suite for dialogue systems (Rudnicky 2008). To produce a pseudo-corpus of likely sentences, Logios uses the productions of a semantic grammar (discussed next) to generate strings of words that conform to the grammar. This pseudo-corpus is then used by a statistical language modeling toolkit (*CMU SLT*) to create an n -gram language model. The speech recognizer may at times generate multiple hypotheses. For each hypothesis, it generates scores from its language and acoustic models, as well as word-level and utterance-level confidence scores. In spoken dialogue systems, speech recognition accuracy is worse than with single-user speech or laboratory conditions, and is often the source of understanding errors (Raux, et al. 2005). Of the available speech recognizers in Olympus, CheckItOut uses *PocketSphinx* (Huggins-Daines, et

al. 2006), a fast and efficient recognizer with comparatively good performance across applications (Yao, et al. 2010).

Semantic analysis determines the semantic content of the ASR hypothesis with respect to a grammar. Olympus' *Phoenix* semantic parser in Figure 1.6 produces one or more semantic frames per ASR hypothesis (Ward and Issar 1994). In Phoenix, semantic frames can include multiple semantic slots. For example, in the Phoenix grammar for CheckItOut, the semantic frame *BookRequest* includes the semantic slots [*Title*], [*Author*], and [*BookNumber*], and the semantic frame *Generic* includes semantic slots [*Confirmation*], [*Rejection*], and [*NotWhatISaid*]. Each Phoenix slot corresponds to an internal system concept that determines its semantic content. For each slot, a *context-free grammar*, whose rules' left-hand side can be replaced with any of the tokens on the right-hand side (terminals or non-terminals), produce a *parse*, the semantic structure of the sentence. When some words do not parse, a semantic frame can be a discontinuous sequence of slots. For example, the ASR CORRECT .ASHENDED. NOT WHAT I SAID is parsed to the semantic frame *Generic* as two discontinuous semantic slots that ignore the unconfident ASR word .ASHENDED. which could not be parsed:

```
{c slot :name [Confirmation] :contents CORRECT :frame Generic}
```

```
{c slot :name [NotWhatISaid] :contents NOT WHAT I SAID :frame Generic}.
```

Returning to our running example, the ASR hypothesis I WOULD LIKE TO READ THE DA VINCI CODE is mapped to the semantic representation

```
{c slot :name [Title] :contents THE DA VINCI CODE :frame BookRequest }
```

where [*Title*] is a semantic slot whose associated context-free grammar parses the THE DA VINCI CODE portion of the ASR hypothesis. This, in turn, causes the internal system concept `book_title` to be assigned as

```
book_title = THE DA VINCI CODE
```

Multiple competing parses may be generated, and the Helios confidence annotation component is used to select the best parse with highest confidence or to determine that no parse is confident above a given threshold (Bohus and Rudnicky 2002). The confidence model used in CheckItOut is a logistic regression on the ratio of words not covered by the parse to total number of words in the utterance. The logistic regression model was trained on 2,932 parses from calls with CheckItOut.

To handle the large vocabulary and rich syntax of book titles, CheckItOut augments Phoenix by parsing the 71,166 titles in the database with the *MICA parser*, a large-coverage dependency grammar (Bangalore, et al. 2009, Gordon and Passonneau 2010). In a dependency grammar, the relation between a word and its *dependents* (tokens associated with it) determines structure, rather than word order. The parses are then mapped to the context-free grammar format used by Phoenix and associated with the Phoenix [*Title*] slot. Ambiguity in parsing may be caused by overlaps between the author field and the title field in the book database. Moreover, title fields are often hard to predict, because they are closer to free text than they are to structured data. Book requests in CheckItOut are, therefore, challenging for semantic analysis, and often a source of understanding error.

The dialogue manager (*DM*) in Figure 1.6 determines the system's response based upon the semantic representation of the caller's speech, the current context of the

dialogue, and the task constraints (e.g., a match in the database). In our running example, the DM may decide to ask for an explicit confirmation of the requested title, and output the *dialogue act* (a semantic representation of the system response):

explicit_confirm: book_title = THE DA VINCI CODE

The dialogue manager is usually also responsible for the update of the dialogue context based on the caller's input speech and system response.

The *RavenClaw* dialogue management framework used in Olympus decouples task-specific implementation from task-independent aspects of dialogue management, such as task-independent error handling (Bohus and Rudnicky 2003, 2009). The overall infrastructure of RavenClaw relies on an expectation agenda and a set of dialogue agents that are executed one at a time. The task is specified as a set of task-specific dialogue agents managed by the task-independent dialogue engine that oversees their execution. For this research, as required by Olympus, we developed CheckItOut's task-specific agents organized into a *task tree*, a hierarchical plan for the overall conversation. This tree is executed depth-first, subject to preconditions on each agent, which may alter the execution traversal.

Figure 1.7 illustrates a portion of CheckItOut's task tree. The full task tree appears in Appendix B. Internal nodes (ovals in Figure 1.7) manage the execution of their children. Leaf nodes (rectangles in Figure 1.7) issue prompts and bind concepts. A conversation with CheckItOut begins with the execution of the *Welcome* agent that issues a prompt to greet the caller and asks if she would like to continue with the call. If the caller's answer is parsed to a confirmation, the *Login* agent is executed next. It manages, in turn, the execution of its child nodes. First, Login executes the

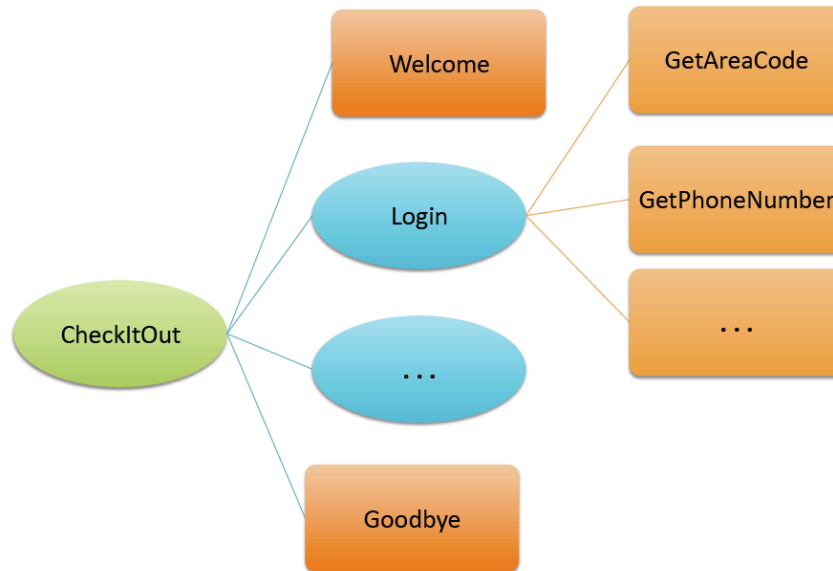


Figure 1.7: A portion of CheckItOut’s task tree.

GetAreaCode agent, which issues a prompt that asks for the caller’s area code and attempts to bind the resulting `area_code` concept. Once the `area_code` concept is bound, *Login* executes *GetPhoneNumber*, which similarly requests and binds the `telephone_number` concept. If the answer to the *Welcome* prompt were a rejection instead, the *CheckItOut* node would override the depth-first execution and jump to the *Goodbye* agent instead.

In Olympus/RavenClaw, task-specific information is handled in the *task reasoner*, a separate component that closely interacts with the dialogue manager to interface with one or more *backends* (databases or Internet lookups). The task reasoner retrieves data from the backend, performs the necessary data manipulation, and returns the data to the dialogue manager in the desired format. This component is referred to as the *domain reasoner* in Olympus/RavenClaw. Because in this document we use the word “domain” to refer to spoken dialogue systems, and the word “task”

to refer to a particular application, we rename this component “task reasoner” here for clarity.

Natural language generation (*NLG*) in Figure 1.6 takes a dialogue act from the DM, and outputs a text transcription in natural language. In our running example, the dialogue act

```
explicit_confirm: book_title = THE DA VINCI CODE
```

would be translated into the string “Did you say The da Vinci Code?” *Rosetta*, Olympus’s *NLG* component, is based on a simple mapping between dialogue acts and text, using a template that accepts concepts and generates prompts. A template contains some fixed words and some variables whose values are input concepts from the interaction.

For example, if the *book_title* concept is assigned the value *THE DA VINCI CODE*, the template “Did you say <*book_title*>?” will generate the prompt “Did you say The da Vinci Code?” More advanced approaches include the use of corpus-based stochastic mapping between dialogue acts and text translations (Oh and Rudnicky 2000), and greater interaction with the dialogue manager using distinct generation agents with different realizations (e.g., different languages, different levels of politeness) to increase adaptivity (Wilcock and Jokinen 2003).

Finally, the text-to-speech (*TTS*) component in Figure 1.6 generates an audio waveform for the speech translation of the *NLG* output. The most natural-sounding *TTS* uses pre-recorded prompts; it is most appropriate when all system prompts can be pre-defined and pre-recorded by a voice talent. In many systems, however, not all

prompts can be predefined. In CheckItOut, *Kalliope* (Olympus' TTS module) is configured to use Microsoft's Speech Application Programming Interface (SAPI).

In summary, a baseline spoken dialogue system for a book-ordering application is used to support this research. This system is built upon a standard spoken dialogue system architecture whose components work in a pipeline to process the input speech and determine an appropriate system action. Our book-ordering application is challenging and error prone due to its large vocabulary, the unrestricted language of book titles, and possible confusability among titles and author names. The next chapter discusses related work on data-driven error detection and recovery in spoken dialogue systems.

Chapter 2

Related work

To achieve high speech recognition accuracy, commercial spoken dialogue systems impose a rigid sequence of user responses with limited content. These systems typically accept only a small vocabulary of command-like words or short sentences. Systems that achieve high recognition accuracy on large vocabularies are generally for single-party applications (i.e., customized to one specific user), and do not generalize to multi-user dialogue. In contrast, fielded dialogue systems must contend with word error rates (*WERs*) of 30% at best, and as high as 70% (Raux, et al. 2005). In the presence of misunderstanding, the user's need to correct the system generally results in even poorer recognition due to frustrated, hyper-articulated speech and out-of-vocabulary non-correction utterances (Litman, et al. 2006). To prevent misunderstanding, a system may request user confirmation. Confirmation can be *explicit* (e.g., "Did you say The da Vinci Code?") or *implicit* (e.g., "The da Vinci code. We have that.")

Implicit confirmation can use language that elicits user responses that are understood by the system (Raux and Eskenazi 2007). Ideally, dialogue should be flexible, habitable, and efficient. Users, however, prefer systems that minimize non-understandings and misunderstandings, even at the expense of dialogue efficiency. Although mixed-initiative and implicit confirmation generally lead to fewer turns for the same task, users of the TOOT train information system preferred a system-

initiative approach to mixed-initiative or user-initiative, and preferred explicit confirmation to implicit or no confirmation (Litman and Pan 1999). If the user adds new information to a response, then two-pass recognition can be used to consider any additional information in the second pass that restricts the expected vocabulary, and thereby achieve better recognition (Stoyanchev and Stent 2009). In the presence of non-understanding, a data-driven enriched strategy can be used to move the dialogue forward (Bohus and Rudnicky 2005b).

When previous work for spoken dialogue systems has used data-driven approaches for recovery-action selection, it has not extensively explored the role that different features play. Moreover, in the context of error detection and recovery, previous work focused on a single error detection and recovery decision point in dialogue, such as a response to a non-understanding (Bohus and Rudnicky 2005b) or a clarification action (Skantze 2005). This thesis extends previous work through the use feature selection and the decomposition of error detection and recovery into multiple decision points along the SLU pipeline. Section 2.1 introduces Wizard of Oz studies and discusses previous work that applied these studies to spoken dialogue systems. Section 2.2 focuses on previous work that used data collected through Wizard of Oz studies for data-driven error detection and recovery in spoken dialogue systems.

2.1 Wizard of Oz studies for spoken dialogue systems

To analyze human error detection and recovery for use in spoken dialogue systems, researchers often use Wizard of Oz (*WOz*) studies. Originally a common approach to iterative prototype design, a *WOz* study gathers information about the characteristics of a successful system before its development (Dix, et al. 2003). In a *WOz* study, only an interface is provided between the users and what they believe to be a system; a person (the *wizard*) is “behind the curtain” instead, manually controlling some or all aspects of the system’s operation. This permits the system designer to observe how people respond to certain system functionalities, to study user behavior and expectations, and to assess the best design before devoting any effort to initial prototypes.

More recently, *WOz* studies have shifted the attention from the user to the wizard. The intent of these studies has been to analyze a wizard’s behavior and then incorporate it into the system design. Such *WOz* studies can assemble corpora from which it is possible to predict wizards’ error detection and recovery actions.

Some *WOz* studies have analyzed error detection and recovery in human wizards, but did not collect any features to construct models for system improvement. These studies did, however, demonstrate that there is much to be learned from a human wizard. For example, Zollo collected seven dialogues with different wizard-caller pairs to develop an evacuation plan (Zollo 1999). Wizards had access to ASR only. With an overall WER of 30%, wizard utterances indicated failure to understand in

only 35% of the cases where the ASR was incorrect. Wizards generally ignored words that did not pertain to the dialogue context and could make educated guesses based on context and phonetic similarities.

In another study, Williams and Young collected 144 dialogues for tourist requests for directions and other information (Williams and Young 2004). The ASR presented to the wizards was simulated (an artificially corrupted transcription of the user's speech), and WER could be manipulated to be high, medium, or low (63%, 46% and 32%, respectively). Their results show that under medium WER, task-related questions in response to a non-understanding led to full understanding more often than explicit repairs did. Under high WER, however, the reverse was true. Even with high WER, wizards were able to assist users and successfully complete dialogues.

Yet another study produced a corpus of 70 dialogues to simulate a multimodal MP3 player with access to a database of 150,000 music albums (Kruijff-Korbayova, et al. 2005). Responses to user requests could be presented verbally or graphically, and six wizards had access to simulated ASR only. To produce WER, the transcription corruption was of several forms: multiple matches in the database, lexical ambiguities, and errors at the acoustic level by random deletion of parts of the user's speech input. The wizards' clarification strategies varied in the modality used to elicit disambiguation and in whether the selection of confirmation was explicit or implicit. In the noisy transcription condition, wizards made clarification requests about twice as often as found in similar human-human dialogue.

2.2 Data-driven error detection and recovery in spoken dialogue systems

Some studies have collected system features to support data-driven error detection and recovery for system improvement. Skantze and Edlund used data-driven models of early error detection: detect whether a recognized word was present in the user utterance (Skantze and Edlund 2004). Ten features were categorized either as confidence features (e.g., recognizer word confidence), lexical features (e.g., the word), contextual features (e.g., the part of speech of the previous word), or discourse features (e.g., whether it was a previously mentioned content word). They trained a model on all 10 features and showed that it outperformed models trained on features from individual categories. There was, however, no detailed analysis of the interactions and combinations of subsets of these features.

A spoken dialogue system often uses some measure of confidence to deem the semantic interpretation confident or unconfident. This measure varies across systems and is usually a function of speech recognition and parse properties (Torres, et al. 2005). It may also include some state information, under the assumption that the tradeoff between the cost of accepting an incorrect parse and rejecting a correct parse may vary across utterance type (Bohus and Rudnicky 2005a). For example, the parse may depend upon whether the user utterance was in response to an open question, a question for a specific response, or a yes/no question. One motivation for the decomposition of error detection along the SLU pipeline in this thesis is that such tradeoffs may differ across dialogue systems and that feature selection can indicate

whether such decisions should be based solely on the confidence metric used by the system or on other system-available features as well (e.g., the results of a backend query).

Bohus used a WOz study to analyze the responses of a single wizard to non-understanding in a system for conference room reservations (Bohus 2007, Bohus and Rudnicky 2005b). The wizard had full access to the users' speech, and would intervene only in case of a system non-understanding. The experiment provided a pre-defined set of 10 actions to recover from a non-understanding, and the wizard selected from among them. For example, the wizard could have the system ask the user to repeat, ask the user to rephrase, or ask the user to repeat and provide hints about what she could say. Results showed that the wizard's responses proved to be more successful than random selection. This experiment produced a corpus of 449 dialogues, from which a data-driven model of wizard recovery actions was produced with logistic regression on approximately 70 features drawn from different system components. These included 20 speech recognition features that described the quality of the ASR, 24 semantic analysis features that described the quality of the parse, 6 lexical features that described the presence of confirmation or disconfirmation markers in the user utterance, 3 features that indicated whether there had been a series of consecutive non-understandings, 8 features that compared the 2 hypotheses obtained from the concurrently-running male and female recognition engines, and 9 features that described the dialog state and dialog history (e.g., the average confidence score in the dialogue). It is unclear which features had greater impact on the prediction.

Rieser and Lemon used the Kruijff-Korbyova MP3 corpus, discussed in Section 2.1, to obtain data-driven models of multimodal clarification strategies (Rieser and Lemon 2006). These models were then successfully incorporated into a multimodal dialogue system. They used Naïve Bayes and decision trees to learn the wizard's strategy from 16 features available to the system at runtime. There were three kinds of features. Five local features were system related, such as the number of hits in the database and the speech act of the previous user utterance. Six dialogue history features described dialogue outcome thus far, such as the number of clarifications requested, how often screen output had been used, and the duration of the dialogue up to the current turn. Finally, five user-model features described the user's behavior during the interaction, such as how often the user clicked, employed verbal input, and had seen a given screen.

In the studies discussed thus far, the problem of feature selection has been neglected. Rieser and Lemon, however, did use feature selection methods to determine which features influenced the wizard's strategy selection. Comparison with different feature selection methods and learning algorithms showed that wizards did not behave arbitrarily; their actions were based on specific features of the dialogue. Results also showed that the models that best predicted wizard action used features from different categories and that feature selection improved action prediction compared to models that used all the features.

This thesis considerably extends the analysis of the role of system features. It uses a much larger set of 163 initial features to obtain multiple data-driven error detection and recovery models at different stages in the SLU pipeline. Two novel feature

selection algorithms, described in Chapter 4, are devised to identify different feature subsets for each model. Of the learned models of wizard-informed error detection and recovery, three are then evaluated in Chapter 5. These models are implemented and incorporated into the baseline system CheckItOut to produce CheckItOut+. The next chapter describes a large-scale WOz study that produced a rich corpus to support this research.

Chapter 3

Data collection: a wizard of Oz study

To analyze human error detection and recovery, and to collect system features to model their behavior for use in spoken dialogue systems, we embed a wizard module into CheckItOut, as illustrated in Figure 3.1 (Passonneau, et al. 2011). Essentially, the wizard replaces the dialogue manager, and decides what action to take next. The wizard is *ablated*, in other words, the wizard's input and actions are restricted to those that would be available to the system (Levin and Passonneau 2006).

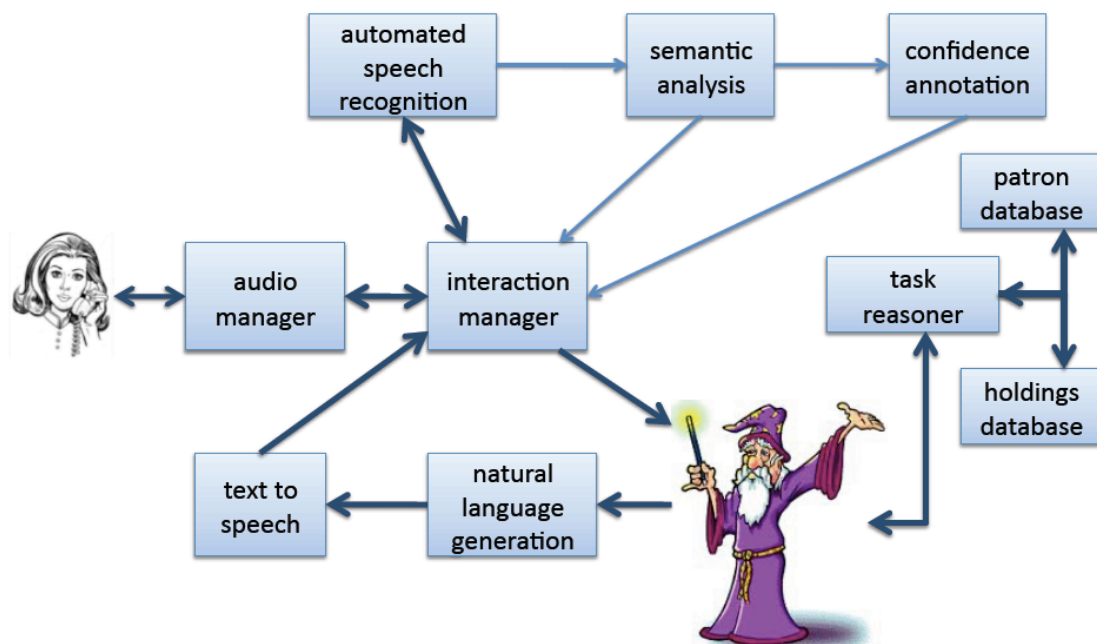


Figure 3.1: An embedded wizard in CheckItOut. Heavy arrows indicate dataflow for the wizard.

Unlike the dialogue manager in CheckItOut, the wizard receives the ASR string, and can perform *voice search*, that is, directly query the database with the ASR. The wizard receives in return a list of candidate matches ranked by similarity score (Sherwani, et al. 2007). Voice search thereby provides useful context for ASR that might otherwise be unintelligible. For example, the ASR SOONER SHEEP MOST DIE is easily identified in the context of the candidate book-title matches:

Soon She Must Die

Why Someone Had To Die

The Messenger Must Die

We used Ratcliff/Obershelp (*R/O*) pattern recognition to measure the similarity of an ASR string to entries in the database (Ratcliff and Metzener 1988). The *R/O score* is the number of matching characters divided by the total number of characters in the two strings. For example, in a database search by title on the ASR string ROLL DWELL, the three top scoring candidate titles are

Cromwell (R/O = 0.666)

Colin Powell (R/O = 0.636)

Robert Lowell (R/O = 0.608)

Based on the ASR and the results of one or more database searches, the wizard then issues the next system prompt. Although a more sophisticated query function could be used to improve the matching (Lee, et al. 2010), our focus here is to capture a wizard's problems solving ability. The wizard is also *embedded*, that is, the wizard

and the system process information concurrently, so that run-time system features can be collected to train data-driven models of wizard behavior (Passonneau, et al. 2010).

Wizards are ablated, that is, restricted to the same ASR and query results as CheckItOut, and to a fixed set of actions. Wizards, however, rely on their knowledge of the real world and of the English language (e.g., semantics and sound similarity (rhyme)) to make judgments about how reasonable the ASR is within the context of the dialogue and query results. Our ability to embed a wizard and predict her actions from system features alone suggests that these features can be a proxy for a wizard's informed judgment.

Our full dialogue WOz study, discussed in Section 3.2, differs from the WOz studies discussed in Section 2.1 in three fundamental ways. First, our study is on a much larger scale in number of dialogues, number of system-features, and number of wizards. Second, we analyze and identify differences among wizards, identify successful behavior, and train models of error detection and recovery to improve a baseline system only from our best wizards. Third, we make extensive use of feature selection to identify the features that best support our models of wizard behavior.

Book requests by title are more challenging than those by author or catalogue number. As previously discussed, the title field is much more like free text than it is like structured data, and it is therefore less predictable and more prone to error. A comparison between two Olympus/RavenClaw systems clearly illustrates the Phoenix parser's difficulty with book requests (Gordon and Passonneau 2010). Under four WERs (from perfect to very poor), book requests by title and by author in CheckItOut were compared to requests for bus information by route and by destination in Let's

Go Public! (Raux, et al. 2005). Requests in CheckItOut were longer and more challenging at every level of WER. For example, with a WER of 0.4, concept accuracy in Let's Go Public! was 74% compared to 36% for CheckItOut.

An offline *pilot study* for book requests by title demonstrates that people are very skilled at matching noisy ASR for book titles to candidate matches (Passonneau, et al. 2009a). Three subjects spoke 50 titles each, producing 3 sets of ASR transcriptions with a high WER that ranged between 0.69 and 0.83. Each set of ASR transcriptions was then presented to one of three different subjects who were asked to identify the correct title for each ASR transcription from among the 71,166 database titles in a plain text file. There were no restrictions on how the subjects searched the text file or how long they had to complete the task. In spite of the high WER, subjects were able to identify 74% of the titles correctly.

Motivated by the promising results of the pilot study, and to further test people's ability to make sense of noisy ASR when playing the role of an embedded wizard, we performed a restricted WOz study, again isolating the most challenging aspect of dialogues in CheckItOut, namely the disambiguation of noisy ASR for book requests by title. In this preliminary study, wizards were not embedded within the full CheckItOut system as depicted in Figure 3.1; they only used CheckItOut's automated speech recognition and databases. This *title experiment*, described in detail in Section 3.1, demonstrates not only that wizards are surprisingly skilled at disambiguating book titles from noisy ASR, but most importantly that their actions can be modeled using system features alone. Supported by the results of the title experiment, we then performed our *full-dialogue experiment*, where wizards are

embedded within the full CheckItOut system as depicted in Figure 3.1; this experiment is detailed in Section 3.2. Section 3.2.1 provides details on the experimental design. Section 3.2.2 describes the produced corpus, one of the main contributions of this thesis, and analyses the wizards' behavior and their skills at the task. Section 3.2.3 describes the system features we collected during this full-dialogue experiment.

3.1 Preliminary title experiment

In this study, callers and wizards engaged in single *title requests*: the caller spoke a title and in response the wizard would either offer a book title or ask a question (Passonneau, et al. 2009b). To present our wizards with a challenging task, we aimed at high but not intractable WER, close to that of the pilot experiment. We built a trigram language model from 7,500 book titles in our database using the CMU Statistical Language Model Toolkit (Clarkson and Rosenfeld). The average WER in this experiment was 71%. The pilot study indicated that single-word titles were easy for wizards to identify, so we eliminated them here. We also eliminated book titles with below-average circulation to make our task more similar to likely book-ordering transactions. The 7,500 titles used to build our language model were randomly sampled from among the remaining 19,708 titles. Voice search, however, matched ASR strings against the full database of 71,166 titles.

3.1.1 Experimental design

Seven undergraduate computer science majors at Hunter College participated in our study. Two were non-native speakers of English (one Spanish and one Romanian). Each of 21 possible pairs met for five *trials*, during which one student was the wizard and the other was the caller for a *session* of 20 title requests. They then immediately reversed roles for a second session. The experiment could have yielded 4,200 title requests (21 pairs \times 5 trials \times 2 sessions \times 20 titles). Participants were, however, permitted to terminate a session early; we collected data for 4,172 titles requests in all.

Callers and wizards sat in separate rooms. For this experiment, the caller did not

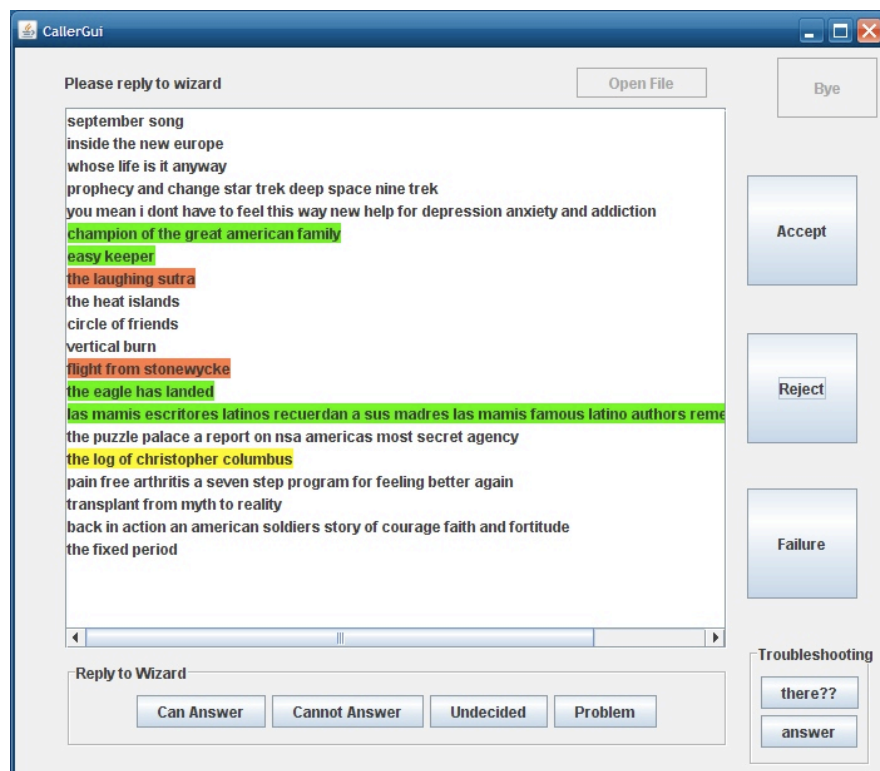


Figure 3.2: The caller GUI for the title experiment.

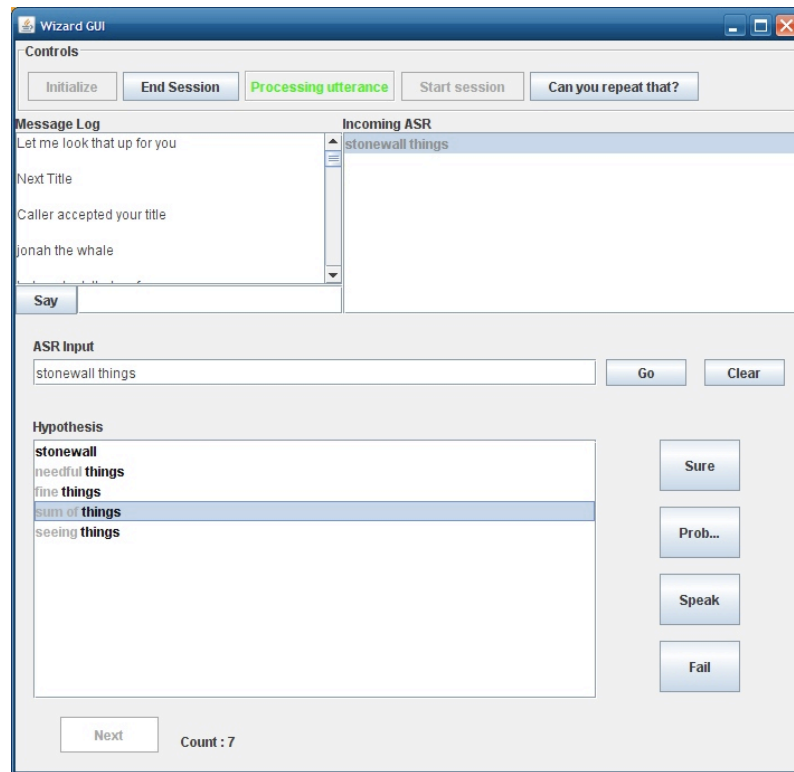


Figure 3.3: The wizard GUI for the title experiment.

call the system by telephone, but had instead a headset with microphone and a graphical user interface (*GUI*), illustrated in Figure 3.2. The wizard also had a microphone and GUI, as illustrated in Figure 3.3, but did not have access to the caller's speech. Instead, the wizard GUI displayed a live feed of the ASR transcription with unconfident words surrounded by periods. Session titles were randomly selected with replacement from among the 7,500 used to build the language model. Callers were given a list of 20 titles prior to each session. To elicit voice quality more like a book request than a mechanical reading of items from a list, we familiarized callers with the titles; they were asked to read a synopsis for each title and number the titles in a way that reflected some logical grouping. Titles were then requested in that order during the session.

For a title request, the caller indicated the book she was about to request from among the 20 displayed on her GUI, and spoke the book title into the microphone. CheckItOut's speech recognizer transcribed the caller's speech. The wizard, in a different room, then saw the ASR transcription of the book title appear on her screen, and searched the database with it. Based on the ASR and the query results, the wizard's task was to guess which candidate title, if any, was correct.

The query results could be of four types:

- *Singleton*: a single best candidate match ($R/O \geq 0.85$)
- *Ambiguous list*: two to five moderately good candidates ($0.85 > R/O \geq 0.55$)
- *Noisy list*: six to ten poor but non-random candidates ($0.55 > R/O \geq 0.40$)
- *Empty*: no candidate titles ($\max R/O < 0.40$)

The distribution of query result types during the experiment was 46.7% singleton, 50.5% ambiguous list, 2.8% noisy list and 0% empty. (During pilot tests 5%-10% of query results had been empty.)

Words in each candidate title that matched a word in the ASR appeared darker on the GUI. They were dark black for singleton or ambiguous lists, and medium black for noisy lists. All other words were in grayscale in proportion to the degree of character overlap. After search, the wizard had four options: make a *confident choice* of a candidate title, make a *tentative choice*, ask a *question*, or *give up*. A tentative choice is one that in a wizard-based spoken dialogue system may require some kind of clarification. If the wizard selected a title, it was synthesized by CheckItOut's text-to-speech module. If the wizard asked a question, she spoke it through a microphone. The caller heard the wizard's response through her headset and scored a choice as

correct or incorrect, or rated a question based on her ability to answer (“can answer,” “cannot answer,” “cannot understand,” “not sure if relevant”). The caller did not answer the question, but rated the wizard’s response, and then began a new title request.

The caller’s rating of the wizard’s response would also appear on the wizard GUI. Book titles on the caller GUI were highlighted in green if correctly identified by the wizard, in red if not correctly identified by the wizard, in yellow if in progress, and not highlighted if still pending. To encourage cooperation among our participants, we offered a prize to the participant with the highest *experiment score*. The wizard scored +1 for a correctly identified title, +0.5 for a thoughtful question, and -1 for an incorrect title. The caller scored +0.5 for a successfully recognized title.

3.1.2 Evaluation of wizards and callers

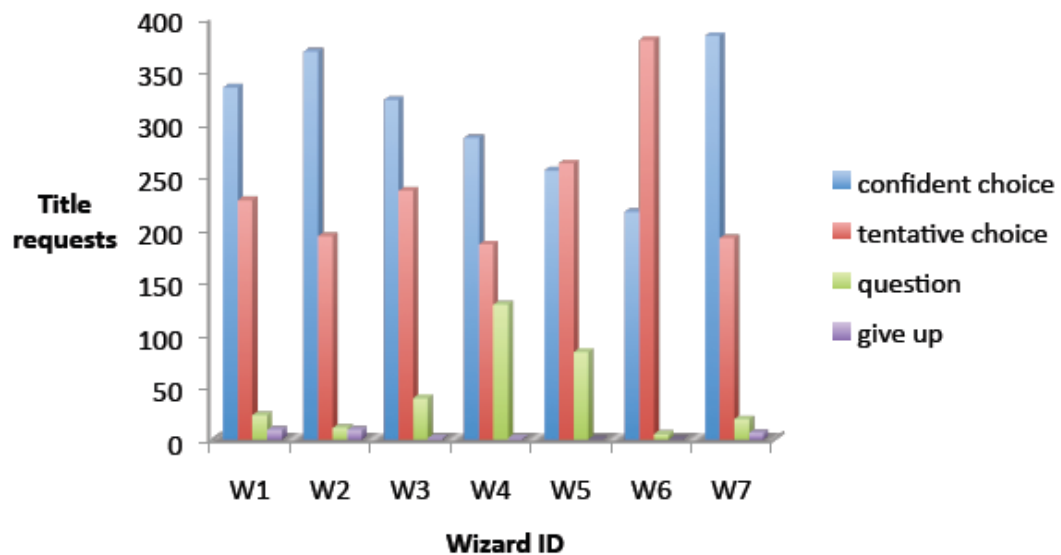


Figure 3.4: Distribution of actions selected by each wizard.

Table 3.1: Wizards ranked by *accuracy* (proportion of times the wizard chose the correct action). Accuracy is correlated to the raw *experiment score* (points assigned to the wizards based on their action). Those wizards with the highest frequency of correct non-selections were the most successful at the task. W3 and W6 are non-native speakers of English.

Wizard	Title requests	Accuracy	Experiment score	Selected candidate 1	Correct non-selections
W4	600	0.8550	0.7585	0.70	0.64
W5	600	0.8133	0.7584	0.76	0.43
W7	599	0.7346	6.6971	0.76	0.14
W1	593	0.7319	0.6936	0.79	0.16
W2	599	0.7212	0.6703	0.75	0.10
W3	581	0.6954	0.6648	0.81	0.20
W6	600	0.6950	0.6103	0.86	0.03

Figure 3.4 illustrates the distribution of wizard actions. Every wizard saw a similar distribution of query result types: singleton ($\mu = 278.57$, $\sigma = 21.16$), ambiguous list ($\mu = 300.57$, $\sigma = 16.92$), and noisy list ($\mu = 16.86$, $\sigma = 4.78$). Ideally a wizard would identify the correct title when present among the candidate titles and, when appropriate, ask a thoughtful question if it was not. Despite the high WER, when the correct title was present among the candidates ($N = 2986$), our wizards did remarkably well at identifying it. They selected the correct title either confidently (68.72%) or tentatively (26.53%), for a remarkable total of 95.25% of the time. Our wizards differed greatly, however, in their performance when the correct title was not among the candidates, a situation that occurred 28.36% of the time.

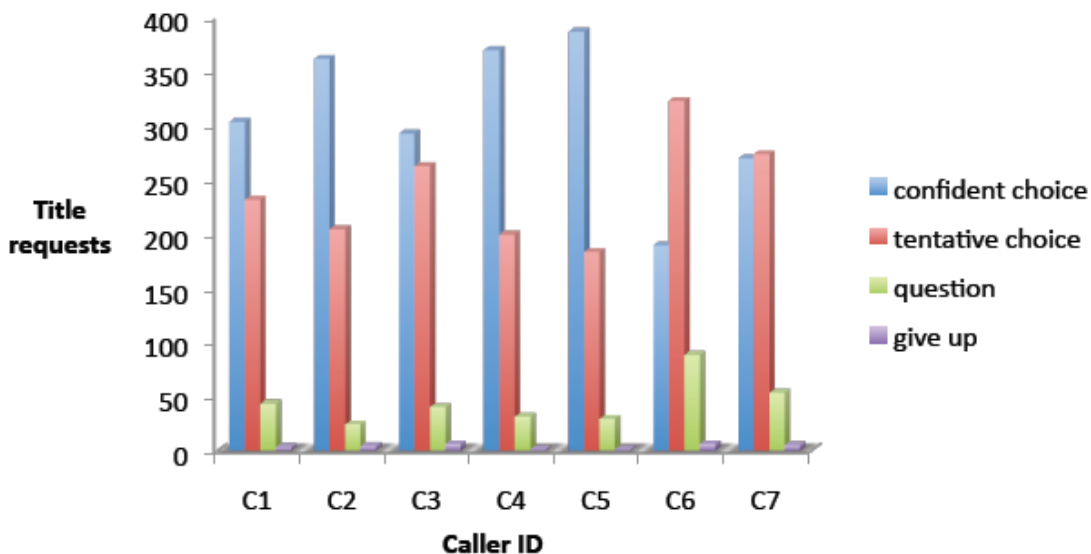
Table 3.1 ranks our wizards by *accuracy*, that is, the proportion of title requests where the wizard selected the correct title or correctly guessed that the title was not present (asked a question or gave up). The experiment score we used to motivate our participants and elicit cooperation between callers and wizards was highly correlated with accuracy. The fifth column in Table 3.1 indicates how often wizards selected the

Table 3.2: Distribution of correct actions

Correct action	N	%
Select candidate 1	2722	65.2445
Select candidate 2	126	3.0201
Select candidate 3	56	1.3423
Select candidate 4	46	1.1026
Select candidate 5	26	0.6232
Select candidate 6	0	0.0000
Select candidate 7	7	0.1678
Select candidate 8	1	0.0002
Select candidate 9	2	0.0005
Question or give up	1186	28.4276
Total	4172	100.0000

first candidate. The sixth column indicates how often wizards correctly identified that the correct title was not among the candidates, that is, asked a question or gave up.

Table 3.2 shows the distribution of correct actions, that is, which candidate (if any) matched the correct title. This distribution illustrates that it was possible to achieve an accuracy of approximately 65% by always selecting the first candidate. The fifth column in Table 3.1 illustrates that those wizards who relied on this simple

**Figure 3.5:** Distribution of actions elicited by each caller.

strategy the most (W3 and W6) were the least skilled at the task. Clearly, the most skilled wizards were those who could recognize when the correct title was not present among the candidates, as shown in the last column of Table 3.1. Wizard responses also varied greatly with the callers, as shown in Figure 3.5. Wizard and caller ID number correspond to the same subject (e.g., W1 and C1 are the same person). C6, the caller who elicited many more tentative choices, was a non-native speaker of English.

3.1.3 Models of wizard behavior

Our wizard analysis in Section 3.1.2 uncovered distinctive wizard strategies. Although previous WOz studies did not focus on individual wizard behavior, we were able to identify those wizards that were most skilled at the task. In anticipation of our full dialogue experiment, in which our models of wizard actions were used to improve a baseline spoken dialogue system, our ultimate goal was to model our best wizards. To test our ability to learn from individual wizards as well, we trained one *overall model* using data from all wizards, and seven individual *wizard models*. To confirm the learnability and quality of our predictions, we learned all models using both decision trees and logistic regression.

To test our ability to learn wizard actions from system features alone, we identified a preliminary set of 60 features (later extended in our full dialogue experiment) that would be available at run time. Seven features described the current title session (e.g., the number of the current title request). Eighteen features described

the quality of the ASR transcription and confidence measures (e.g., the average word confidence). Thirty-one features described the similarity between the ASR string and the title candidates (e.g., the average number of exact word matches). Because the number of candidate titles varied, features pertaining to title candidates were averaged. Finally, four features described the relevance of the ASR to the database (e.g., the frequency of ASR words in the database).

Although we did not intend to explore feature selection extensively in this experiment, preliminary cross correlations indicated that many of the 60 features were heavily interdependent. We isolated groups of features with $R^2 > 0.5$, and when features in a group had similar semantic meaning, we retained a single representative feature. For example, from a set of features that described the similarity between the ASR and the candidate titles, we retained the one that was most comprehensive (the number of exact word matches). We also summarized, by a single representative feature, each of the following sets: three features that described the gaps between exact word matches, three that described the query result type, nine that described various system confidence scores, and three that described the user's speech rate. This rudimentary feature selection process left us with 28 features from which to learn models of wizard responses.

Next, for each of our eight models, we ran the CfsSubsetEval feature selection algorithm available in the Weka data mining software suite (Hall, et al. 2009). This greedy hill-climbing algorithm evaluates feature subsets. It maximizes features' relevance to the label and minimizes redundancy among the features in the subset. CfsSubsetEval further reduced our 28 features to between 8 and 12 features per

model. To further reduce overfitting in decision trees, we enabled pruning and subtree raising, as included in Weka’s implementation of the C4.5 decision tree algorithm (Quinlan 1993). (We discuss feature selection further in Chapter 4, which extends our exploration of the effects of feature selection in the full dialogue experiment and discusses our novel approaches.)

Our ability to predict wizard actions was comparable across learning algorithms, as measured by model accuracy (percent of instances correctly classified) and F measure (a traditional comprehensive measure of precision and recall). For the overall model, accuracy for the logistic regression was 75.2%, and for the decision tree it was 82.2%. As previously discussed in Section 3.1.2 and illustrated in Figures 3.4 and 3.5, most of our wizards either made a confident choice or a tentative choice. Data labeled as “question or give up” was scarce and therefore less predictable by our models. In the overall model, F measure for firm choices was 0.83 with logistic regression and 0.82 with a decision tree. For tentative choice, F measure was 0.72 with logistic regression and 0.71 with decision tree.

Table 3.3 ranks individual wizards by their skill at the task, as in Table 3.1. For each wizard, it includes the accuracy of the individual decision tree model and corresponding F measure for firm and tentative choices. It also shows the number of

Table 3.3: Decision trees for individual wizard models.

Tree	Rank	Nodes	Features	Accuracy	F-firm	F-tentative
W4	1	55	12	75.67	0.85	0.62
W5	2	21	10	76.17	0.85	0.75
W1	3	7	8	80.44	0.87	0.77
W7	4	45	11	73.62	0.83	0.60
W3	5	33	10	77.42	0.84	0.75
W2	6	35	10	78.49	0.85	0.71
W6	7	23	10	85.19	0.80	0.89

features and nodes in each tree. (For most of our wizards, data-points labeled with a question or give up were too scarce to be predicted. We therefore omit F measures for these choices in Table 3.3.) Our models could predict when to ask a question only for W4 ($F = 0.75$), the wizard most skilled at detecting when a title was not among the candidates, and thereby the one who asked the most questions. Because our data is skewed toward firm and tentative choices, wizard skill at the task does not correlate well with wizard model accuracy (i.e., how well the model represents the wizard on the experiment data). To focus on W4's questioning behavior, we trained an additional decision tree model to learn how W4 chose between two actions: select a title or ask a question. The resulting decision tree with 37 nodes and 8 features had an accuracy of 86.5%, and an F measure of 0.91 for selecting a title and 0.68 for asking a question.

In summary, our title experiment results supported our investigation in three fundamental ways. First, we found that human wizards are skilled at disambiguating noisy ASR in the context of query results from voice search. Second, we were able to distinguish different wizard behaviors, and found that our best wizard could not only identify a title when it appeared among the query results, but could also recognize that the title was not among the query results and ask a thoughtful question instead. (This behavior would naturally extend to full dialogue and motivated our full dialogue experiment.) Third, we were able to train accurate models of wizard actions, and in particular, we were able to accurately model W4's questioning behavior, one that would be a desirable addition to a spoken dialogue system. Although we did not explore feature selection extensively here, our results demonstrated that there is much

redundancy among the many features available in a spoken dialogue system, and that automated feature selection should be used to train accurate models of wizard behavior.

3.2 Full dialogue experiment

Our full dialogue experiment extends the title experiment to a WOz study of full dialogue interactions like those with the CheckItOut baseline system (Ligorio, et al. 2010b). Full dialogue is more challenging than single title requests. A full dialogue in our application consists of sub-dialogues: patron login, book requests, and order summary. Book requests may be by title, author, or catalogue number. Moreover, full dialogue is subject to *disfluencies* (pauses, repetitions, self corrections, and false starts), and corrections (“That’s not what I said”). In full dialogue, wizards have the opportunity to ask questions whose answer will help them disambiguate noisy ASR.

The language model for this experiment was built on 3,000 randomly selected book titles with the same tools as for the title experiment. Although the corpus has not yet been transcribed, a sample of 315 transcribed utterances from 6 speakers with the same recognition settings suggests that the WER for this experiment was about 50%.

3.2.1 Experimental design

In this experiment, subjects made calls to the system by telephone, unaware of the wizards’ presence. The wizards interfaced with the dialogue system through a GUI described below. Ten callers (5 male and 5 female) each made 15 calls to each of 6

wizards (3 male and 3 female), for a total of 900 calls. Both wizards and callers were recruited by email and flyers to students at Hunter College, Columbia University, and New York University. All subjects received monetary compensation.

We trained four male and five female subjects as wizards in two phases. The first training phase served to familiarize the wizards with the task of disambiguating noisy ASR against candidate titles. Trainees were given a text file with 24 ASR strings from titles spoken in the title experiment and were asked to select which, if any, of the corresponding candidate titles matched the noisy ASR. In the second training phase, trainees were given a visual and verbal description of a new wizard GUI (described below), and they watched the trainer perform as wizard in a sample call. Each trainee then made five test calls during which she could ask questions and talk to the trainer. Six of the nine trainees were then chosen as wizards based on their motivation and skill at the task. Each caller also made five training calls, during which she could ask the trainer questions via chat. During the experiment, the trainer sat in the room with the wizard, and could also communicate with the caller via chat. This facilitated the complex wizard-caller scheduling and allowed the trainer to deal with unforeseen difficulties. On the rare occasion of a system crash, calls were discarded.

Before each call, callers accessed a web page to obtain a *scenario* with a patron identity (patron name, telephone number and address) and four books to order (title, author, and catalogue number). Books in a scenario were randomly selected from among the 3,000 titles used to build the language model. To collect data representative of the three types of requests, callers were instructed to request in each

call one book by title, one by author, one by catalogue number, and one by any method of their choice. They could do so in any order they liked. Calls with a wizard were structured as calls with CheckItOut. Callers would first identify themselves as patrons of the library by providing their area code and telephone number, and then proceeded to order four books. Callers also had the option of listening to a summary of the order at the end of the call.

The wizard module replaced the dialogue manager in CheckItOut, as illustrated in Figure 3.1. The wizard interacted with the system through two similarly organized GUIs, one for the patron login sub-dialogue, and one for the remainder of the call. The GUIs interfaced with the wizard module, which in turn forwarded and received frames from a hub just like any other CheckItOut module. On an incoming call, the

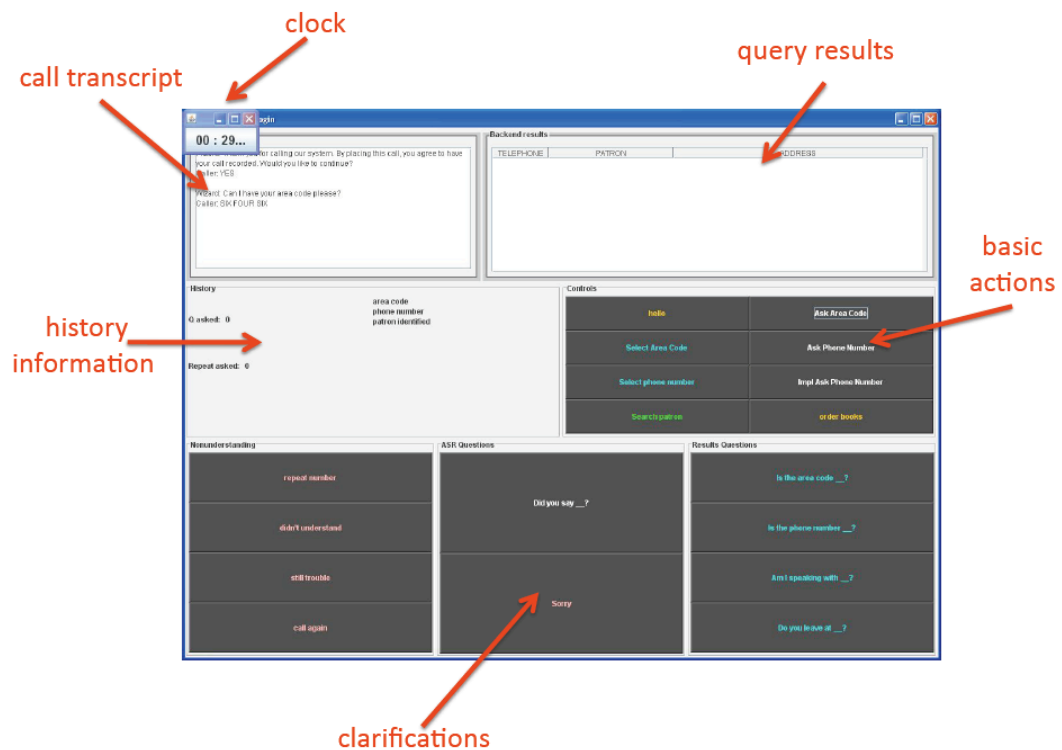


Figure 3.6: Annotated screenshot of the login GUI.

login GUI, depicted in Figure 3.6, appeared on the wizard screen. After the caller had been successfully identified as a patron of the library, the login GUI was replaced by the *book-request GUI*, depicted in Figure 3.7. The two GUIs had similar functionalities. Through the login GUI, the wizard could search the patron database with the ASR transcriptions of the patron's area code and telephone number, issue prompts to ask for clarifications if necessary, identify the patron, and initiate the first book request, which would, in turn, launch the book-request GUI.

Because the login GUI is essentially a simplified version of the book-request GUI, we describe in detail the latter only to avoid a repetitive discussion. In general, patron identification is much less error prone than book requests are. Area code and telephone number are well recognized due to the restricted vocabulary for numbers

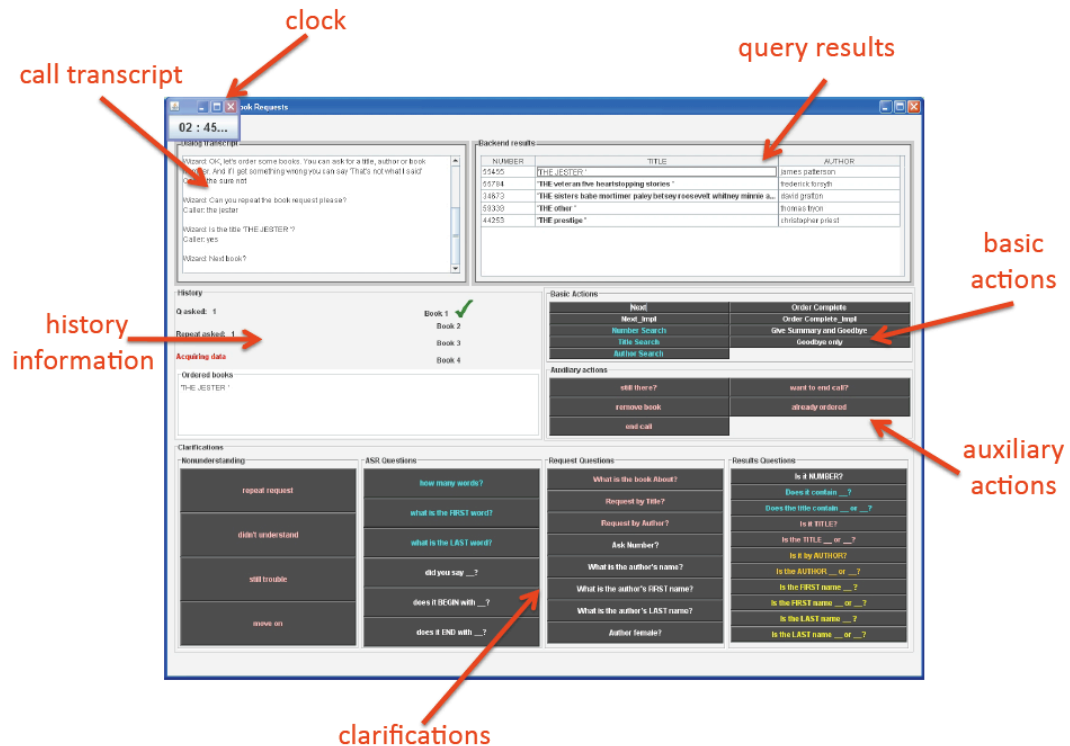


Figure 3.7: Annotated screenshot of the book-request GUI.

and the known number of digits. This research, therefore, focuses on the study of wizard behavior in the much more difficult task of ASR disambiguation for book requests.

As in the title experiment, the wizard had access to a live feed of the ASR transcription of the user utterances, and could search the database with the ASR. The top-left frame in Figure 3.7 was a scrollable live transcript of the call (sequences of ASR transcriptions of the caller's speech and wizard-issued prompts). The ASR string was matched against database entries in the same way it had been in the title experiment. In this experiment, however, the wizard selected one of several predefined prompts to respond to the caller. The top-right frame in Figure 3.7 displays the results of a database query. The center-left frame provides some history on the current call: the number of books ordered so far, the number of clarifications asked so far, the number of times the wizard signaled a non-understanding, and the titles of books that had been ordered.

On incoming ASR, the wizard could optionally make one or more database queries and then issue a prompt to move the dialogue forward. Prompts selected by the wizard were forwarded to the text-to-speech module and thereby back into the CheckItOut pipeline. Figure 3.8 shows a portion of a dialogue between a wizard and a caller. A full list of the available prompts appears in Appendix C.

Basic actions, in the center-right frame of Figure 3.7, include three database queries (one by catalogue number, one by title, and one by author), and predefined prompts to request the next book (with or without implicit confirmation of the book

- basic action: request the first book
1. **Wizard:** OK, let's order some books. You can ask for a title, author or book number. And if I get something wrong you can say 'That's not what I said'
 2. **Caller:** .GLORY. GILBERT
- clarification: question about the book request
3. **Wizard:** Did you ask for an author?
 4. **Caller:** YES
- clarification: question about the ASR
5. **Wizard:** Did you speak the word GILBERT?
 6. **Caller:** YES
- clarification: elicit specific information
7. **Wizard:** What is the author's first name?
 8. **Caller:** LARRY
- clarification: question about the ASR
9. **Wizard:** Did you speak the word LARRY?
 10. **Caller:** YES
- clarification: elicit specific information
11. **Wizard:** What is the author's name?
 12. **Caller:** .LARRY. GELBART
- basic action: author search on ASR: .LARRY. GELBART
- clarification: question about the query results
13. **Wizard:** Is the title *laughing matters on writing mash tootsie oh god and a few other funny things*?
 14. **Caller:** YES
- click on candidate in query results frame:
offer a particular book by an author
15. **Wizard:** Would you like *laughing matters on writing mash tootsie oh god and a few other funny things* by LARRY GELBART
 16. **Caller:** YES
- click on candidate in query results frame: order the book
- basic action: request the next book
17. **Wizard:** Next book?

Figure 3.8: An excerpt from a dialogue between a wizard and a caller annotated by wizard action. Caller turns are ASR transcriptions of the caller utterances, and wizard prompts are generated by the system in response to a command from the wizard. Words delimited by periods denote unconfident ASR.

just ordered), prompts to inform the caller that the order is complete (also with or without implicit confirmation), and prompts to offer the caller an optional summary of the order, or say goodbye. For example, line 17 in Figure 3.8 is a prompt issued by a basic action that requests the next book from the caller. Recall that a book request here refers to a portion of the dialogue concerned with the identification of the next book to order. For example, lines 1 through 16 in Figure 3.8 together form a single book request. When the wizard selected a query action, a pop-up window containing all ASR in the current book request appeared on the wizard's screen. The wizard then selected from that window the ASR words with which to query the database.

Each database search for queries by title or catalogue number returned the five top-scoring candidate book matches ranked by R/O score. If the wizard could identify the correct book from among the candidates, she clicked on it in the database-results frame (top-right in Figure 3.7), and the book was added to the order. After ordering a book, the wizard selected a basic action to request the next book. For queries by author, the search returned at most three of the most popular books for each of the five top-scoring author candidates, also ranked by R/O score. To issue a prompt that would offer a particular title by an author, the wizard clicked on a book in the database-results frame. If the caller's response was positive, clicking on that book a second time ordered it. For example, the prompt in line 15 in Figure 3.8 was issued by clicking on a book in the query results frame after a query by author. After the confirmation in line 16, the wizard clicked on that book again to order it.

If the wizard could not identify a book from among the candidates returned by a query, or if she thought the ASR was too noisy to search from, she could ask a

clarification to advance the dialogue. *Clarifications* were of four types (as indicated by their grouping into four frames at the bottom of Figure 3.7):

- Four clarifications issued prompts that signaled non-understanding: ask the caller to repeat, inform the caller she was not understood, signal continued difficulty, or *move on*. A “move on” told the caller that the system was having too much difficulty with the current book request, and asked the caller to request the next book on the scenario, and come back to this one later.
- Six clarifications asked questions about the ASR (e.g., “Did you say _?”). When the wizard selected a prompt that required the selection of some ASR word(s), such as “Did you say _?”, a pop-up window with all ASR in the current book request appeared on the GUI and the wizard selected words from it.
- Eight clarifications asked general questions about the book request (e.g., “Did you ask for a book title?”), or elicited a change in request type (e.g., “May I have the book number?”).
- Eleven clarifications asked questions about the query results (e.g., “Is the title _?”). Here again, a pop-up window would appear on the wizard screen from which she could select database candidates (title, author, number, or words) to ask about.

Five *auxiliary actions*, in the center-right frame of Figure 3.7, provided prompts to ask the caller whether she was still there, inform the caller that the book just requested had already been ordered, confirm whether the caller had requested to end the call, remove a book that had mistakenly been ordered, or prematurely end the call.

The response time (the time between the user's utterance and the next prompt) in a wizard experiment is necessarily longer than it would be with a fully automated system. Although wizards were instructed to issue the next prompt in a reasonable amount of time, they were allowed to make more than one database query if necessary, deliberate over ASR and book candidates, and choose what to say next. This process required thought, and wizards often pondered. Callers were also instructed that they were calling a research dialogue system that needed some time to consider complex alternatives. Although wizards were instructed to issue the next prompt in a reasonable amount of time, they were not told how long to persist in the identification of any individual book. Each wizard could use the move-on action freely, whenever and how often she thought it appropriate. To avoid exhausting our participants with overly lengthy calls, however, the clock in the upper-left corner of Figure 3.7 enlarged and turned bright yellow after six minutes on the call. Wizards were instructed to complete the current book request at that point as quickly as possible, and then use the auxiliary action to prematurely end the call, even if all four books had not yet been ordered.

Wizards were surveyed immediately before calls 1, 60, and 120. The first survey collected demographic information. The second and third surveys allowed wizards to report on the ease or difficulty of the task, and elicited strategy information. Callers were surveyed after calls 15, 30, 60, and 90. All caller surveys were the same; they elicited user satisfaction measures and allowed the callers to make comments.

3.2.2 The corpus: a wizard analysis

We collected 913 dialogues (7 of the 10 callers made a few extra calls), which comprised 3,394 book requests, and 17,288 adjacency pairs (Ligorio, et al. 2010a, Ligorio, et al. 2010b). Because of the time delay between system prompts, some adjacency pairs include multiple caller utterances, 20,378 in all. In the remainder of this discussion we report averaged data in the form μ (range, σ). Each call averaged 22.36 (4 – 40, 5.06) caller utterances, with 2.99 (1 – 10, 2.27) words per utterance. Book titles can indeed be long. The average title in the scenarios was 5.96 (1 – 34, 4.38) words. Lengthy titles are likely to contain a *subtitle* (an extra phrase that follows the title and is separated by a colon). In the random sample of titles used to generate the scenarios, 39% contained subtitles. Callers were given no instructions on how to make a book request, and therefore liberally chose whether or not to speak a subtitle. Before issuing the next prompt, the wizard was free to make zero or more database queries. In all, 32% of adjacency pairs contained 1.09 (1 – 6, 0.33) database queries.

The task was not an easy one. Queries by catalogue number returned the correct book 58% of the time, but were the least frequent. Author and title queries returned the correct book 33% and 28% of the time respectively. Even at this harder task of full dialogue, however, wizards could successfully identify the requested books. Of all ordered books, 92% (2.26 of the 2.45 per call) were correct. During the entire experiment, only 22 books were ever countermanded once ordered (in response to “that’s not what I said”). Because of the 6-minute time limit, wizards terminated 63% of the calls before the caller had ordered all 4 books. Of the complete calls, 76% were

fully successful, that is, identified all 4 scenario books correctly. A book could still be ordered incorrectly only if the wizard did not recognize a “that’s not what I said” response.

Table 3.4 compares wizard performance. WA and WB were the most successful wizards; they made the most fully successful calls, and had the fewest *failed calls* (in which no books were ordered). By the same measures, WD and WE were the least successful wizards. When the ASR recognition was poor, the most effective strategy for understanding what the caller said was to use the context provided by the database

Table 3.4: Comparative wizard behavior and performance. WA and WB were the most successful wizards with most fully successful calls and fewest failed calls.

Question type	Overall	WA	WB	WC	WD	WE	WF
Signal non-understanding	37%	34%	42%	55%	40%	33%	22%
About the ASR string	8%	2%	0%	5%	12%	15%	6%
About the query results	36%	36%	40%	33%	26%	32%	5%
About the book requested	19%	27%	18%	7%	22%	20%	21%
Total	1156	2321	1549	1982	2013	1868	1838
Query type							
Catalogue number	26%	30%	26%	24%	25%	26%	25%
Author	31%	29%	28%	30%	35%	30%	34%
Title	43%	41%	46%	46%	40%	44%	41%
Actions per call							
Questions before any search	0.35	0.29	0.21	0.20	0.67	0.42	0.36
Explicit confirmations	6.07	6.76	4.18	6.13	5.32	6.56	7.02
Implicit confirmations	0.40	0.62	0.68	0.01	0.20	0.86	0.00
All confirmations	6.47	7.38	4.86	6.14	5.5	7.45	7.02
Move-on strategy	0.67	0.39	1.19	0.56	0.65	0.43	0.80
Actions per book request							
Database searches	1.73	2.10	1.72	1.73	1.70	1.70	1.67
Questions	3.41	4.09	2.28	3.53	3.68	3.90	3.28
Confirmations	1.74	2.05	1.10	1.66	1.56	2.37	1.93
Total	6.92	8.24	5.10	6.92	6.94	7.97	6.88
Call statistics							
Fully successful calls	28%	33%	31%	33%	24%	16%	32%
Failed calls	17%	7%	12%	25%	16%	24%	16%
Requests per call	3.72	3.64	4.44	3.75	3.57	3.19	3.69
Correct titles	2.26	2.69	2.53	2.19	2.07	1.89	2.28

query. Recall our example of the ASR string SOONER SHEEP MOST DIE and its natural identification in the context of the three candidate book-title matches:

Soon She Must Die

Why Someone Had To Die

The Messenger Must Die

Our best wizards distinguished themselves from our poorest wizards in the use of such context. They typically asked clarification questions to disambiguate search candidates, and explicitly offered the title, author, or catalogue number they guessed to be the correct one among the candidates (e.g., “Is the author _?”). Our poorest wizards, in contrast, focused more on the ASR alone. They asked the most clarifications about the ASR (5.23 and 4.09 questions per call respectively), and they asked the most questions before making any query. WA and WB, on the other hand, almost never asked about the ASR alone.

WA’s distinguishing characteristic was her persistence in the identification of any single book. She asked the most questions per book request. She also made the most database queries, and requested the second-most confirmations per book request. She rarely abandoned a book request by the use of a move on action, and used a broad variety of clarifications. She also developed a distinctive strategy to shift the book request type when she experienced difficulty, or to confirm uncertain information. She asked most often what the book was about and about the author’s name, and requested the catalogue number three times as often as any other wizard. For example, in Figure 3.9, after an author query on John Egerton, WA led the caller to shift query type from author to catalogue number, and was able to successfully

1. **Wizard:** Next book?
2. **Caller:** JOHN EGERTON
3. **Wizard:** Can I have the RC number?
4. **Caller:** FOUR
5. **Caller:** .ROHAN. FIVE FOUR SIX
6. **Wizard:** Is the title *Speak now Against the Day: The Generation Before the Civil Rights Movement in the South?*
7. **Caller:** YES
8. **Wizard:** Would you like *Speak now Against the Day: The Generation Before the Civil Rights Movement in the South* by John Egerton?
9. **Caller:** YES

Figure 3.9: WA leads the caller to switch book request type and successfully identifies the correct book from among the result candidates in spite of ASR error.

1. **Wizard:** Next book?
2. **Caller:** .PAGAN. .GEORGE. AMERICA
3. **Wizard:** Did you ask for a title?
4. **Caller:** YES
5. **Wizard:** Can you repeat the book request please?
6. **Caller:** .PAGAN. .TURNER. DIABETES
7. **Caller:** DIDN'T I AM DONE
8. **Wizard:** Did you speak the word DIABETES?
9. **Caller:** YES
10. **Wizard:** Is the title *Conquering Diabetes: A Cutting Edge Comprehensive Program for Prevention and Treatment?*
11. **Caller:** NO
12. **Wizard:** What is the author's name?
13. **Caller:** .STARDANCE. SERVICES
14. **Wizard:** Is the title *Controle Su Diabetes Guia Para El Ciudadano De Su Salud: Take Charge Of Your Diabetes A Guide For Care?*
15. **Caller:** YES

Figure 3.10: WA successfully identifies the correct book from among the candidates with only two very noisy ASR transcriptions of the title and author name.

identify the correct book among the multiple results for that author, despite ASR error in the transcription of a digit in the five-digit catalogue number.

WA's successful questioning behavior is further illustrated in Figure 3.10. In lines 5 and 6 of Figure 3.10, the caller's request for the book *Controles Su Diabetes Guia Para El Ciudadano De Su Salud: Take Charge Of Your Diabetes A Guide For Care* is transcribed by the speech recognizer as two distinct utterances. The ASR suggests that the caller chose to speak the English subtitle only. Words delimited by periods indicate unconfident words. Upon confirmation that the caller had said DIABETES, WA queried the database with the ASR in line 6. The query returned these candidates ranked by title R/O score:

- *Conquering Diabetes: A Cutting Edge Comprehensive Program for Prevention and Treatment* by Anne L Peters
- *Coping With Diabetes* by Pat Kelly
- *Controle Su Diabetes Guia Para El Ciudadano De Su Salud: Take Charge Of Your Diabetes A Guide For Care* by US Dept of Health and Human Services
- *I'm Tougher than Diabetes* by Alden R Carter
- *The Other Diabetes Living and Eating Well With Type 2 Diabetes* by Elizabeth Hiser

Because all five titles contained the only confident and confirmed word, DIABETES, WA requested the author's name to further disambiguate. Although the ASR .STARDANCE. SERVICES would have been unintelligible as an author otherwise, in the context of the query results WA successfully identified the book with only two very noisy transcriptions of a title and author name.

WB, the other top wizard, was primarily goal-oriented. He focused on the overall task of identifying as many books as possible within the six-minute time limit. His

Table 3.5: Comparative caller performance. C1 was the most well recognized caller.

	All	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9
Correct books per call	2.26	1.01	3.26	1.03	2.21	2.38	2.70	2.29	2.76	2.49	2.68
Elapsed time per call	345.5	353.6	270.3	346.7	355.5	335.2	335.9	368.2	344.4	348.5	325.7
Fully successful calls	28%	3%	63%	5%	13%	24%	36%	23%	43%	28%	44%
Failed calls	17%	38%	7%	48%	10%	10%	10%	10%	8%	11%	14%
Wizard found caller's book on query by											
Title	28%	12%	42%	11%	28%	38%	31%	30%	35%	30%	30%
Author	33%	20%	55%	18%	33%	30%	40%	30%	35%	30%	30%
Catalogue number	58%	35%	77%	44%	46%	64%	58%	54%	74%	66%	44%
Wizard searched based on caller utterances by											
Title	43%	44%	32%	47%	45%	42%	43%	52%	38%	42%	41%
Author	31%	37%	27%	25%	33%	28%	28%	27%	34%	31%	30%
Catalogue number	26%	19%	41%	18%	22%	30%	29%	21%	28%	27%	29%
Searches per request	1.77	2.05	1.45	2.01	1.85	1.81	1.77	1.91	1.51	1.68	1.78
Utterances per call	22.36	23.97	19.29	21.99	23.54	23.31	21.86	20.74	20.12	22.09	22.02
Words per utterance	2.99	2.98	2.82	3.11	2.97	3.09	2.87	3.42	2.72	2.81	3.16

distinctive characteristic was his extensive use of the move-on strategy. If any book request appeared to be problematic, WB would move on to the next book. WB asked the fewest questions, searched about as often as other wizards (except WA), and confirmed the least often. By the liberal use of move on, however, he processed the highest number of book requests, in part because callers were forced to return to earlier requests.

Our caller population presented our wizards with a variety of challenges. Table 3.5 compares our callers. C1 was the caller with best-recognized speech. Requests

from C1 were correctly recognized most often, and calls with C1 were the shortest. Database queries on ASR from C1's requests returned the correct book most often, and therefore wizards made the fewest queries per request with C1. In contrast speech from C0 and C2 was the worst-recognized, and they had the most failed calls, the fewest fully successful calls, and the fewest correctly identified books per call. (There was no difference between the two on a paired *t*-test.) Wizards also made more queries during calls with C0 and C2.

In summary, our full dialogue experiment produced a rich corpus that supported this research and will be released for use to the research community. Although both of our most successful wizards focused on the context of database queries to disambiguate noisy ASR, they developed distinct strategies. There was no correlation between accuracy and either confirmation frequency or use of the move-on strategy. Two callers, however, indicated in the surveys that a complete lack of confirmation, as in calls with WB, confused them. They had interpreted the request for the next book without any confirmation as a move on, and were then surprised to learn from the summary at the end of the call that their books had indeed been correctly ordered. The next section discusses the system features we collected to train data-driven models of wizards' error detection and recovery.

3.2.3 System features

Many descriptive features are available to a spoken dialogue system at runtime. Here we discuss the system features identified and collected during our full dialogue experiment, features from which we trained models of wizard error detection and recovery, as described in Chapter 4. Recall that in the title experiment of Section 3.1 we collected features that described a title request so that we could learn models to predict wizards' responses. To extend that idea here to full dialogue, we defined a training instance for supervised learning of wizard actions to be a feature vector that describes an adjacency pair, with a label that is the wizards' next action (i.e., the action that will initiate the next adjacency pair). Because the login subdialogue is less challenging to speech recognition than book requests are, we focus on the prediction of wizards' actions with respect to the book-request portion of the dialogue, and do not use adjacency pairs from the login subdialogue for model training.

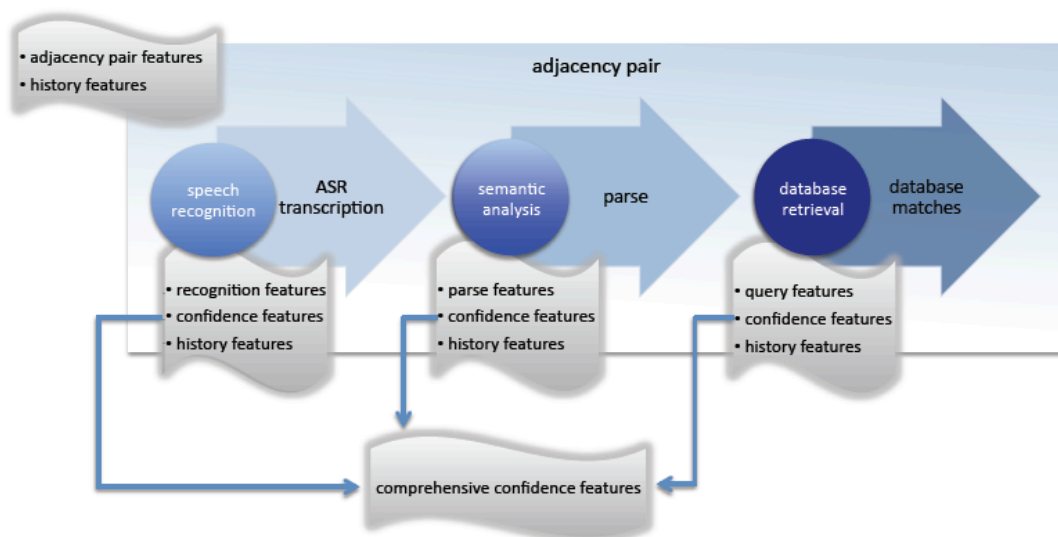


Figure 3.11: Levels of SLU processing (encompassed within an adjacency pair) and their features. (Reproduced from Figure 1.1)

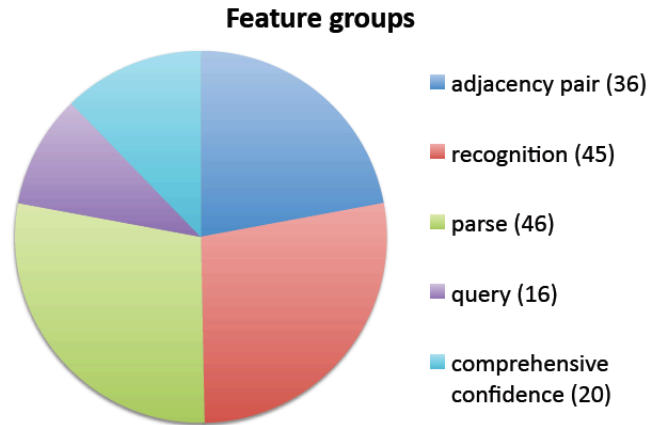


Figure 3.12: Number of features in each group, given 163 features.

Recall from our discussion in Section 1.1 that an adjacency pair is a portion of the dialogue that begins with a system prompt, ends just before the next system prompt, and encompasses the SLU of a user utterance. As previously discussed in Section 1.1 and repeated here as Figure 3.11, features are produced at different processing stages in the SLU pipeline. We manually identified and collected 163 features to comprehensively describe an adjacency pair and its SLU process. The three levels in the SLU pipeline in Figure 3.11 produce recognition, parse, and query features, along with their corresponding confidence and history features. Some confidence features from different SLU stages may be combined to obtain comprehensive confidence features. Adjacency-pair features predominantly describe the system prompt that initiated this adjacency pair and other possible wizard actions within the adjacency pair (e.g., select a book to order). Figure 3.12 partitions the dialogue-system features into these five *feature groups* and reports the number of features identified in each group.

It is, of course, possible to vary the granularity of such a partition. Because input and parse features both refer to the interpretation of the input signal, they could have been grouped together. Features for comprehensive confidence can be categorized as either input features or query features. Figure 3.13 illustrates the number of features in the resulting *feature meta-groups*. Furthermore, because history features capture the dialogue thus far and refer to a portion of the dialogue larger than a single adjacency pair, these could be considered as separate groups. Figure 3.14 illustrates the resulting *feature sub-groups*. By *feature category* we refer to some partition of the features set (e.g., groups, meta-groups, sub-groups). Figure 3.15 depicts a training instance that describes an adjacency pair with features from each of the five feature groups, and labeled with the wizards' next action, which will initiate the next adjacency pair. Figure 3.16 illustrates the hierarchical relationship among feature sub-groups, groups and meta-groups.

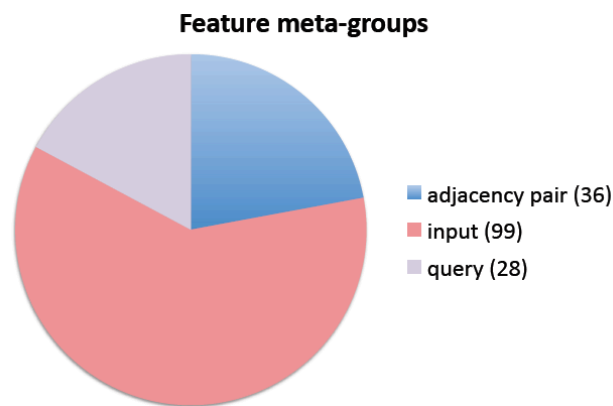


Figure 3.13: Number of features in each meta-group, given 163 features.

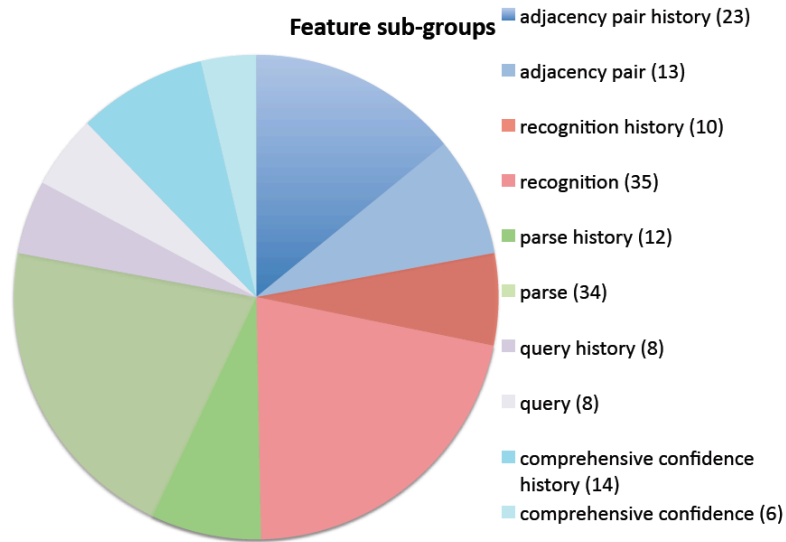


Figure 3.14: Number of features in each sub-group, given 163 features.

After the patron login, a dialogue in CheckItOut can be subdivided into four sub-goals, where each sub-goal consists of the identification of the next book to order. Each sub-goal is achieved through a book request subdialogue. History features can pertain either to the dialogue so far (e.g., the average acoustic model score in this call) or to the book request so far (e.g., the average acoustic model score in this book request). Before model training, all feature values are converted to numeric values and normalized.

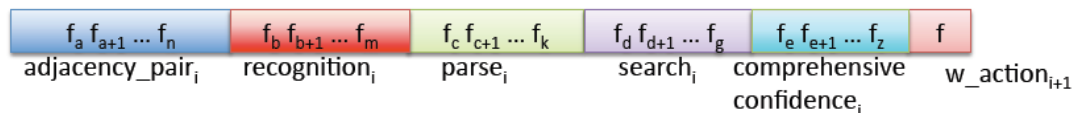


Figure 3.15: A training instance describes an adjacency pair as a vector of features from the five groups identified in Figure 3.12. The label on the instance (w_action_{i+1}) represents the next wizard action that will initiate the next adjacency pair.

Tables 3.6 through 3.15 describe the features in each sub-group. An asterisk indicates confidence features within each sub-group. Here we discuss only those features that require some explanation beyond the description in the table. Recall that

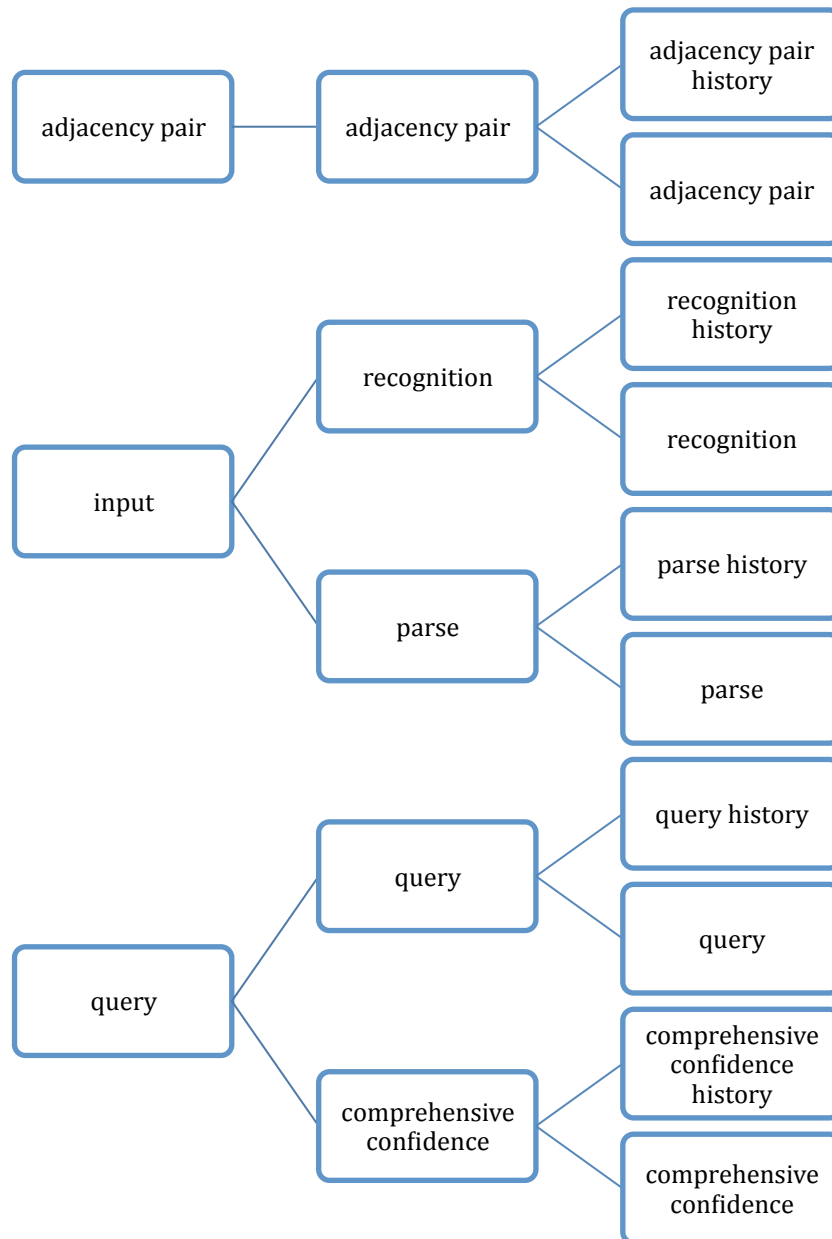


Figure 3.16: A hierarchical illustration of feature meta-groups (leftmost), groups (center) and sub-groups (rightmost).

Table 3.6: Adjacency-pair history features.

Adjacency-pair history features
1. Number of adjacency pairs in this call
2. Number of questions asked in this call
3. Number of clarifications (questions, explicit confirmations and non-understandings) in this call
4. Number of non-understandings in this call
5. Number of explicit confirmations in this call
6. Number of partial explicit confirmations in this call
7. Number of implicit confirmations in this call
8. Number of confirmations in this call that lead to a book identification
9. Number of suggested books by author in this call
10. Number of books ordered so far
11. Number of books removed so far (by use of "That's not what I said")
12. Number of questions in this call that were asked before making any database query
13. Number of move ons in this call
14. Number of book requests so far
15. Number of adjacency pairs in this book request
16. Number of questions asked in this book request
17. Number of clarifications in this book request
18. Number of non-understandings in this book request
19. Number of explicit confirmations in this book request
20. Number of partial explicit confirmations in this book request
21. Number of suggested books by author in this book request
22. Number of questions in this book request that were asked before making any database query
23. Number of move ons in this book request

after a number of consecutive non-understandings, the system or a wizard may decide it has had too much difficulty with the current request and may prompt the user to ask for another book and return to this one later on. This is referred to, in CheckItOut, as a “move on”, and is captured by features 13 and 23 in Table 3.6. A partial explicit confirmation, captured by feature 6 in Table 3.6, is an explicit confirmation of one or more words, as opposed to a full concept. For example, “Did you say Egerton?” is a partial explicit confirmation, as opposed to “Did you say John Egerton?” which explicitly confirms a value for the full author concept. In a goal-directed dialogue

Table 3.7: Adjacency pair features.

Adjacency pair features
1. Number of database searches in this adjacency pair
2. Number of user utterances in this adjacency pair
3. Whether a title was selected in this adjacency pair
4. Prompt that initiated this adjacency pair
5. Class of the prompt that initiated this adjacency pair
6. Whether the prompt that initiated this adjacency pair indicates a non-understanding
7. Whether the prompt that initiated this adjacency pair is an explicit confirmation
8. Whether the prompt that initiated this adjacency pair is a partial explicit confirmation
9. Whether the prompt that initiated this adjacency pair is an implicit confirmation
10. Whether a title by author was suggested in this adjacency pair
11. Index of the title by author that was suggested
12. Index of the title selected by the wizard in this adjacency pair
13. Whether a new book request was initiated

system, some system actions correspond to system prompts that elicit a response from the caller. As previously discussed in Section 3.2, these prompts were categorized for our wizards on the GUI. The prompt categories were basic actions, auxiliary actions, and four kinds of clarifications: signal non-understanding, ask about the ASR, ask about the book request, and ask about the query results. This classification is captured by feature 5 in Table 3.7.

In Tables 3.8 and 3.9, decoder refers to PocketSphinx, the speech recognition engine used in CheckItOut. (A detailed description of our speech recognition module appears in Section 1.3.) Feature 11 in Table 3.9 computes the Z-value of the language model score as $\frac{x - \mu}{\sigma}$, where μ is the mean of the language model score and σ its standard deviation. As previously mentioned, a speech recognizer may produce more

Table 3.8: Recognition history features. * denotes confidence features.

Recognition history features
1. Number of user utterances in this call
2. Average acoustic model score in this call
3. Average decoder score in this call
4. Average language model score in this call
5. Average word confidence in this call *
6. Number of user utterances in this book request
7. Average acoustic model score in this book request
8. Average decoder score in this book request
9. Average language model score in this book request
10. Average word confidence in this book request *

Table 3.9: Recognition features. * denotes confidence features. _1 denotes features that refer to the first recognition hypothesis; each has a dual, feature _2, for the second recognition hypothesis.

Recognition features
1. Number of hypotheses (at most two) generated by the recognizer
2. Number of words _1
3. Acoustic model score _1
4. Normalized acoustic model score _1
5. Maximum word confidence * _1
6. Minimum word confidence * _1
7. Average word confidence * _1
8. Decoder score _1
9. Decoder did not produce a hypothesis _1
10. Language model score _1
11. Normalized Z-value of the language model score _1
12. Whether the language model is greater than the mean so far _1
13. Speech rate measured as number of frames per word _1
14. Speech rate measured as number of frames per phonemes _1
15. Speech rate measured as number of frames per syllable _1
16. Number of 10ms frames in the utterance _1
17. Number of unconfident words * _1
18. Ratio of unconfident words to total number of words * _1
19. – 35. Dual features _2

than one recognition hypothesis. In CheckItOut, at most two recognition hypotheses are generated, represented by feature 1 in Table 3.9. Every _1 feature has a dual _2

Table 3.10: Parse history features. * denotes confidence features. $_1$ denotes features that refer to the first recognition hypothesis; each has a dual, feature $_2$, for the second recognition hypothesis.

Parse history features
1. Average number of gaps in the parse thus far in the call $_1$
2. Average number of words not covered by the parse thus far in the call * $_1$
3. Number of user utterances that could not be parsed in this call
4. Most frequent top slot in the parse tree in this call
5. Average number of grammar slots in this call
6. Average number of MICA slots in this call (See Section 1.3)
7. Number of user utterances that could not be parsed in this book request
8. Most frequent top slot in the parse tree in this book request
9. Average number of grammar slots in this book request
10. Average number of MICA slots in this book request
11. – 12. Dual features $_2$

Table 3.11: Parse features. * denotes confidence features. $_1$ denotes features that refer to the first recognition hypothesis; each has a dual, feature $_2$, for the second recognition hypothesis.

Parse features
1. Number of parses for this recognition hypothesis $_1$
2. Number of parses for this user utterance $_1$
3. Whether there was a parse for this user utterance $_1$
4. Average number of words per grammar slot in the best parse $_1$
5. Number of repeated grammar slots with respect to the previous user utterance in the best parse $_1$
6. Number of grammar slots in the best parse $_1$
7. Top grammar slot in the best parse tree $_1$
8. Number of words not covered by the best parse * $_1$
9. Ratio of words not covered by the best parse to total number of words * $_1$
10. Number of words covered by the best parse * $_1$
11. Number of MICA slots in the best parse $_1$
12. Average number of MICA slots per grammar slot in the best parse $_1$
13. Whether there was at least one author grammar slot in the best parse $_1$
14. Whether there was at least one title slot in the best parse $_1$
15. Whether there was at least one number slot in the best parse $_1$
16. Whether there was an interpretation for this user utterance $_1$
17. Whether there was no interpretation for this user utterance $_1$
18. – 35. Dual features $_2$

feature that refers to the second recognition hypothesis. If there is no second hypothesis, $_2$ features are set to a default value.

In Tables 3.10 and 3.11, features with a $_1$ also have a $_2$ dual feature for the second recognition hypothesis. Parse features that refer to a unique parse, refer to the parse selected for consideration by the Helios confidence annotator. (See the detailed discussion of semantic analysis in Section 1.3.) In Table 3.12, for adjacency pairs where there was no database query, all query-history features and query features are set to default values (0 for features whose values are always greater than 0, and -1 for features whose values can be 0.)

Table 3.12: Query history features.

Query history features
1. Total number of queries in this call
2. Number of catalogue number queries in this call
3. Number of title queries in this call
4. Number of author queries in this call
5. Total number of queries in this book request
6. Number of catalogue number queries in this book request
7. Number of title queries in this book request
8. Number of author queries in this book request

Table 3.13: Query features. * denotes confidence features.

Query Features
1. Query type (catalogue number, title, or author)
2. Number of words in the query selection
3. Number of query results
4. Average number of words in the query results
5. Maximum R/O score *
6. Minimum R/O score *
7. Average R/O score *
8. R/O score standard deviation *

Table 3.14: Comprehensive-confidence history features. $_1$ denotes features that refer to the first recognition hypothesis; each has a dual, feature $_2$, for the second recognition hypothesis.

Comprehensive-confidence history features
1. Average Helios confidence in this call $_1$
2. Whether the Helios confidence score so far is greater than the mean $_1$
3. Average combined confidence A in this call $_1$
4. Average combined confidence B in this call $_1$
5. Average Helios confidence in this book request $_1$
6. Average combined confidence A in this book request $_1$
7. Average combined confidence B in this book request $_1$
8. – 14. Dual features $_2$

Table 3.15: Comprehensive confidence features. $_1$ denotes features that refer to the first recognition hypothesis; each has a dual, feature $_2$, for the second recognition hypothesis.

Comprehensive confidence features
1. Helios confidence $_1$
2. Combined confidence A (Helios confidence + average R/O score + R/O score standard deviation) /3 $_1$
3. Combined confidence B (Helios confidence \times average R/O score \times R/O score standard deviation) $_1$
4. – 6. Dual features $_2$

In Tables 3.14 and 3.15, Helios is the confidence annotator used in CheckItOut. Helios selects the parse with the highest confidence score (feature 1 in Table 3.15). Section 1.3 provides details on the computation of confidence. Helios confidence is further combined with R/O score measures to produce two additional comprehensive confidence features for each recognition hypothesis (features 2 and 3 in Table 3.15). In adjacency pairs where there is no database query, these combined-confidence features have default values, as previously described.

Given the large number of features available at run time in a spoken dialogue system, and the available knowledge about their provenance along the SLU pipeline,

this research augments a general feature selection approach with this knowledge in Chapter 4. To experiment with feature categories at different levels of granularity, we experiment with feature groups, feature meta-groups, and feature sub-groups. We will discuss feature selection further in Chapter 4, and show that, while our novel general feature selection approach alone does well at removing redundancies, its knowledge-augmented version can often identify a better feature subset by sampling additional features from underrepresented feature categories.

Chapter 4

Feature selection for error detection and recovery in spoken dialogue systems

The data collected in the full dialogue experiment was used to implement data-driven models of wizards' error detection and recovery. These models were then implemented in the baseline system to improve it (as described later, in Chapter 5). Ideally, the features a learning algorithm (here, a *learner*) considers for model training are predictive, that is, relevant to the prediction outcome and useful to the learner. Given a set of input features, *feature selection* is intended to identify a subset of those features that are most predictive for a learning task (Guyon and Elisseeff 2003, Liu and Motoda 2008). Many domains, including error detection and recovery modeling in spoken dialogue systems, are characterized by high dimensionality; in such cases feature selection is essential to obtain robust models.

Unlike much previous work that considered error detection and recovery as a one-step process, this research addresses it at three different decision points along the SLU pipeline. Although all 163 features identified in Section 3.2.3 are available to describe an adjacency pair, different subsets will be relevant to different decision points. This chapter discusses our exploration of feature selection to learn models of wizard error detection and recovery, and describes two novel feature selection algorithms, one general and one knowledge-driven. Section 4.1 discusses our identification of the three decision points for error detection and recovery, defines

training instances for each of them, and describes necessary data pre-processing. Section 4.2 discusses related work in feature selection. Section 4.3 discusses a novel experimental framework developed for this research; it facilitates experimentation with feature selection methods and comparison among them. Sections 4.4 and 4.5 introduce two novel algorithms for feature selection and report on experimental results with them.

4.1 Training instances and data pre-processing

In most pipelined spoken dialogue systems, error detection and recovery occur in a single step at the end of the pipeline. (Figure 4.1, reproduced here from Figure 1.1, illustrates once again the SLU pipeline.) In particular, most dialogue systems do not use voice search, which allow partial matches to database entries to be used for SLU. Therefore, the SLU ends with a parse, and error detection amounts to a decision on

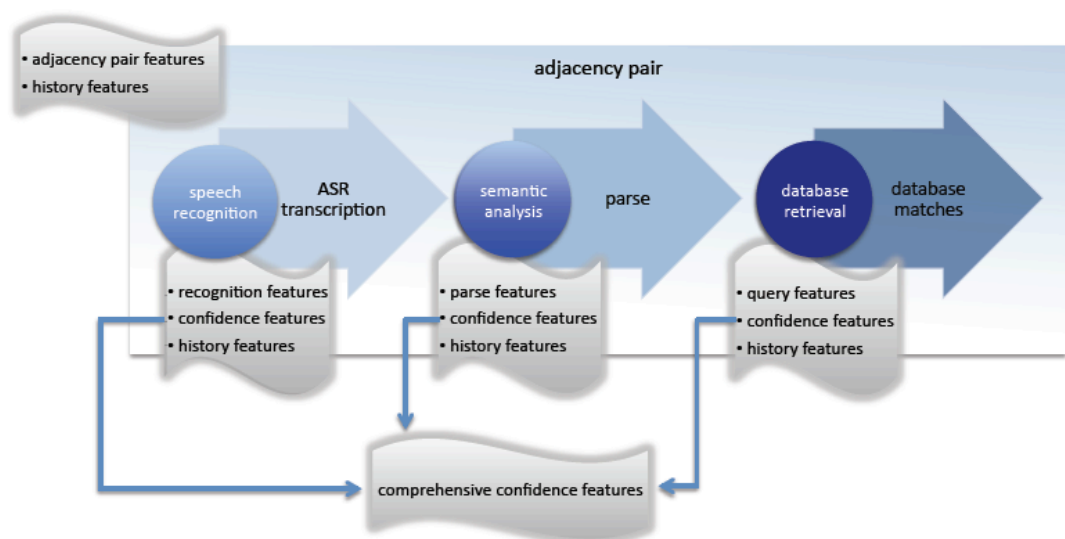


Figure 4.1: Levels of SLU processing (encompassed within an adjacency pair) and their features. (Reproduced from Figure 1.1)

Table 4.1: Wizards’ understanding of the ASR and database matches, with their corresponding actions. Darker shades of blue indicate greater progress along the SLU pipeline.

Wizard Confidence	Wizard Action
The ASR is too noisy and cannot be used	Signal non-understanding or move on
There is some useful information that can be clarified	Ask a question about the ASR or about the request
The ASR is good enough for voice search	Perform voice search
The query results don’t match the ASR well	Ask a question about the request or signal non-understanding
The query results are a good match	Ask a question about the query results

whether the parse is confident or not. In Olympus/RavenClaw systems such as the CheckItOut baseline, for example, the Helios confidence annotator determines whether a non-understanding occurred by setting a threshold on the confidence score assigned to the parse, as detailed in Section 1.3.

Our wizards, on the other hand, could make decisions at different points in the SLU pipeline through examination of the ASR transcription and voice search results. Wizards identified different degrees of signal in the noisy ASR and query results, and acted accordingly. Table 4.1 illustrates degrees of wizards’ understanding of the ASR and of the query results, with their corresponding actions. As indicated by the different shades of blue, these actions were in response to data obtained at different levels of SLU processing. As a result, three different models of error detection and recovery were trained for three distinct decision points along the SLU pipeline. Figure 4.2 illustrates our models’ decision points and their execution sequence in a tree diagram. Here again, darker shades of blue correspond to later stages in the SLU pipeline.

The three models in Figure 4.2 emulate wizard error detection and recovery. The *voice-search model* determines whether the system should perform voice search or not. The *no-query model* determines how to clarify the request, given that there has been no database query. The *yes-query model* determines clarification given that there has been a query. To focus on the more challenging aspects of dialogue in our application, only data from the book-request portion of dialogues in the full dialogue experiment was used to model wizard actions. (Adjacency pairs from the login subdialogue were discarded for model training.)

Training instances for each of the three models were defined as illustrated in Figure 4.3. The voice-search model in Figure 4.2 is trained on all the book-request data. A training instance for the voice-search model does not include query features

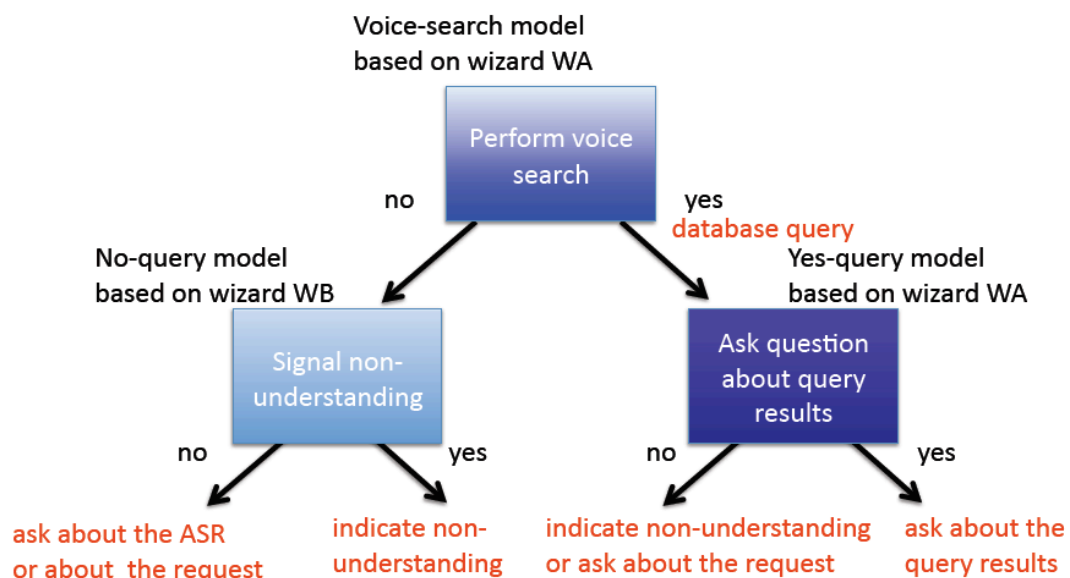
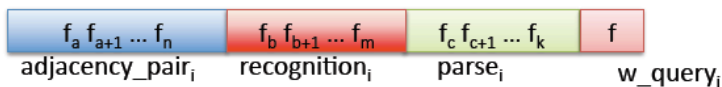


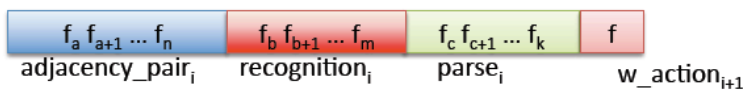
Figure 4.2: Models of wizards’ decision making. The voice-search model decides whether or not to make a database query. The no-query model decides the next action to take given that there has been no database query. The yes-query model decides the next action to take given that there has been a database query. Darker blue indicates later stages in the SLU pipeline. Text in red indicates wizard actions predicted by the models.

and comprehensive confidence features, because the model predicts whether the wizard performed a database query or not. The confidence feature retained was Helios'; it was grouped with the parse features. Similarly, a training instance for the no-query model does not include search and comprehensive confidence features, because the model predicts a wizard's action given that there has been no database query in the adjacency pair. The no-query model is trained only on data that describes adjacency pairs where there has been no database query. A training instance for the yes-query model, however, does include query features and comprehensive confidence features. The yes-query model is trained only on adjacency pairs that include a database query. The decomposition of error detection and recovery into three decision points also avoids the data sparsity problem that would be encountered if a model considered features from all groups (e.g., query features with default values for adjacency pairs where there had been no query).

Voice-search model



No-query model



Yes-query model

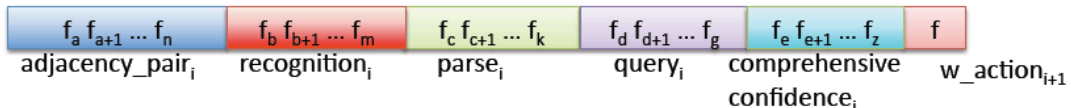


Figure 4.3: Training instances for the models in Figure 4.2. The label on the instance for the voice-search model is the wizard's choice to query or not in this adjacency pair. The label on instances for the no-query and yes-query models is the wizard's next action, the one that will initiate the next adjacency pair.

Recall that, in the full dialogue experiment, a wizard was permitted to make multiple database queries within a single adjacency pair. When a caller's utterance was long or contained pauses, often the interaction manager would split it into two or more utterances, each with its own recognition and parse features to describe it. Moreover, while the wizard made database queries and pondered over the results to select the next action, the recognizer would keep listening and would produce ASR for any incoming signal. Any incoming noise or speech was recognized as an additional utterance within that adjacency pair. This caused some adjacency pairs to contain multiple queries and/or multiple caller utterances, producing instances with recognition and parse features for each query and each utterance. Tables 4.2 and 4.3 indicate the distribution of database queries and caller utterances respectively over adjacency pairs for the book request portion of dialogues in the full dialogue experiment. In contrast, in a fully automated system, a single caller utterance is considered within each adjacency pair. To obtain training instances that conformed to those that would be available in a fully automated system, instances with a single database query and a single caller utterance were obtained as described next.

Table 4.2: Distribution of database queries over adjacency pairs in the book-request portion of dialogues from the full dialogue experiment.

# of database queries	# of adjacency pairs
1	5072
2	390
3	43
4	5
6	2
	Total: 5512

Table 4.3: Distribution of caller utterances over adjacency pairs in the book-request portion of dialogues from the full dialogue experiment.

# of caller utterances	# of adjacency pairs
1	13988
2	2066
3	459
4	97
5	42
6	21
7	3
8	2
9	4
10	2
12	4
16	1
	Total: 16689

Figure 4.4 illustrates some examples of multiple caller utterances occurring within the same adjacency pair. (Recall that unconfident words are surrounded by periods.) Noise was often recognizable to the wizards as an ASR string of unconfident words that were meaningless together, like those in the second example in Figure 4.4. Noise was often disguised as confident, but unlikely words, such as THE TESSERACT, FIREBREAK, and DIDN'T DONE. These words occurred often and at unlikely intervals.

Example 1:

1. THE TESSERACT
2. DIDN'T .DONE. .COULD. .JANET. .FOR. A LEW
3. .LAURIEN. WHY TRADITIONAL FAITH IS DYING
4. NEW FAITH IS BEING BORN
5. FIREBREAK

signal noise
noisy ASR

signal noise

Example 2:

1. .GRUEL. .WAR. .BOLD. FOUR
2. .OPEN. THAT SHOULD .BE. .BLESSED. .JUNTA.

Example 3:

1. COME TWILIGHT A NOVEL OF ECOTERRORISM
2. .LAURIEN. .STRANGER. .RUTH.

Example 4:

1. THE HAIRSTONS AN
2. TWO NINE FIVE TWO TWO
3. THREE TWO
4. MOON

switch in request type

Figure 4.4: Examples of multiple ASR strings in a single adjacency pair. Annotations in red indicate signal noise, noisy ASR and a switch in request type by the caller.

The wizards quickly identified them as the ASR's transcription of an incoming non-speech signal, and ignored them. In the first example of Figure 4.4, the wizard only considered lines 3 and 4 for voice search, because she recognized lines 1, 2, and 5 to be signal noise. The third example illustrates a case in which the wizard could easily distinguish between good ASR and noise, and only queried the database with line 1. Finally, the fourth example illustrates a case in which the caller switched request type at mid-sentence. The wizard recognized the switch, and queried the database only with line 2. (Most catalogue numbers are five digits long.)

Discarding the adjacency pairs with multiple utterances would have greatly reduced the dataset, and removed the most interesting training instances. Nonetheless, replication of an adjacency pair multiple times, where each replica contained features that described each utterance, would be misleading. For example, in an adjacency pair with the two utterances `.DIDN'T. .DONE.` and `NEW FAITH IS BEING BORN`, the wizard queried the database with the second utterance only, completely disregarded the first one, and correctly identified the title from among the returned candidates. In an adjacency pair with the ASR `.DIDN'T. .DONE.` alone, the wizard would probably have asked a clarification question instead, without making any database query. The introduction of a second adjacency pair with features that describe the first utterance alone, but still labeled with the action that the wizard took (i.e., query the database with the second utterance) would be misleading, and would introduce noise into the data.

Thus, to obtain instances with a single user utterance from adjacency pairs with multiple ones, the utterance attended to by the wizard, and therefore used in a training instance, was identified as follows:

- If the preceding wizard action asked for an explicit confirmation, keep the utterance that contains either a confirmation or rejection.
- If the adjacency pair contains a database query, identify the ASR the wizard searched on, and retain that utterance in the single training instance.
- If the database query spans multiple ASR strings, produce multiple instances, one for each utterance that was used to query the database.

- If there was no database query, but the wizard asked a question that required selection from the ASR, retain only the utterance(s) the wizard inquired about.
- If the wizard asked a question about the request or signaled non-understanding, then she was not attending to any of the ASR strings. Select an utterance at random for the training instance.

Following the above criteria, utterances were automatically labeled as having been attended to or not by the wizard, and training instances with a single caller utterance were obtained.

Although wizards could make multiple database queries, only the results of the last query were considered for clarification or to select a book. For example, if the wizard first queried for a title and then queried for an author, but upon inspection of the query results decided that the ASR matched the titles better, she then had to execute the title query a second time, so that she could review the candidates from which to select a book or about which to ask a question. The last query was, therefore, always retained in a training instance for adjacency pairs with multiple queries. Adjacency-pair features indicated the original number of utterances and the original number of queries in the adjacency pair, and that information was recorded in each instance. This process produced 16,956 instances that describe adjacency pairs with a single caller utterance and a single database query.

The 163 features discussed in Section 3.2.3 described each utterance. Many of these features are, however, likely to be redundant, and not all features are equally relevant to prediction in all three models. Feature selection can be applied to improve model accuracy. The next section discusses related work in feature selection.

4.2 Related work in feature selection

Formally, given an initial set of features F , a data set D , and prediction outcome C , feature selection seeks to identify some subset $S \subseteq F$ that is predictive for D during supervised learning. (In unsupervised learning, *projection pursuit* can achieve dimensionality reduction higher than second-order (Fodor 2002).) Ideally in feature selection, S supports the highest accuracy for D and yet it is small enough to reduce training costs and prevent overfitting. Two fundamental approaches to feature selection are filters and wrappers (Guyon, et al. 2006, Liu and Motoda 2008). A *filter* is an algorithm that selects features independently from the learner, based on some pre-specified *feature-relevance criterion* q . This criterion is a metric that assesses each feature's *predictiveness*, its intrinsic desirability for inclusion in the selected subset. Examples of feature-relevance criteria include correlation with the outcome and information gain (Guyon and Elisseeff 2003).

A *wrapper*, on the other hand, is an algorithm that uses the learner to evaluate the prediction quality of subsets of features (Kohavi and John 1997). A wrapper searches the power set $P(F)$ of the possible feature subsets. For $S \subseteq F$, the wrapper trains a learner L on the data set D with the reduced feature set S to produce $L(D,S)$, and seeks to identify the S that maximizes $A(L,D,S)$, the accuracy of $L(D,S)$. Traditionally, wrappers perform either forward selection or backward elimination. *Forward selection* begins with $S = \emptyset$ and adds features from F to S as long as $A(L,D,S)$

increases. *Backward elimination* begins with F and gradually removes features from S as long as $A(L,D,S)$ does not decrease.

Filters are often preferred to wrappers for their relative simplicity, ease of implementation and speed (Liu, et al. 2010). They often produce a ranking of the features that can be useful for inspection and provide insight into the data. Because a feature relevance criterion that is adequately predictive across data sets is difficult to construct, however, wrappers support more accurate learning than filters do on many learning problems (Kohavi and John 1997). Unfortunately, a wrapper's increased accuracy requires extensive search during training, and it is therefore significantly slower.

Some learners perform feature selection intrinsically, and are referred to as *embedded methods* for feature selection. These include pruning and sub-tree raising in some decision tree algorithms (e.g., C4.5 (Quinlan 1993)), multilayer perceptrons where the number of units in the first hidden layer is less than the number of inputs (Alpaydin 2004), and least absolute shrinkage and selection operator (*Lasso*) for linear classification models (i.e., L_1 regularized linear models (Tibshirani 1996)). Lasso minimizes the sum of the squared errors in linear classification through a constraint $\sum_i |\beta_i| \leq t$, where β_i is the weight of feature f_i , and the constraint poses a bound on the sum of the absolute values of the weights. The value of t determines the number of weights that will be set to zero, effectively removing features from the classification. Embedded methods limit the choice of the learner, they are difficult to re-implement, and they are not always readily available to the learning community.

4.2.1 Filter and wrapper approaches

Most filters differ in their definition of the feature-relevance criterion q . FOCUS selects the minimal subset sufficient to determine the label for all the instances in the training set (Almuallim and Dietterich 1991). Correlation-based feature selection (CFS) searches best-first to identify features that are individually highly correlated with the target function, but minimally correlated with one another (Hall 2000). CFS is effective and fast when the problem contains many redundant features or features that are highly correlated with the target. Recent work extended CFS to decouple the analysis of feature relevance from the analysis for feature redundancy with separate pre-processing steps (Yu and Liu 2004). Traditionally the redundancy of a feature with respect to a subset S was measured as the correlation of a feature with those already in S . Instead, Yu and Liu re-define redundancy; a feature is redundant with respect to S if the feature has a Markov blanket within S . $M \subseteq S$ is a *Markov blanket* of feature f_i if and only if the joint probability of the features in F , excluding M and f_i , and of the classification outcome C , given both f_i and M , is the same as that joint probability given M alone:

$$P(F - M - \{f_i\}, C \mid f_i, M) = P(F - M - \{f_i\}, C \mid M).$$

To find a good feature-relevance criterion q , some work on filters has sought features' causal relationship to the classification outcome (Bontempi and Meyer 2010). That algorithm, min-Interaction Max-Relevance (*mIMR*) identifies a predictive feature f_i to be one with high mutual information with the target C , and high negative

interaction information with other features f_j in S . The *mutual information* between feature f_i and target C is measured as:

$$I(f_i, C) = \sum_{k \in K} \sum_{c \in C} p(f_{ik}, c) \log\left(\frac{p(f_{ik}, c)}{p(f_{ik})p(c)}\right).$$

The *conditional mutual information* of features f_i and f_j with respect to target C is measured as:

$$I(f_i, f_j | C) = \int_{k \in K} \int_{z \in Z} \int_{c \in C} p(f_{ik}, f_{jz}, c) \log\left(\frac{p(f_{ik}, f_{jz} | c)}{p(f_{ij} | c)p(f_{jz} | c)}\right) df_{ik} df_{jz} dc.$$

The *interaction information* of f_i and f_j with respect to target C is measured as:

$$I(f_i, f_j, C) = I(f_i, C) - I(f_i, C | f_j).$$

Zhao and Liu, instead, determined q by applying spectral graph theory to the graph induced from pair-wise instance similarities (Zhao and Liu 2007). Given a set of pair-wise instance similarities P , a graph G can be constructed to represent it (Chappelle, et al. 2006). A feature is considered to be predictive if it is *consistent* with the graph structure, that is, if it assigns similar or equal values to instances that are close to each other in G .

Most wrappers differ in their exploration of the space of feature subsets. Because exhaustive search in $P(F)$ is prohibitive for large $|F|$, effective ways to explore $P(F)$ are of interest. Although wrappers were originally used with greedy search (John, et al. 1994), the same researchers also tested best-first search with backward elimination and with forward selection (Kohavi and John 1997). Others have used a greedy search that combined backward elimination with forward selection, either adding or removing variables, whichever achieved greater improvement (Caruana and Freitag 1994). Guyon has exploited support vector machines' weights to determine the

features to retain under backward elimination (Guyon, et al. 2002). While this work retained the structure of a wrapper, it did not retain the desirable independence of wrappers from a unique learning algorithm.

Wrapper methods may overfit the training data (Reunanen 2003). Because the extent of overfitting is often related to the depth of the search, early stopping has been explored to reduce it (Loughrey and Cunningham 2005). This approach used a double cross-validation process: an inner cross-validation was used to estimate the point at which overfitting occurred, while an outer cross-validation determined the generalization accuracy.

4.2.2 Randomization and weighting

New applications produce high-dimensional data sets that challenge traditional feature selection methods. Examples include data streams from the web and genomics microarrays. These data sets have too many features for intensive search in the space of possible subsets by a wrapper, and are too ill-understood to identify a reliable feature-relevance criterion for a filter. In response, feature selection methods intended for these data sets have become more involved, and are not readily available to the learning community (in comparison, say, to Weka). Randomized feature weighting (*RFW*), the novel feature selection algorithm contributed by this research and described in detail in Section 4.4, scales well to large data sets, yet it is general and simple enough so that it is easily implemented and can be broadly applied. *RFW* incorporates randomness in a wrapper to learn weights that identify predictive feature

subsets. Other feature selection methods have also benefited from randomization; their differences from RFW are discussed here.

Relief, for example, was a filter that calculated the relevance weight for features in a binary classification problem (Kira and Rendell 1992). Relief randomly sampled a training instance and updated the features' weight based on the difference between the sampled instance and the two instances most similar to it in both classes. It selected all features with feature relevance criterion q above a specified threshold. Relief's feature weighting differs fundamentally from our approach, because the weights that RFW learns do not measure a feature's individual relevance to a class label. Instead, they identify features that, when included in a group, train accurate learners. Moreover, Relief randomly samples training instances, while RFW randomly samples features.

Another feature selection method that uses randomness is *randomized variable elimination*, a backward-elimination wrapper. At each step, it estimates the optimal number of random features to remove from S , based on how many are still in the set and the estimated number of relevant features in F (Stracuzzi and Utgoff 2004). This method assumes that the number of relevant features is considerably smaller than the number of irrelevant features. Randomized variable elimination randomly samples features for elimination. In contrast, RFW randomly samples feature subsets so it can learn feature weights that will ultimately rank the features.

Wrapper-filter hybridization attempts a balance between filters and wrappers. It uses local search to modify the feature subset initially selected by a filter (Esseghir 2010). This approach uses random restarts to escape from local minima, but still

relies on the filter’s feature-relevance criterion to select the initial solution. In contrast, RFW heavily emphasizes initial exploration of the feature subsets by random sampling.

Jaiantilal and Grudic boost feature selection accuracy for L_1 regularized linear classification models through the random sampling of subsets of the data (Jaiantilal and Grudic 2010). They repeatedly obtain L_1 regularized linear models of subsets of the data, and count how often a feature is retained in these models. These counts are then used as initial weights to train a final L_1 regularized linear model on the full data set. In contrast, RFW randomly samples features rather than data, and then learns on the selected features. RFW’s feature weights are not counts; rather, they reflect the quality of the classifier in whose training each feature participates. Most importantly, RFW’s learned feature weights are used to rank the features, and RFW is a general wrapper that does not limit the choice of the learner.

Gaudel and Sebag cast feature selection as a one-player game relying on the Monte-Carlo tree search UTC (Upper Confidence Tree) (Gaudel and Sebag 2010). A node in the search tree is a feature to be added to the subset S . A path in the tree is a subset of features, and a solution is a subset that contains a stopping feature node f_s . A reward function is defined on the final states based on a k -Nearest Neighbor (k -NN) classifier. The search for a solution is divided into two phases, a *bandit phase* and a *random phase*. In the bandit phase, the next feature to be added is the one that maximizes the reward according to k -NN. In the random phase, features are uniformly sampled, until the stopping feature f_s is selected with probability $1 - q^d$, where d is the number of features currently in the subset, and $q \in [0,1]$ is a parameter.

Like RFW, this algorithm seeks a balance between exploration and exploitation. RFW, however, only uses one random phase to learn feature weights, while this method uses a hill-climbing approach in the first phase, and only after it uses a second random phase. In this method, the random exploration is only a refinement of the solution found through hill climbing. RFW, on the other hand, promotes exploration much further through a purely random approach, while it records how well the features perform. Only at the end does RFW use those weights to select the final subset of features, having allowed the features to acquire their weights unbiased by a pre-defined partial solution.

Random forests train a large number of decision trees on random subsets of features of a fixed size (Breiman 2001). These models are then used as an ensemble classifier whose final output is the mode of the class output of the individual models. This is also fundamentally different from using classifiers trained on random subsets to learn a ranking of the features and then train a final model on the top-ranked features.

During supervised learning for a mixture of expert agents, Petrovic and Epstein showed that learning weights on randomly-chosen subsets of experts fostered the emergence of a good mixture (Petrovic and Epstein 2009). In that work, randomly-selected subsets of experts voted to make decisions to solve a problem. If they solved it, their weights were adjusted. Each new subset of experts was selected independently of the learned weights. To avoid filters' requirement of a pre-specified feature-relevance criterion q while retaining its simplicity, and to avoid a wrappers' search-intensive computation and overfitting, RFW adapts and extends Petrovic and

Epstein's work for feature selection. RFW is discussed in detail in Section 4.4. The next section discusses XFF, a novel experimental framework developed for this research to compare feature selection methods.

4.3 XFF: an experimental framework for feature selection

To experiment with feature selection, this research designed and implemented XFF (eXperimental Framework for Feature selection), an experimental framework for feature selection methods that extends the Weka application programming interface (API) (Hall, et al. 2009). Weka is an open source collection of machine learning algorithms written in Java, and developed at the University of Waikato, New Zealand. The motivation behind XFF was threefold. First, the framework provides the ability to implement multiple feature-selection approaches, including approaches that are not publicly available or that are novel. In particular, we implemented feature selection that uses domain-specific knowledge about the provenance of features along the SLU. Second, the framework allows the combination of selection approaches, such as the combination of general and knowledge-based selection. Third, the framework supports experiments to compare feature selection approaches for predictive accuracy against any number of baselines. Figure 4.5 illustrates the framework.

On the right in Figure 4.5, XFF supports four kinds of feature selection. Two of these are filters: *general filters* and *heuristic filters*. They both impose a fixed ordering on the features and retain the top-ranked ones. The difference between them is that general filters use domain-independent feature rankings, while heuristic filters

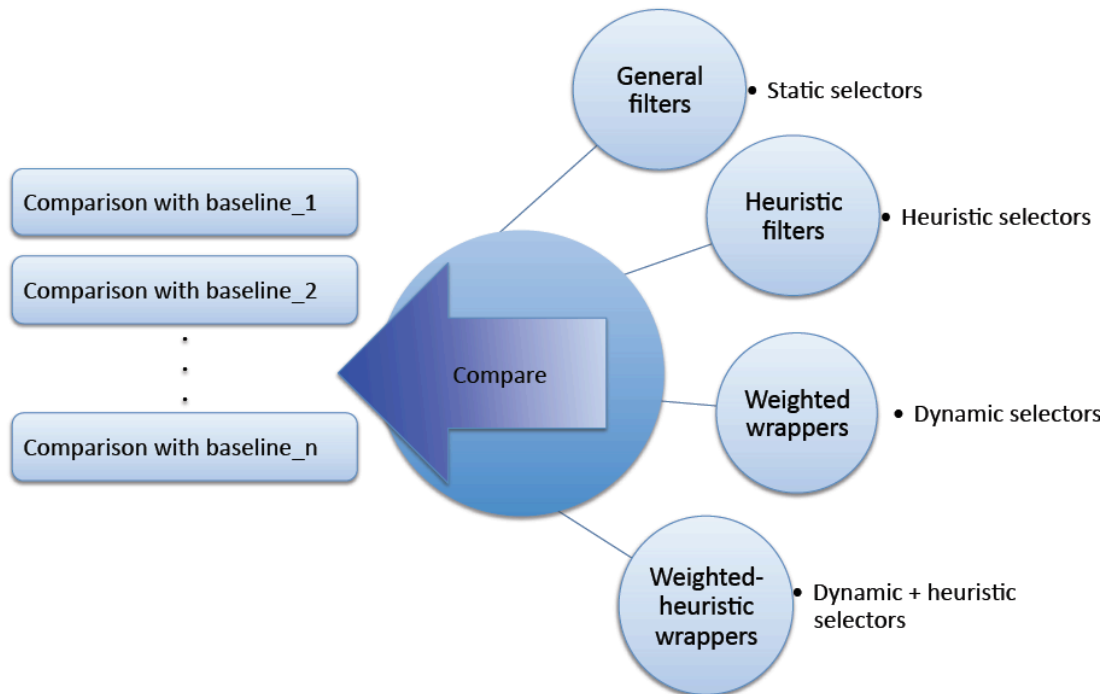


Figure 4.5: XFF, an experimental framework for feature selection. On the right are different kinds of feature selection. On the left, the subset selected by one of the selection methods is compared to any number of baselines.

apply domain-dependent knowledge. (In this research, the heuristic knowledge comes from spoken dialogue systems.) *Weighted wrappers* and *weighted-heuristic wrappers* are dynamic, that is, features' rankings change during the selection process under some weight update function. Here again, the first approach is general, while the second uses heuristic knowledge about the domain to guide the dynamic selection. Any learner L implemented in Weka can be used with any feature selection method.

To evaluate the subset S selected by one of the four feature selection methods M based on a training set T (on the right in Figure 4.5), XFF runs comparison experiments on a holdout set (on the left in Figure 4.5). Figure 4.6 shows pseudocode for FFX. First, one or more baseline feature-selection methods also select subsets S_1 to S_k on the training set T (line 7). Then XFF compares subset S to each baseline-

selected subset S_i on the holdout set O . It trains $L(O,S)$ (line 8) and $L(O,S_i)$ (line 11) using 10-fold cross validation and compares the resulting accuracies $A(L,O,S)$ (line 9) and $A(L,O,S_i)$ (line 12) averaged across the ten folds.

A *selector* is a function called by a feature selection method to identify a feature subset. Each feature selection method has its own pool of selectors. General filters use *static selectors* that impose a fixed ordering of the features. Heuristic filters use *heuristic selectors* that order the features guided by knowledge about the application domain. Filters apply a selector only once to select a feature subset.

Figure 4.7 shows pseudocode for the general structure of weighted wrappers in XFF. A *phase* in a wrapper (lines 1 – 12) is a sequence of subset selections and evaluations. A phase uses the same selector in all wrapper iterations (line 2). A

-
1. Specify a dataset D described by a set of features F
 2. Specify a partition of D into a training set T and a holdout set O
 3. Specify an accuracy metric A
 4. Specify a feature selection method M (e.g., a particular filter or a particular weighted wrapper)
 5. Specify baseline feature-selection methods B_1, \dots, B_k
 6. Let M select a subset S of F using the training set T
 7. Let B_1, \dots, B_k select subsets S_1, \dots, S_k of F using the training set T
 8. train $L(O,S)$ on O using 10-fold cross validation
 9. compute $A(L,O,S)$ averaged over the 10 folds
 10. **for each S_i do:**
 11. train $L(O,S_i)$ on O using 10-fold cross validation
 12. compute $A(L,O,S_i)$ averaged over the 10 folds
 13. run paired t-test and report $A(L,O,S)$ and $A(L,O,S_i)$ with their p values
-

Figure 4.6: Pseudocode for XFF to evaluate feature-selection method M .

dynamic phase uses a dynamic selector, while a *heuristic phase* uses a heuristic selector. Weighted wrappers can execute multiple *dynamic phases*, as specified by the input parameter n , each with a different dynamic selector. Given a learner L and a training set T , each wrapper iteration (lines 6 – 12) selects a new subset of features S , trains L on T and S to record its accuracy $A(L,T,S)$ (lines 7 – 9), and updates the weights of the features in S (line 10). If an improvement in accuracy is observed, the weighted wrapper updates the maximally observed accuracy $A\text{-best}$ (lines 11 – 12). This is not hill-climbing because at each iteration only weights are updated, and the subset of features tested in the next iteration is randomly selected. Weighted-heuristic wrappers are like weighted wrappers, but first they execute one or more dynamic phases, and then they execute one or more heuristic phases.

WeightedWrapper(training set T , learner L , initial feature set F , accuracy metric A , number of wrapper phases n , termination *criterion*)

1. **for phase from 1 to n do:**
 2. specify a selector
 3. specify a weight initialization method
 4. specify a weight update method
 5. initialize all feature weights
 6. **do until** *criterion*
 7. select a subset S from the full feature set F using a selector
 8. train $L(T,S)$ on T using 10-fold cross validation
 9. compute $A(L,T,S)$ averaged over the 10 folds
 10. update the weights of the features in subset S
 11. **if** $A(L,T,S)$ is greater than the maximally observed accuracy $A\text{-best}$
 12. **then** update $A\text{-best}$
 13. select and return a final subset S_{final}
-

Figure 4.7: Pseudocode for a general weighted wrapper in XFF.

Table 4.4 lists the selectors currently implemented in XFF. SS2 and SS3 treat each feature as if it were a single-feature classifier, and measure its classification accuracy and κ , respectively. The *kappa coefficient* κ measures agreement among classifiers (Cohen 1960). SS5, SS6, and SS7 use different measures for subset similarity. In each of them, S' is a randomly sampled subset, and S is the subset

Table 4.4: Selectors currently implemented in XFF. General filters use static selectors, and heuristic filters use heuristic selectors. Weighted wrappers use dynamic selectors, and weighted-heuristic wrappers use both static and heuristic selectors.

Static selectors

SS1: Rank features by their correlation with the outcome and select the top ranked.

SS2: Rank features by their accuracy as a single-feature classifier and select the top ranked.

SS3: Rank features by their κ as a single-feature classifier, and select the top ranked.

SS4: Randomly sample features with a parameterized bias towards the top of the ranking (e.g., select 2/3 of the features from the top half ranking), for any of the rankings produced by SS1, SS2, and SS3.

SS5: Select out of m random subsets, the one most similar, as measured by

$$\frac{|S' \cap S|}{\max(|S'|, |S|)}, \text{ to the } n \text{ top ranked by SS1, SS2, or SS3.}$$

SS6: Select out of m random subsets, the one most similar, as measured by $\frac{|S' \cap S|}{|S' \cup S|}$, to the n top ranked by SS1, SS2, or SS3.

SS7: Select out of m random subsets, the one most similar, as measured by $\frac{2 \times |S' \cap S|}{|S'| + |S|}$ to the n top ranked by SS1, SS2, or SS3.

Heuristic selectors (with respect to the categories described in Section 3.2.3)

HS1: Maximize inter-group diversity (e.g., sample from different feature groups).

HS2: Maximize inter-sub-group diversity (e.g., sample from different feature sub-groups).

HS3: Maximize inter-meta-group diversity (e.g., sample from different feature meta-groups).

Dynamic selectors

DS1: Sample a random feature subset.

DS2: Perform roulette-wheel selection based on the dynamic weights.

DS3: Perform tournament selection based on the dynamic weights.

Table 4.5: The weight-initialization and weight-update methods currently implemented in XFF for weighted and weighted-heuristic selection

Weight initialization

WI1: Set all weights to be equal.

WI2: Set all weights to their correlation to the outcome.

WI3: Set all weights to the classification accuracy of a single-feature classifier.

WI4: Set all weights to the κ of a single-feature classifier.

Weight update

WU1: Update a feature's weight as the running average of the classification accuracy of classifiers it has trained so far.

WU2: Update a feature's weight as the running average of the κ of classifiers it has trained so far.

obtained by one of SS1, SS2, or SS3.

For weighted feature selection, any number of weight-initialization and weight-update functions also may be added to the framework. Table 4.5 lists the weight initializations and weight updates currently implemented in XFF for weighted wrappers. Different wrapper phases can use different weight initialization and update methods. Experimentation with XFF led to the design and evaluation of two novel algorithms for feature selection, one general and one knowledge-guided, discussed next.

4.4 Randomized feature weighting (RFW)

Ideally, a feature selection method scales to large data sets, yet is general and simple enough so that it is easily implemented and widely used. Such methods should offer the simplicity and computational efficiency of filters, but without their requirement for a pre-specified feature-relevance criterion. Effective feature selection must

balance exploration and exploitation during search through the space of all subsets of an initial feature set. RFW seeks a balance between the efficiency of filters and the accuracy of wrappers. To do so, RFW uses random sampling of feature subsets in a wrapper framework to learn feature weights. As implemented in XFF, RFW is a weighted wrapper with a single dynamic phase that uses a random selector.

RFW is best understood as a filter that uses a wrapper to learn its own feature-relevance criterion q from experience on a training set T with a learner L . For simplicity, hereafter we describe L as a classifier. Figure 4.8 provides pseudocode for RFW. RFW takes as its input the training set T , the learner L , the initial feature set F , the accuracy metric A , a termination parameter *persistence*, and a range $[min, max]$ from which to select the size of the subset $|S|$ as a percent of $|F|$ ($0 < min < max < 1$). Likely accuracy metrics A include the classifiers' kappa coefficient κ , and *%correct*, the percentage of instances correctly classified. If A is κ , for example, the weight for $f \in F$ will be the average κ of all the classifiers in whose training f participated. All feature weights are initialized to zero (line 1). RFW repeatedly selects a random subset S of features from F (line 4). The size of S varies somewhat because RFW randomly selects a value in $[min, max]$ to determine $|S|$ (line 3).

RFW(training set T , learner L , initial feature set F , accuracy metric A , termination criterion $persistence$, $|S|$ range $[min,max]$)

1. initialize the weights of all the features in F , A -best, and $unimproved$ to 0
2. **do until** $unimproved = persistence$
3. let $|S|$ be $v \cdot |F|$ with v randomly sampled in $[min,max]$
4. let S be $|S|$ randomly selected features from F
5. *EvaluateAndUpdateWeights*(L,T,S,A) to produce $A(L,T,S)$
6. **if** $A(L,T,S)$ is greater than the maximally observed accuracy A -best
7. **then** reset $unimproved$ to 0 and update A -best
8. **else** increment $unimproved$
9. **return** F' : a copy of F ranked by the features' acquired weight

EvaluateAndUpdateWeights(L,T,S,A)

- I. train $L(T,S)$ on T using 10-fold cross validation
 - II. compute $A(L,T,S)$ averaged over the 10 folds
 - III. update the weight of all features $f \in S$ as the running average of $A(L,T,S)$
 - IV. **return** $A(L,T,S)$
-

Figure 4.8: Pseudocode for RFW

RFW calls *EvaluateAndUpdateWeights* (line 5) to train and evaluate a classifier $L(T,S)$ with S on T using 10-fold cross validation. It uses the accuracy $A(L,T,S)$ of that classifier (averaged over the 10 folds) to set the weight of each feature $f \in S$ to the running average of the accuracy of the learners in whose training f participated thus far. RFW tracks *unimproved*, i.e., how many consecutive times $A(L,T,S)$ does not improve the maximally observed accuracy A -best. RFW repeatedly selects random subsets and evaluates classifiers trained on them (lines 2 – 8) until *unimproved* reaches the user-specified criterion *persistence*, at which point RFW freezes the

weights. The final weight of each feature $f \in F$ is RFW's approximation of the true feature-relevance criterion q . Finally, RFW returns F' , a copy of F ranked by w .

RFW is not hill-climbing because it selects a random feature subset at each iteration, and uses the evaluation of the subset only to update weights, not to select the next subset. Run time for RFW depends on the size $|S|$ of the random subset, the definition of *persistence*, and L .

4.4.1 Experimental design for RFW

To evaluate RFW, we compared it to the state-of-the-art CFS filter and to what we refer to here as *GFS* (Greedy Forward Selection), the original John, Kohavi and Pfleger wrapper. (Both algorithms were discussed in Section 4.2.1.) CFS and GFS are both implemented and readily available in Weka (Hall, et al. 2009). All experiments reported here were conducted within XFF, which compared RFW to the baselines CFS and GFS using 10-fold cross validation on the holdout set O . All comparisons were repeated using Weka's implementation of three different learners for L : a C4.5 decision tree (Quinlan 1993), a multinomial logistic regression model (leCessie and van Houwelingen 1992), and a support vector machine (*SVM*) (Platt 1998).

To focus on the performance of the feature selection method, rather than on the predictability of an individual wizard's behavior, we trained the voice-search, no-query and yes-query models (described in Section 4.1) on data from all the wizards.

Recall that the voice-search model predicts whether the wizard performed voice search or not, and is trained on all the data from the full dialogue experiment. We refer to the dataset for this model as DIALOGUE ALL. The no-query model, on the other hand, predicts the wizard’s behavior given that there had been no database query. It is, therefore, trained only on instances where there had been no search; the dataset for it is called DIALOGUE UNSEARCHED. Finally, the yes-query model determines the wizard’s behavior given that there had been a query; it is trained only on instances where there had been a database query and its data set is called DIALOGUE SEARCHED. Only DIALOGUE SEARCHED references all 163 features. DIALOGUE UNSEARCHED and DIALOGUE ALL do not reference features about a query (i.e., eight features in the query

Table 4.6: Data sets for comparison of RFW with CFS and GFS baselines, the number of features, the number of instances, the number of classes, and whether the number of instances was reduced from the original data set.

Data set	Features	Instances	Classes	Reduced
UCI REPOSITORY				
CARDIOTOGRAPHY	23	2,126	3	x
ROBOT NAVIGATION	24	5,456	4	x
STEEL PATES FAULTS	27	1,941	7	x
BREAST CANCER (W,D)	32	569	2	x
COVERTYPE	54	10,000	7	✓
MUSK (VERSION 2)	168	6,598	2	x
ISOLET	617	1,600	26	✓
DIALOG				
DIALOGUE ALL	151	16,956	2	x
DIALOGUE CORRELATED	151	16,956	2	x
DIALOGUE UNSEARCHED	151	10,901	2	x
DIALOGUE SEARCHED	163	6,055	2	x

feature sub-group and four features in the comprehensive confidence feature sub-group that pertain to query confidence). These two datasets, therefore, reference 151 features instead.

To test the generality of RFW, we also experimented with seven data sets from the UCI repository (Frank and Asuncion 2010). We chose these particular data sets because they had both many instances and many features. Table 4.6 lists the data sets, the number of features, the number of instances, and the number of outcomes. (A brief description of each UCI data set appears in Appendix D.) DIALOGUE CORRELATED is DIALOGUE ALL with eight of its original features artificially highly correlated with the outcome. This data set was produced to compare RFW to CFS in its ability to detect relevant features as those highly correlated to the outcome. Because GFS could not complete execution within 24 hours on two of the UCI datasets, they were reduced for comparison with GFS, as indicated by the last column in Table 4.6. The COVERTYPE dataset was reduced to 10,000 from its original 581,012 instances, and the ISOLET dataset was reduced to 1,600 instances from its original 7,797. RFW did complete execution on the full data sets with C4.5 and SVM, and those results are reported in the next section.

For each data set in Table 4.6, 20% of the instances were randomly selected for the holdout set O , and the remainder were retained for the training set T . Comparisons between RFW and GFS and CFS are averaged over 10-fold cross validation on the holdout set O . Because RFW returns a ranking of the features, for each data set and for each learner, our experiments compare RFW's top-ranked features, denoted as $\text{RFW}_{|S_{top}|}$, for subset sizes $|S_{top}| \in \{5, 10, 15, 20\}$. Because the ISOLET data set had

many more features, we used $|S_{top}| \in \{10, 15, 20, 30\}$ for it. In all experiments, RFW randomly selected the subset size $|S|$ in the range $[0.1|F|, 0.3|F|]$, and terminated with $persistence = 75$, unless otherwise noted. In preliminary testing, no empirical difference was detected when A was κ or $\%correct$. Therefore, all experiments reported here used κ . All experiments ran on an iMac 2.93 GHz Intel Core i7 with 8GB of memory.

4.4.2 Experimental results with RFW

Results for C4.5 appear in Figures 4.9 through 4.21. These report on the percentage of instances correctly classified by the decision tree trained on the subset selected by a feature selection method, and the time, in seconds, it took the feature selection algorithm to select a feature subset. A statistically significant difference in accuracy from CFS and/or GFS ($p < 0.05$) is indicated by shape. An x indicates that RFW outperformed a baseline, a – indicates that it underperformed a baseline, a square indicates that it outperformed both baselines, a circle indicates that it outperformed one baseline and underperformed the other, and a diamond indicates no difference from the baselines. More thorough statistical methods to compare multiple classifiers over multiple datasets are available, but beyond the scope of this dissertation (Demsar 2006). Tables with detailed results for C4.5, for logistic regression and for SVM are provided in Appendix E.

Preliminary testing showed no empirical difference among *persistence* = 75, 100, and 200. Data reported here is primarily on *persistence* =75. On three data sets for C4.5, however, a larger *persistence* value made RFW more accurate, and is indicated with a – following the algorithm name (e.g., RFW20–200).

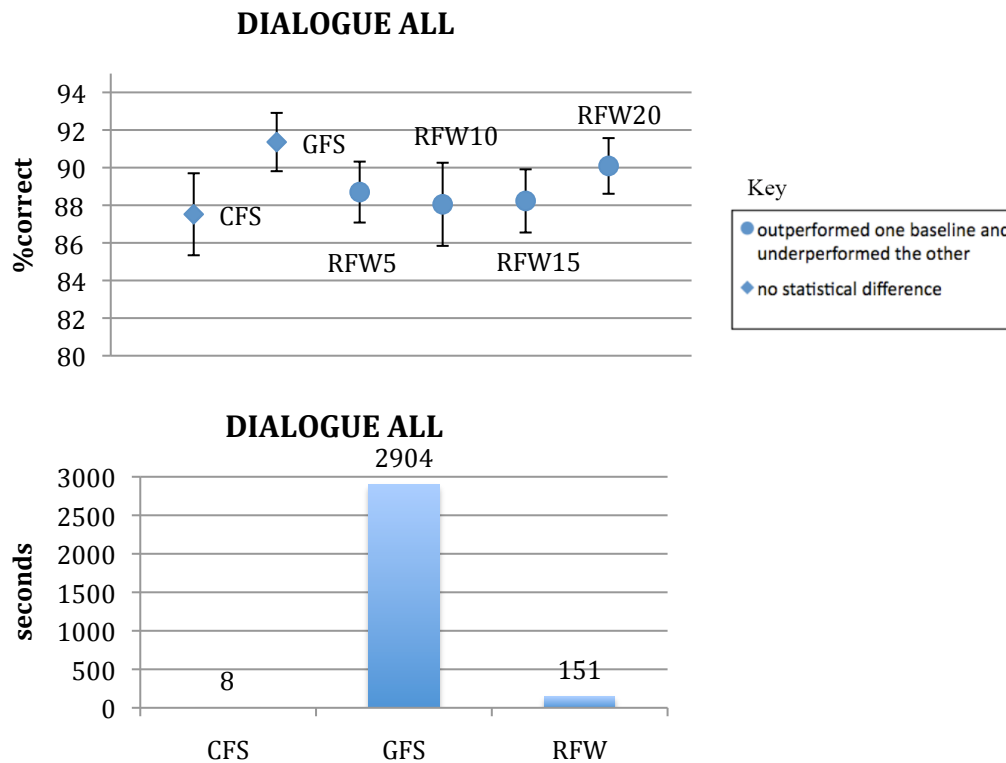


Figure 4.9: Results for feature selection with a C4.5 learner on the DIALOGUE ALL data set. %correct is reported as mean with the error bars indicating the standard deviation. Time is reported to the nearest second.

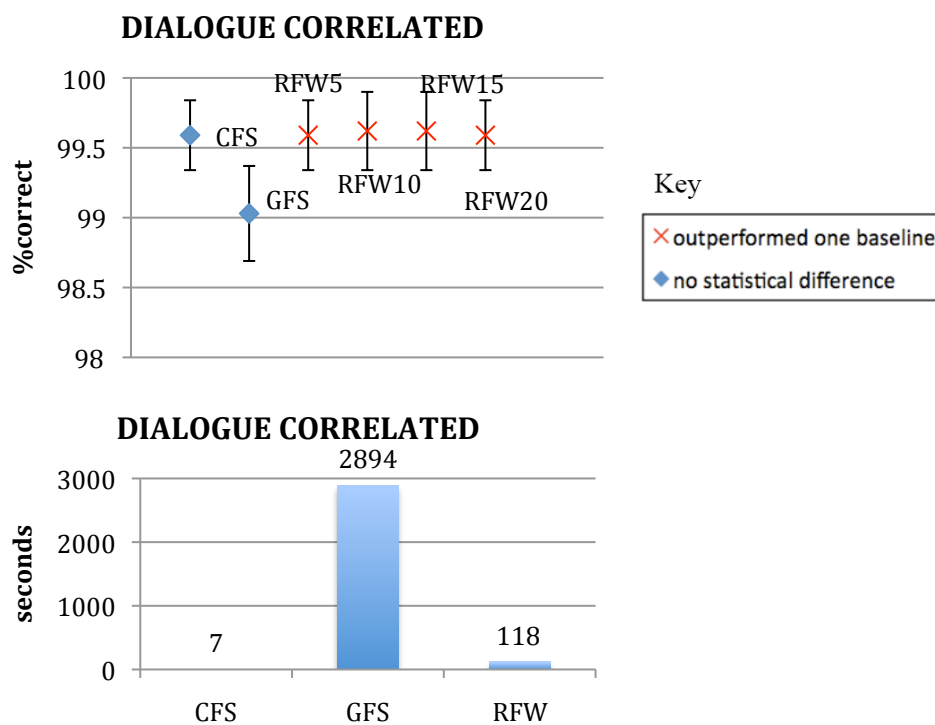


Figure 4.10: Results for feature selection with a C4.5 learner on the DIALOGUE CORRELATED data set. %correct is reported as mean with the error bars indicating the standard deviation. Time is reported to the nearest second.

On DIALOGUE UNSEARCHED with *persistence* = 200, RFW20–200 outperformed both CFS and GFS, while with *persistence* = 75, it outperformed only CFS on all subset sizes. On MUSK with *persistence* = 100, RFW20–100 also outperformed both CFS and GFS, while with *persistence* = 75, it outperformed only GFS with subset sizes 5 and 15. On ISOLET with *persistence* = 200, RFW20–200 outperformed GFS, but with *persistence* = 75, it was as good as either CFS or GFS.

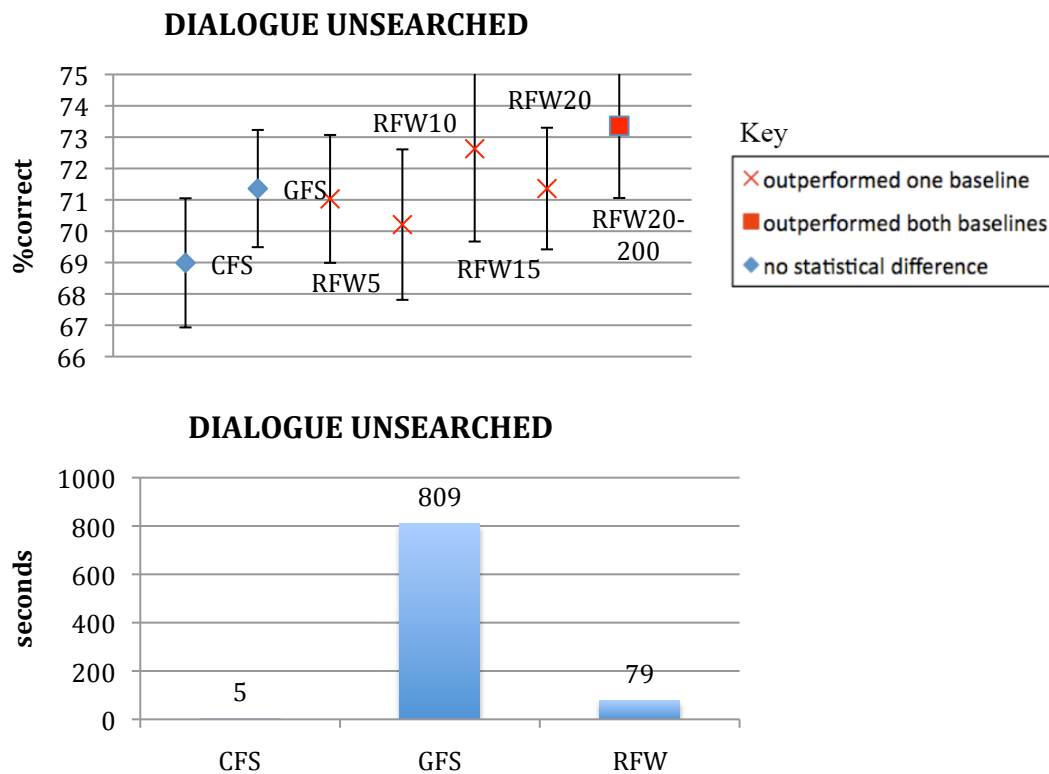


Figure 4.11: Results for feature selection with a C4.5 learner on the DIALOGUE UNSEARCHED data set. %correct is reported as mean with the error bars indicating the standard deviation. Time is reported to the nearest second.

In general for C4.5, at least one and often most RFW subsets were at least as accurate as the better of CFS and GFS on 10 of the 11 data sets. DIALOGUE ALL is the exception; all of RFW's subsets outperformed CFS but underperformed GFS. On the seven data sets where RFW was at least as accurate as CFS but did not outperform GFS (DIALOGUE UNSEARCHED, DIALOGUE SEARCHED, CARDIOTOGRAPHY, ROBOT NAVIGATION, STEEL PLATES FAULTS, BREAST CANCER, COVETYPE), RFW and GFS were equally accurate, but RFW was always at least an order of magnitude faster.

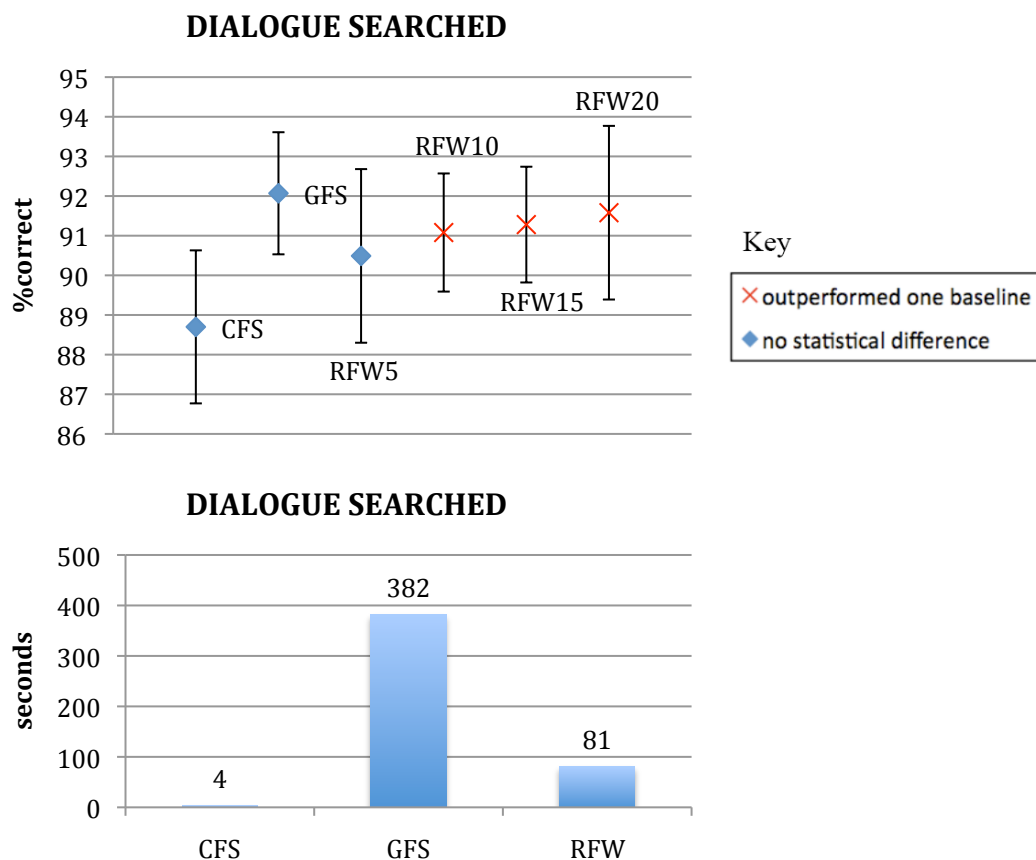


Figure 4.12: Results for feature selection with a C4.5 learner on the DIALOGUE SEARCHED data set. %correct is reported as mean with the error bars indicating the standard deviation. Time is reported to the nearest second.

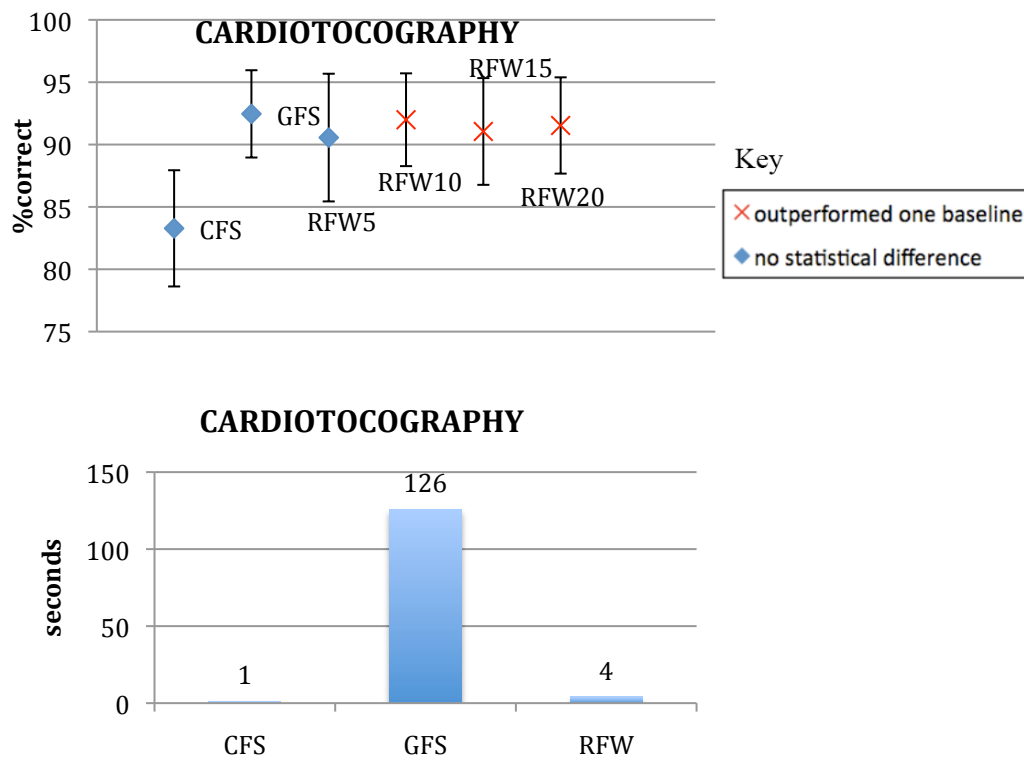


Figure 4.13: Results for feature selection with a C4.5 learner on the CARDIOTOCOGRAPHY data set. %correct is reported as mean with the error bars indicating the standard deviation. Time is reported to the nearest second.

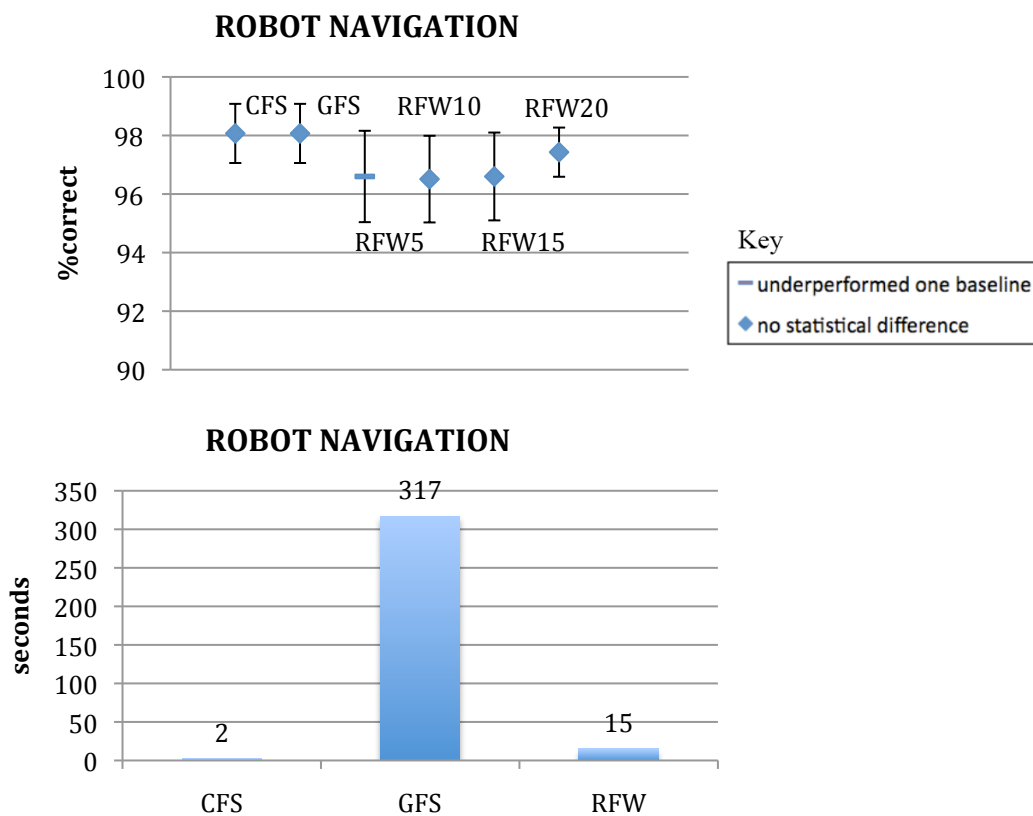


Figure 4.14: Results for feature selection with a C4.5 learner on the ROBOT NAVIGATION data set. %correct is reported as mean with the error bars indicating the standard deviation. Time is reported to the nearest second.

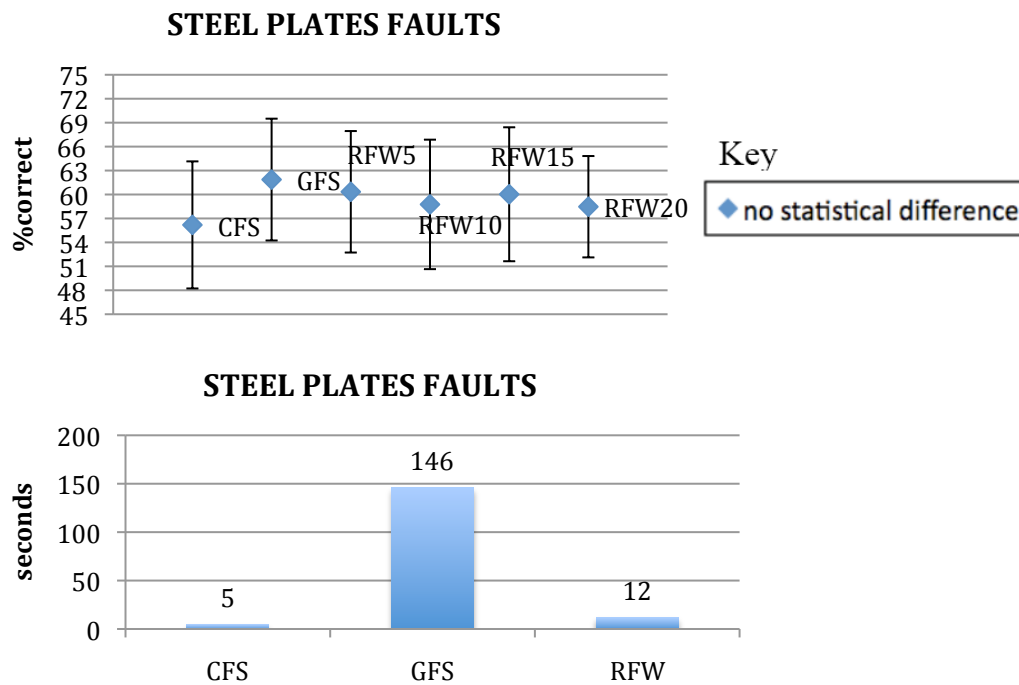


Figure 4.15: Results for feature selection with a C4.5 learner on the STEEL PLATES FAULTS data set. %correct is reported as mean with the error bars indicating the standard deviation. Time is reported to the nearest second.

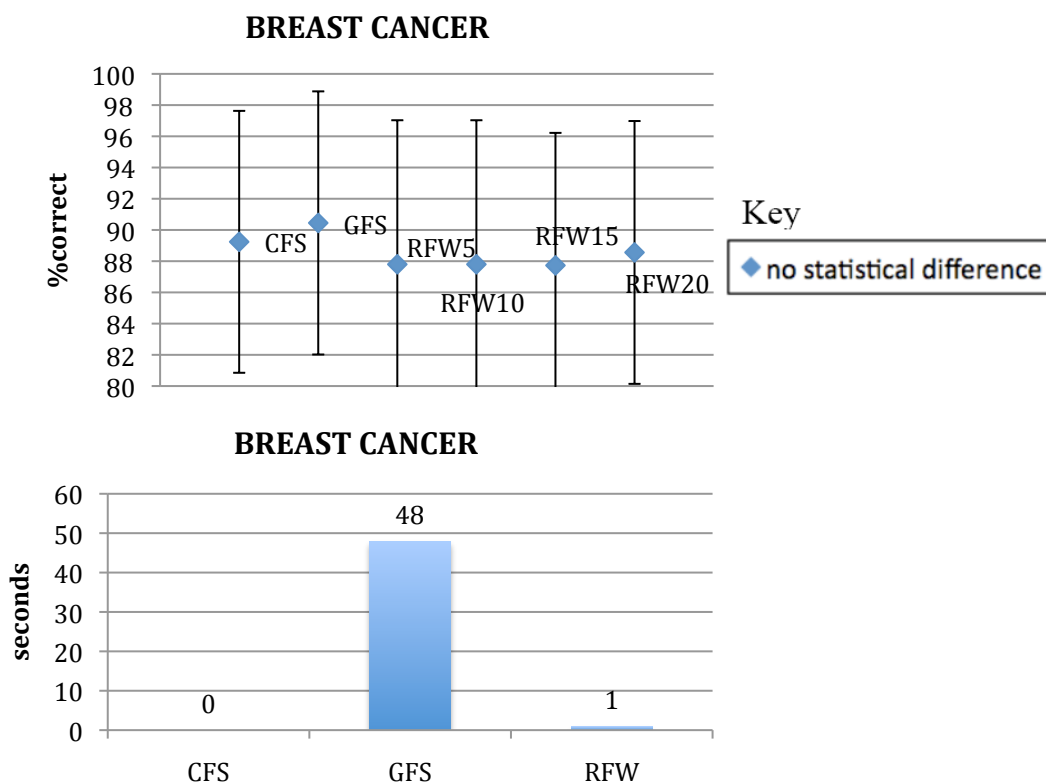


Figure 4.16: Results for feature selection with a C4.5 learner on the BREAST CANCER data set. %correct is reported as mean with the error bars indicating the standard deviation. Time is reported to the nearest second.

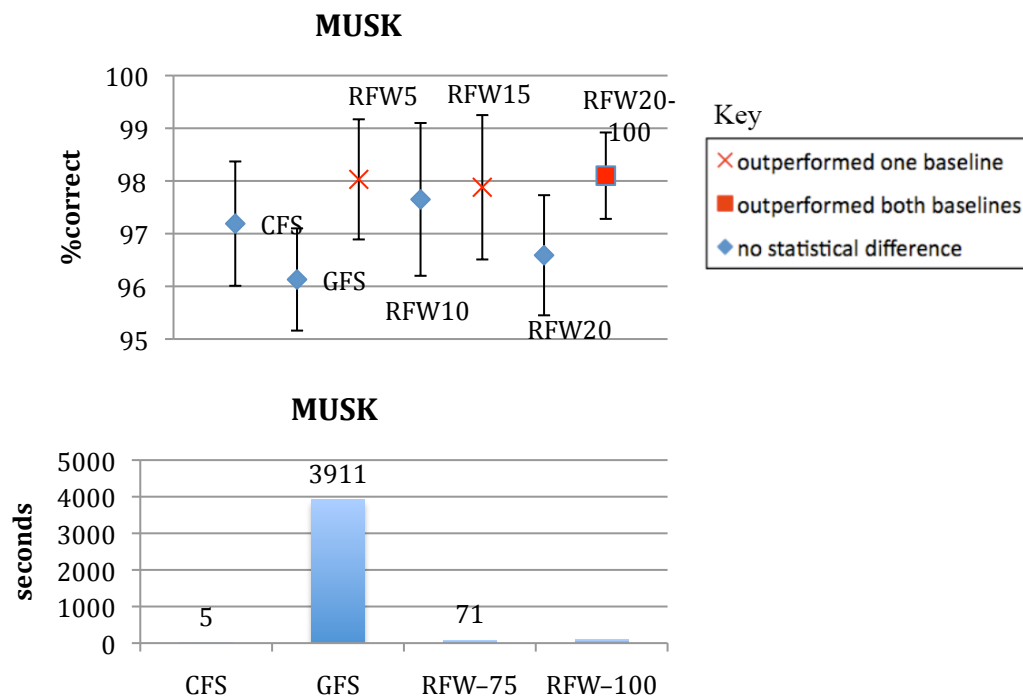


Figure 4.17: Results for feature selection with a C4.5 learner on the MUSK data set. %correct is reported as mean with the error bars indicating the standard deviation. Time is reported to the nearest second.

In Figure 4.18 on the full COVERTYPE data set with C4.5, RFW outperformed CFS on all subset sizes, but GFS did not complete within 24 hours. On the reduced COVERTYPE data set with C4.5 in Figure 4.19, GFS and RFW had equivalent accuracy both greater than CFS, except for RFW5. Every test with RFW, however, was two orders of magnitude faster, even when it produced a subset half the size of GFS.

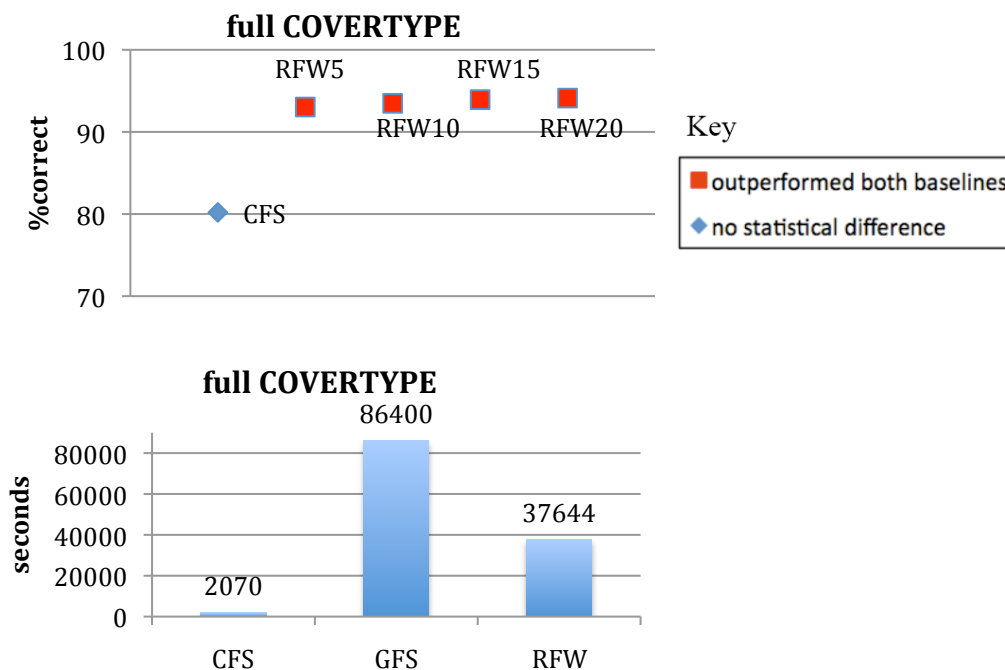


Figure 4.18: Results for feature selection with a C4.5 learner on the full COVERTYPE data set. %correct is reported as mean with the error bars indicating the standard deviation. Time is reported to the nearest second. GFS did not complete in 24 hours. Error bars are too small to be visible.

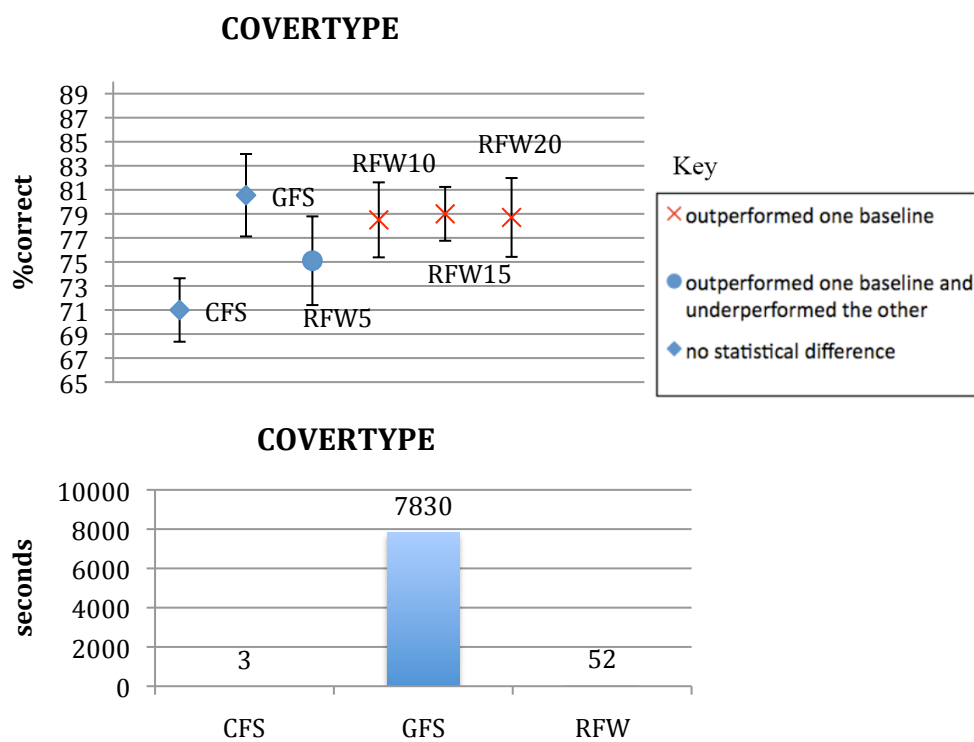


Figure 4.19: Results for feature selection with a C4.5 learner on the reduced COVERTYPE data set. %correct is reported as mean with the error bars indicating the standard deviation. Time is reported to the nearest second.

On the full ISOLET dataset in Figure 4.20, GFS did not complete in 24 hours, and RFW with *persistence* = 75 underperformed CFS. With *persistence* = 100, however, RFW30–100 with C4.5 was as accurate as CFS, which selected 192 features. Although RFW selected far fewer features, it did so more slowly (3,206 seconds compared to 316 for CFS).

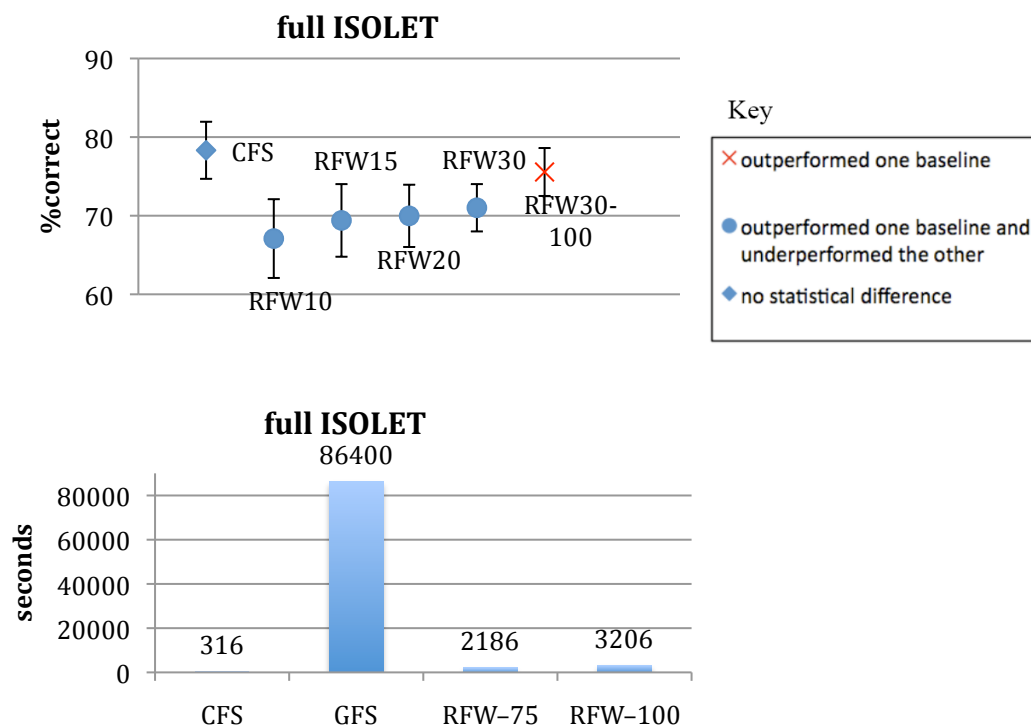


Figure 4.20: Results for feature selection with a C4.5 learner on the full ISOLET data set. %correct is reported as mean with the error bars indicating the standard deviation. Time is reported to the nearest second. GFS did not complete in 24 hours.

On the reduced ISOLET dataset with C.4.5 in Figure 4.21, GFS took about 21 hours; RFW with *persistence* = 75 was as accurate as both GFS and CFS on all subset sizes in under 8 minutes. In 14 minutes, RFW20–200 was more accurate than GFS, and equivalent to CFS with a far smaller feature subset.

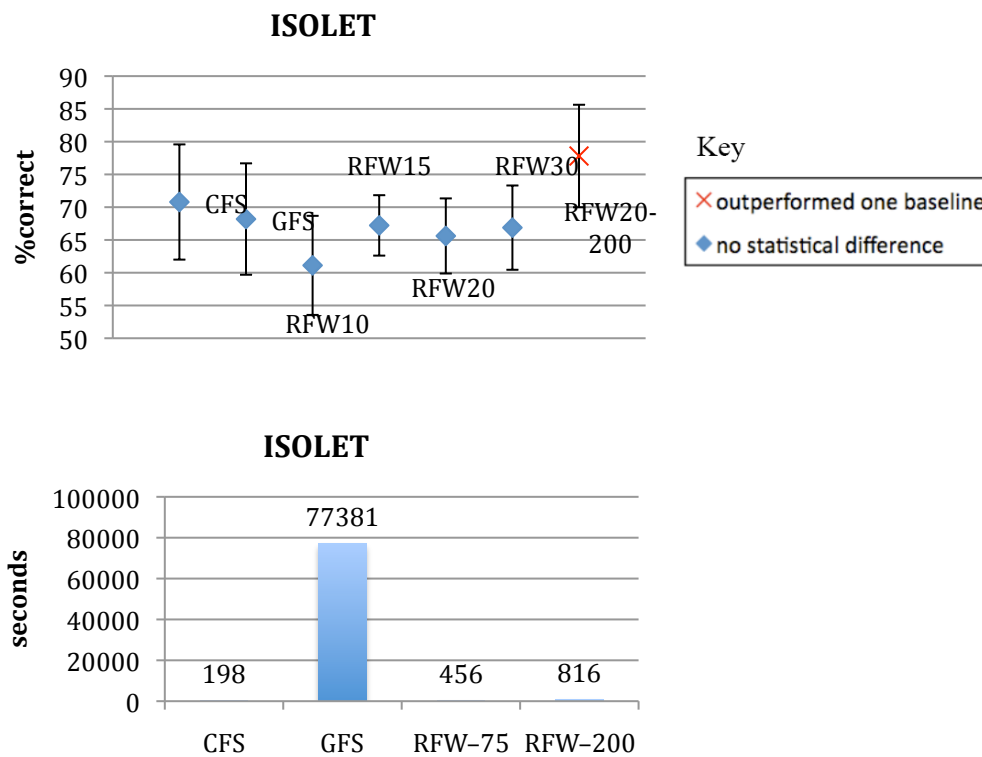


Figure 4.21: Results for feature selection with a C4.5 learner on the reduced ISOLET data set. %correct is reported as mean with the error bars indicating the standard deviation. Time is reported to the nearest second.

When the learner L is logistic regression or SVM (results reported in Appendix E), RFW also consistently performs as well as the better of CFS and GFS, and often outperforms them. There are, however, variations in accuracy from one learner to another, as one would expect.

4.4.3 Discussion of the results with RFW

RFW's use of randomly-selected subsets promotes exploration in the space of feature subsets $P(F)$. To avoid the bias of greedy selection, RFW learns feature weights from the evaluation of classifiers learned on those random subsets, but does not bias the selection of S by the weights. This allows relevant features to emerge, because any feature that often helps train good classifiers will eventually acquire a high weight. The learned feature weights are only considered for the final ranking of the features. RFW generally performs as well as, and sometimes better than, the GFS wrapper but considers fewer subsets during search, as evidenced by the running times. RFW tests moderately many random subsets of features, and records how those subsets performed. It thereby avoids the bias of a filter's pre-defined feature relevance metric, and strikes a good balance between the desirable properties of filters and wrappers. Moreover, a desirable property of filters is their ability to rank features (Liu, et al. 2010). Although RFW is a wrapper, it learns and stores feature weights, and it returns a ranked list of features by weight, much like a filter. Feature weights provide a feature-relevance metric that is not pre-specified, but rather learned from experience

in building classifiers on the data. These weights may themselves be of interest to investigators of the data set.

On data sets where some features are highly correlated with the outcome, a method like CFS that is tailored to that property should work best. Recall that DIALOGUE CORRELATED was artificially constructed to test that premise. Here too, RFW performed as well as CFS, albeit slower. Inspection indicates that RFW with all three learners identified and selected all the artificially correlated features. Further inspection of the overlap among features selected by the three methods indicates that, on some data sets there was considerable agreement. For example, with C4.5, when features are selected by both CFS and GFS, RFW learns high weights for them and ranks them high. On CARDIOTOGRAPHY, all three methods share 5 features in their subsets of size 10. On STEEL PLATE FAULTS four features, and on ROBOT NAVIGATION six features, were common to GFS and CFS. These were among those ranked highest by RFW, and therefore included in all its subset sizes. On several data sets, however, the three feature selection methods made idiosyncratic choices with comparable performance, suggesting that different feature subsets can be equally desirable for a classifier and a data set. This is also supported by the fact that, in most cases, different size subsets of the top-ranked RFW features achieve comparable performance. In that case, a less search-intensive method like RFW can provide as good a starting point for a learner and do so far faster than search-intensive methods that seek to approximate an optimal subset that may not be unique.

The parameters for subset size $|S|$ and *persistence* were not tuned. Although, in preliminary trials with RFW on the data from the full dialogue experiment only, we

explored many options in combination, for the most part there were no apparent empirical differences in accuracy. The exception was the $[min, max]$ bounds for $|S|$, where the other intervals that we tested ($[0.04, 0.1]$, $[0.05, 0.15]$, $[0.2, 0.5]$, $[0.3, 0.6]$, $[0.4, 0.7]$) produced less reliable results. Petrovic and Epstein observed similar behavior (Petrovic and Epstein 2009).

We also tested further guidance for RFW with a second dynamic phase. After *persistence* iterations with randomly-selected subsets that showed no improvement in *A-best*, RFW continued to learn weights, but now it selected subsets biased by the features' weights, using roulette wheel selection in one experiment and tournament selection in another. (See Table 4.4.) The second dynamic phase terminated with the same halting condition as the first phase. We also tested ranking the features by placing at the top pairs and triplets of features that had acquired high weights when selected together, rather than ranking the features on their individual performance. Neither approach improved accuracy.

In summary, when the data is well understood and a filter selects good feature subsets, the extra computational cost incurred by RFW is unnecessary. When more exploration in the space of feature subsets is required, however, RFW adroitly balances computational cost and simplicity, while it proves consistently accurate and scalable. Nonetheless, domain knowledge, if available, can further guide feature selection for a specific application. The next section discusses RFW+, an extension of RFW with domain knowledge about spoken dialogue systems.

4.5 Randomized Feature Weighting with knowledge (RFW+)

Filters exploit knowledge about the relationship between features and the prediction outcome. When domain-specific knowledge is available and easy to use, a feature selection algorithm should exploit it too. There is a wealth of knowledge about the provenance of features in a spoken dialogue system, as discussed in Section 3.2.3. To select features to train models of error detection and recovery for spoken dialogue systems, we extended RFW with an additional phase that uses this knowledge to select from the features top-ranked by RFW.

Section 3.2.3 discussed the features available in spoken dialogue systems, and their possible categorization into groups, meta-groups, or sub-groups, relative to the SLU pipeline. As discussed in Section 2.2, Skantze and Eldlund showed that, based on their grouping of 10 dialogue system features into 4 categories, models trained on features from multiple categories outperformed models trained on features from individual categories (Skantze and Edlund 2004). Although some feature categories may be more relevant to, and should therefore have greater impact on, the prediction of error detection and recovery at any given point along the SLU pipeline, information from features at all stages of SLU may be informative. The inclusion of some features from each category, albeit to different extents, could improve prediction accuracy.

Randomized feature weighting with knowledge (*RFW+*) is a weighted-heuristic feature selection method implemented within XFF. First, it executes a dynamic phase

with a random selector (RFW); then it executes one or more heuristic phases that apply knowledge about the feature categories of Section 3.2.3. Given a subset S , an *underrepresented category* is a feature category that has relatively few features in S , and the *least-represented category* is the category with the fewest features in S . The heuristic phase is intended to add features from categories that are underrepresented in the subset selected by RFW in the first phase. RFW+ could alternatively work with any feature categorization (e.g., clustering), given that distribution over feature categories is considered desirable for the application.

RFW+ was inspired by genetic algorithms, and adds a hill-climbing phase to RFW. It attempts one principled heuristic modification at a time to RFW's subset and retains the modification only if it improves accuracy. In each modification attempt of the heuristic phase, RFW+ either with very low probability deletes a random feature, or it adds the top-ranked feature in the least-represented category. Unlike genetic algorithms, however, the heuristic phase begins with a subset already tuned for accuracy by RFW through its random exploration of the space of feature subsets $P(F)$. The search space is narrowed by RFW, and far fewer modification cycles are therefore necessary than those usually found in genetic algorithms.

Figure 4.22 provides pseudocode for RFW+. First RFW produces F' (line i), a ranking of the features by their weight. The user-specified input parameter R provides subset sizes $|S_{top}|$ of top-ranked features in F' to be refined, each in its own heuristic phase. For each $|S_{top}|$ in R (lines ii – iii), *HeuristicPhase* applies heuristic knowledge to modify S , the top $|S_{top}|$ ranked features in F' .

RFW+(training set T , learner L , initial feature set F , accuracy metric A , termination criterion, $|S|$ range $[min,max]$, top-ranked subset sizes R , categories C)

- i. execute the dynamic phase $RFW(T, L, F, A, \text{criterion}, [min,max])$ to produce the ranked features F'
- ii. **for** each subset size $|S_{top}| \in R$
- iii. execute $HeuristicPhase(T, L, F', A, |S_{top}|)$
- iv. return all subsets selected by each heuristic phase

HeuristicPhase($T, L, F', A, |S_{top}|$)

1. let S be the top-ranked $|S_{top}|$ features $\in F'$
2. initialize $A\text{-best}$ to the accuracy $A(L, T, S)$ of S
3. **do until** no feature in the least represented group improves S
4. **either** drop a randomly selected feature from S with probability 0.05
5. **or** from the least represented category in C add the top-ranked feature that is not already in S
6. *TrainAndEvaluate* (L, T, S, A) on the modified subset S to produce $A(L, T, S)$
7. **if** $A(L, T, S)$ is greater than the maximally observed accuracy $A\text{-best}$
8. **then** retain the modification to S and update $A\text{-best}$
9. return S

TrainAndEvaluate (L, T, S, A)

- I. train $L(T, S)$ on T using 10-fold cross validation
 - II. compute $A(L, T, S)$ averaged over the 10 folds
 - III. return $A(L, T, S)$
-

Figure 4.22: Pseudocode for RFW+.

HeuristicPhase attempts modifications to the current best feature subset S . It either removes a random feature with probability 0.05 (line 4), or from the least-represented feature category it adds the top ranked feature that is not already in the subset S (line 5). To limit the introduction of redundancy in S , at each modification attempt, a feature is added to S with probability inversely proportional to the correlation of that feature to the other features already in S . (This detail was omitted from the pseudocode for clarity.)

A -best is initially the accuracy of the $|S_{top}|$ top-ranked features (line 2). As *HeuristicPhase* repeatedly attempts modifications to S , *TrainAndEvaluate* (line 6) trains a classifier $L(T,S)$ on the modified subset S with 10-fold cross validation and returns its accuracy $A(L,T,S)$ averaged over the 10-folds. If $A(L,T,S)$ improves over A -best, *HeuristicPhase* retains the modification, otherwise it discards it (lines 7 – 8). *HeuristicPhase* halts when no feature in the least-represented category can be added that improves the maximally observed accuracy A -best.

RFW+ finally returns all subsets of sizes in R that were modified by *SelectWithHeuristic* (line iv). The user-defined subset sizes $|S_{top}| \in R$ are meant primarily for experimentation. If a single $|S_{top}|$ is provided, RFW+ returns a single final subset S , as would any wrapper.

4.5.1 Experiments and results with RFW+

Because RFW+ uses knowledge specific to spoken dialogue systems, we evaluated it on the DIALOGUE ALL, DIALOGUE SEARCHED and DIALOGUE UNSEARCHED datasets

produced in the full dialogue experiment. As we did for RFW, RFW+ was compared for accuracy to CFS and GFS using 10-fold cross validation on the holdout set O . For these experiments we used only logistic regression for L , because it facilitated the experiment in the next chapter, and because we had observed no substantial difference in model accuracy across the three learners (C4.5, logistic regression, and SVM) on the dialogue data sets. Figures 4.23 through 4.25 report results for RFW and for each RFW+ heuristic phase RFW+ $|S_{top}|$, with $|S_{top}| \in \{3, 5, 10, 15, 20\}$ and *persistence* = 75 on DIALOGUE ALL, DIALOGUE UNSEARCHED, and DIALOGUE SEARCHED, respectively. All other parameters are the same as they were in the experiments with RFW in Section 4.4.2.

Results for RFW+ are reported with groups, meta-groups, and subgroups as defined in Figures 3.12, 3.13, and 3.14 of Section 3.2.3, respectively (detailed results appear in Appendix F). For DIALOGUE ALL and DIALOGUE UNSEARCHED, query features and comprehensive confidence features that reference query features were excluded from the groups. In general, RFW+ can improve upon RFW by outperforming the baselines where RFW did not, or continuing to outperform them where RFW had already done so. On DIALOGUE ALL in Figure 4.23, RFW with logistic regression performed as well as both CFS and GFS. RFW+, however, improves on RFW. On DIALOGUE ALL, RFW+ always outperforms CFS; RFW+5 with meta-groups, RFW+15 and RFW+20 with sub-groups outperform both CFS and GFS.

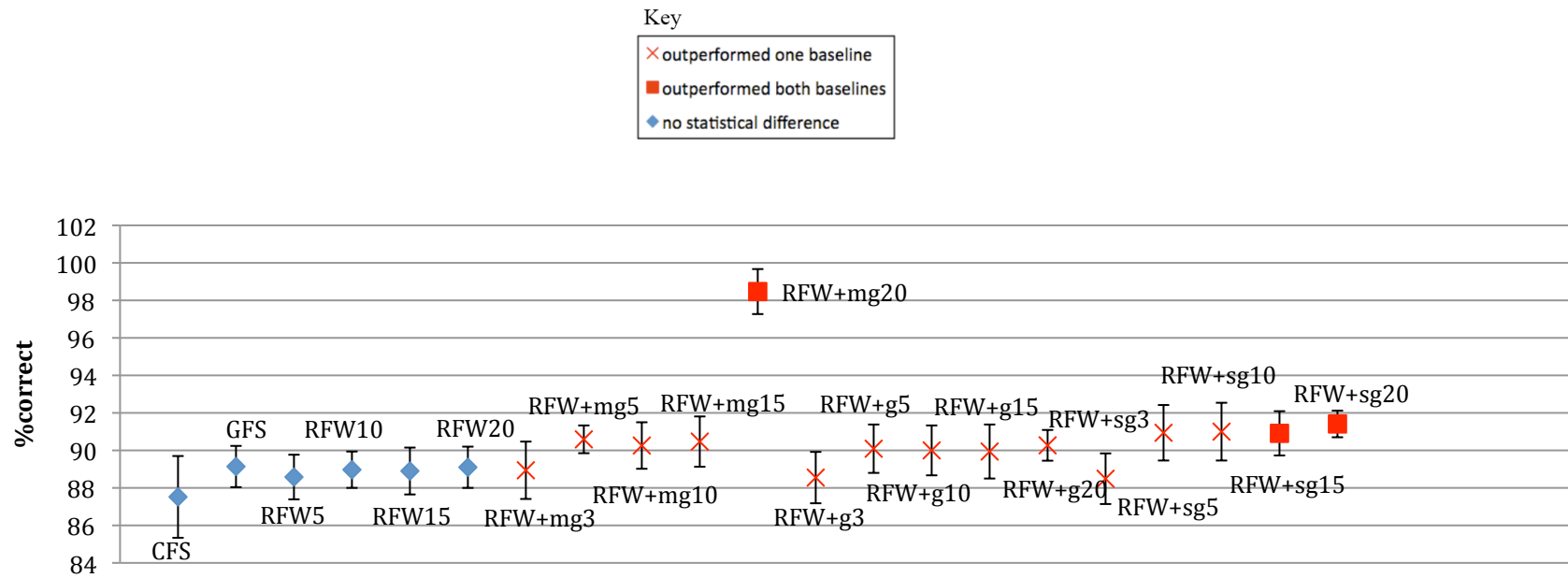


Figure 4.23: Results for RFW, RFW+ with meta-groups (RFW+mg), RFW+ with groups (RFW+g), and RFW+ with sub-groups (RFW+sg) with logistic regression on the DIALOGUE ALL data set. %correct is reported as mean with the error bars indicating the standard deviation.

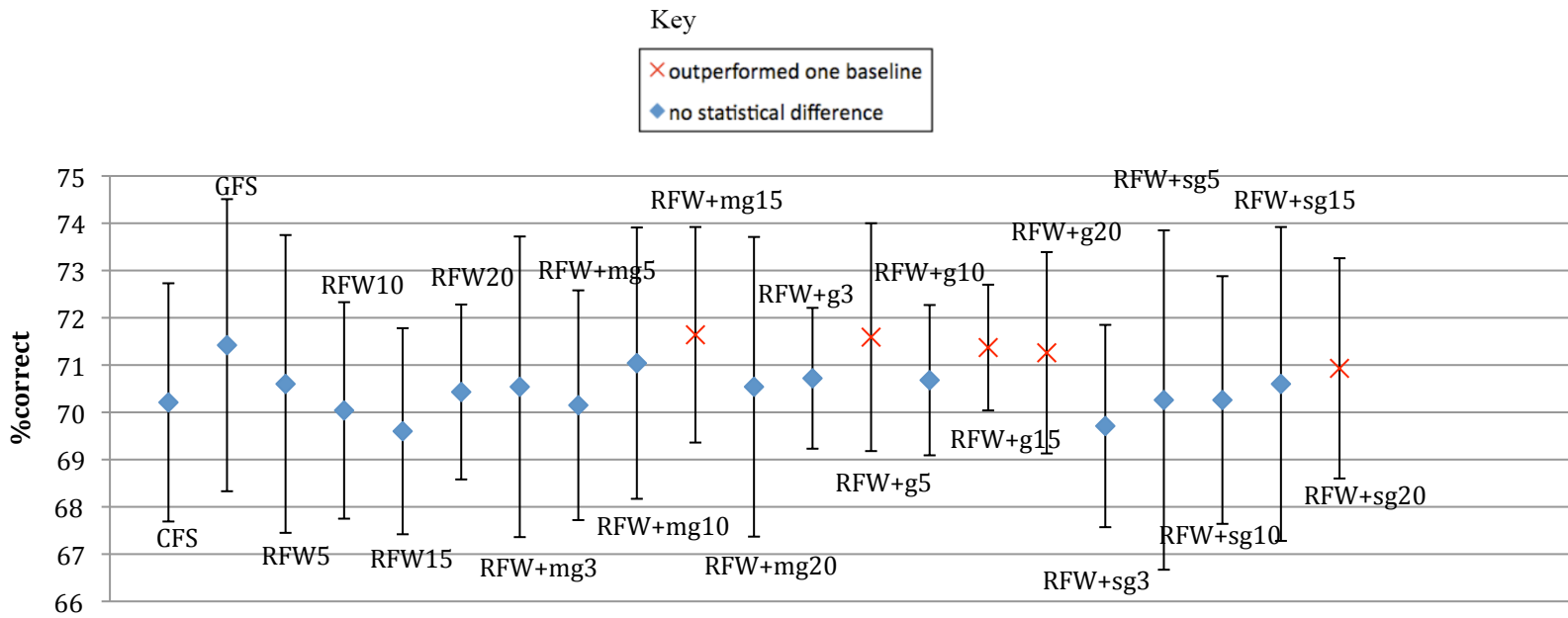


Figure 4.24: Results for RFW, RFW+ with meta-groups (RFW+mg), RFW+ with groups (RFW+g), and RFW+ with sub-groups (RFW+sg) with logistic regression on the DIALOGUE UNSEARCHED data set. %correct is reported as mean with the error bars indicating the standard deviation.

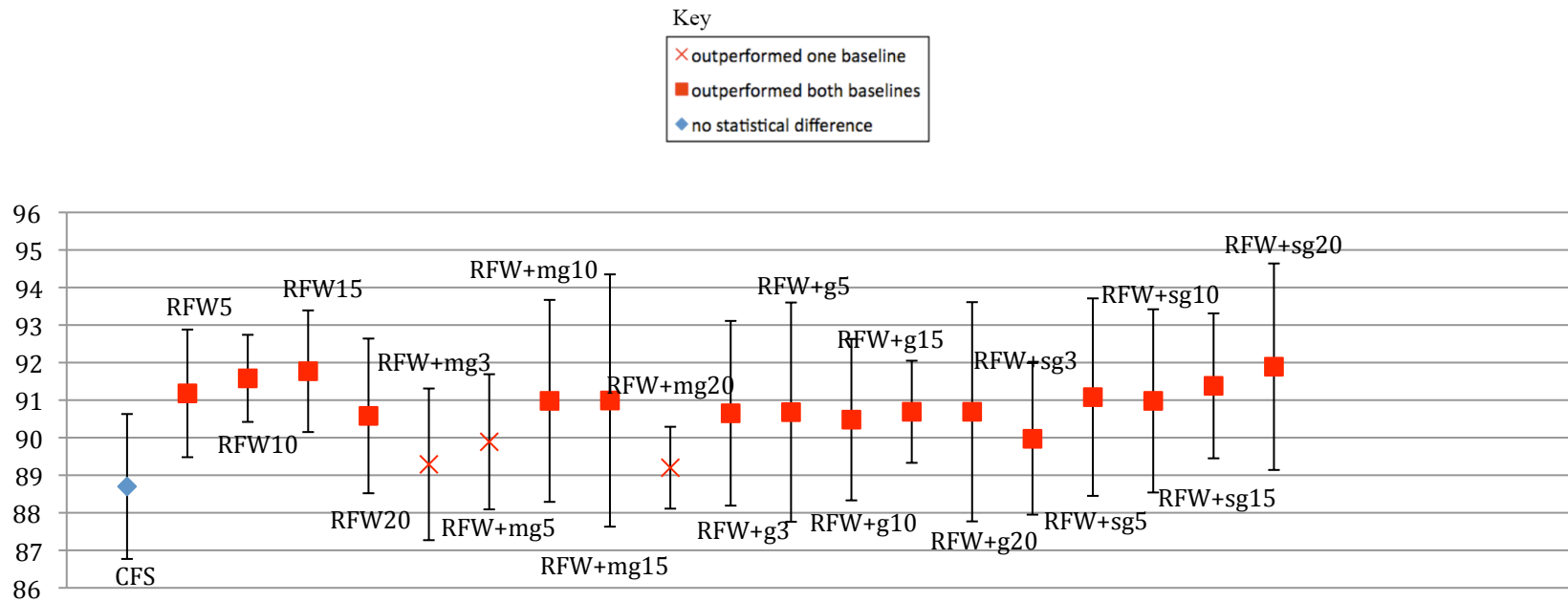


Figure 4.25: Results for RFW, RFW+ with meta-groups (RFW+mg), RFW+ with groups (RFW+g), and RFW+ with sub-groups (RFW+sg) with logistic regression on the DIALOGUE SEARCHED data set. %correct is reported as mean with the error bars indicating the standard deviation. GFS did not complete in 24 hours.

On DIALOGUE UNSEARCHED in Figure 4.24, RFW with logistic regression also performed as well as CFS and GFS. RFW+ improves on RFW by outperforming CFS under the following experimental conditions: RFW+15 with meta-groups, RFW+5, RFW+15 and RFW+20 with feature groups, and RFW+20 with sub-groups. RFW+ otherwise remains as accurate as CFS and GFS.

Finally, on DIALOGUE SEARCHED in Figure 4.25, RFW+ outperforms CFS on all but three experimental conditions. (RFW+3, RFW+5 and RFW+20 with meta-groups are the exceptions.) This result is consistent with what we had already observed for RFW on this data set. Overall, feature subsets with RFW+ tend to become larger. Recall that the heuristic phase $\text{RFW}+|S_{top}|$ begins with an $|S_{top}|$ -feature subset and attempts to modify it. With respect to dimensionality reduction, RFW+ may not be as competitive as RFW with user-specified subset sizes.

4.5.2 Discussion of the results with RFW+

Each time a feature from the least-represented category is added to or removed from S , that category may no longer be the least represented. In that case, a different category is considered in the following iteration. For each $|S_{top}|$, Tables 4.7 through 4.9 report the number of features added by the heuristic phase to the $|S_{top}|$ top ranked features, the number of iterations in the dynamic phase and, in the heuristic phase, the number of categories covered by the final subset selected by RFW+, and LR , the number of distinct least-represented categories ever considered during the heuristic

phase. (Recall that there are 3 meta-groups, 5 groups, and 10 sub-groups in all, and that DIALOGUE ALL and DIALOGUE UNSEARCHED do not reference 2 of the sub-groups).

Recall that RFW first randomly explores the space of feature subsets $P(F)$. Results with RFW indicate that its random exploration of $P(F)$ learns to rank the features well in relatively few iterations, as indicated by how many subsets it tests (the fourth column in Tables 4.7 through 4.9). RFW is competitive with much more search intensive methods like GFS, and RFW+ retains that desirable property.

Because the RFW phase narrows the search space, RFW+'s heuristic phase is not meant to continue extensive search in $P(F)$, as a genetic algorithm would. It is meant, instead, to balance the distribution of features across categories, under the assumption that all levels of SLU processing contribute to the prediction of error detection and recovery actions, although perhaps to different degrees. Recall that if no feature from

Table 4.7: Results for RFW+ on DIALOGUE ALL with logistic regression. Columns report the initial top-ranked subset size $|S_{top}|$, the number of features added by the heuristic phase, the number of iterations in the dynamic phase and in the heuristic phase, the number of groups covered by the final subset, and LR , the number of different categories ever considered to be least-represented.

category	$ S_{top} $	# added features	dynamic iterations	heuristic iterations	covered categories	LR
META-GROUPS	3	27	225	61	3	3
	5	11	225	28	3	3
	10	30	225	67	3	3
	15	17	225	35	3	3
	20	12	225	39	3	3
GROUPS	3	12	153	29	5	4
	5	10	330	24	5	4
	10	6	330	21	5	3
	15	5	330	10	5	2
	20	5	330	8	5	3
SUB-GROUPS	3	15	153	39	7	5
	5	10	153	35	6	6
	10	7	153	19	6	5
	15	5	153	11	8	3
	20	4	153	8	8	2

the least-represented category achieves improvement when added to S , the heuristic phase halts. This halting condition prevents excessive exploration of $P(F)$ in the heuristic phase. Its effects are illustrated by the number of changes made to the set of features originally selected by RFW (the fifth column in Tables 4.7 through 4.9). This condition can prevent the addition of any feature at all (as in DIALOGUE UNSEARCHED with sub-groups and $|S|=10$ in Table 4.8, and DIALOGUE SEARCHED with sub-groups and $|S| = 5$ in Table 4.9). RFW+ with sub-groups is clearly more prone to this behavior. With the finer granularity of sub-groups, there is a greater likelihood that no feature in the initial least-represented group can improve upon the original S .

The number of different categories ever labeled as least represented during the execution of RFW+ (the rightmost column in Tables 4.7 through 4.9) was comparable, regardless of how features are categorized. This is due to the halting

Table 4.8: Results for RFW+ on DIALOGUE UNSEARCHED with logistic regression. Columns report the initial top-ranked subset size $|S_{top}|$, the number of features added by the heuristic phase, the number of iterations in the dynamic phase and in the heuristic phase, the number of groups covered by the final subset, and LR , the number of different categories ever considered to be least-represented.

category	$ S_{top} $	# added features	dynamic iterations	heuristic iterations	covered categories	LR
META-GROUPS	3	15	218	42	3	3
	5	6	218	25	3	3
	10	3	218	145	3	1
	15	1	218	145	3	1
	20	4	218	14	3	1
GROUPS	3	7	227	20	4	3
	5	12	277	31	4	4
	10	7	277	21	5	4
	15	2	277	8	4	2
	20	4	277	13	5	2
SUB-GROUPS	3	5	281	10	5	2
	5	3	281	9	6	3
	10	0	281	5	5	1
	15	1	281	10	6	2
	20	3	281	12	8	2

condition and to the size of the categories. Meta-groups are larger, but there are fewer of them. Although there are more sub-groups, few were ever labeled least-represented, because RFW+ soon halted due to their small size. For the same reasons, the sixth column in Tables 4.7 through 4.9 indicates that RFW+'s final subset covered meta-groups and groups well, but that RFW+ can halt with a subset that includes few sub-groups.

To examine the change in accuracy produced by learning with the initial feature subset S and the one produced by the heuristic phase, Figures 4.26 through 4.28 plot A -best at each iteration during training for RFW+20, $A = \kappa$, and $A = \%correct$ with meta-groups, groups, and sub-groups, for DIALOGUE ALL, DIALOGUE UNSEARCHED, and DIALOGUE SEARCHED, respectively. The similar plateaus for κ and $\%correct$

Table 4.9: Results for RFW+ on DIALOGUE SEARCHED with logistic regression. Columns report the initial top-ranked subset size $|S_{top}|$, the number of features added by the heuristic phase, the number of iterations in the dynamic phase and in the heuristic phase, the number of groups covered by the final subset, and LR , the number of different categories ever considered to be least-represented.

category	$ S_{top} $	# added features	dynamic iterations	heuristic iterations	covered categories	LR
META-GROUPS	3	3	269	13	3	2
	5	11	269	41	3	3
	10	4	269	20	3	2
	15	3	269	20	3	2
	20	2	269	16	3	1
GROUPS	3	5	306	43	4	3
	5	5	306	56	5	3
	10	1	306	24	4	2
	15	5	306	35	5	3
	20	5	306	49	5	3
SUB-GROUPS	3	2	286	12	3	2
	5	0	286	15	3	1
	10	1	286	9	3	2
	15	5	286	23	8	3
	20	4	286	15	9	2

illustrate that they identify the same improvements.

Because RFW is non-deterministic, different RFW+ executions with a different categorization of the features begin the heuristic phase with different RFW rankings F' , and therefore different accuracies for the initial subset S . Figures 4.26 through 4.28 illustrate that all three knowledge-based categorizations can improve on the set

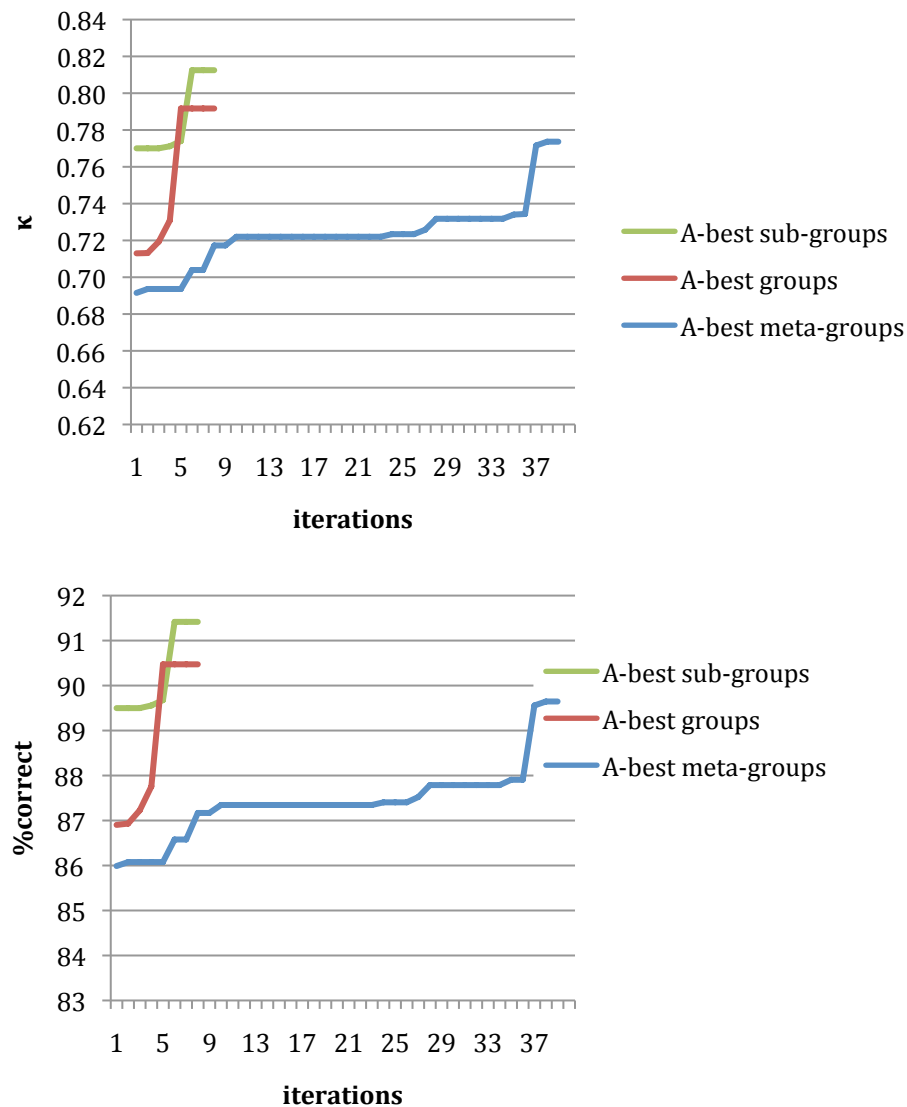


Figure 4.26: *A-best* in RFW+20 with *persistence* = 75 during testing on the training set T for DIALOGUE ALL with meta-groups, groups and sub-groups for $A = \kappa$ and $A = \%correct$.

initially selected by RFW. Note that with feature groups the size of that improvement is greater on all three data sets. (3.5% compared to 3.0% and 1.9% on DIALOGUE ALL, 0.88% compared to 0.40% and 0.21% on DIALOGUE UNSEARCHED, and 0.80% compared to 0.24% and 0.20% on DIALOGUE SEARCHED.) They also illustrate that during training on T , for each data set a different categorization of the features

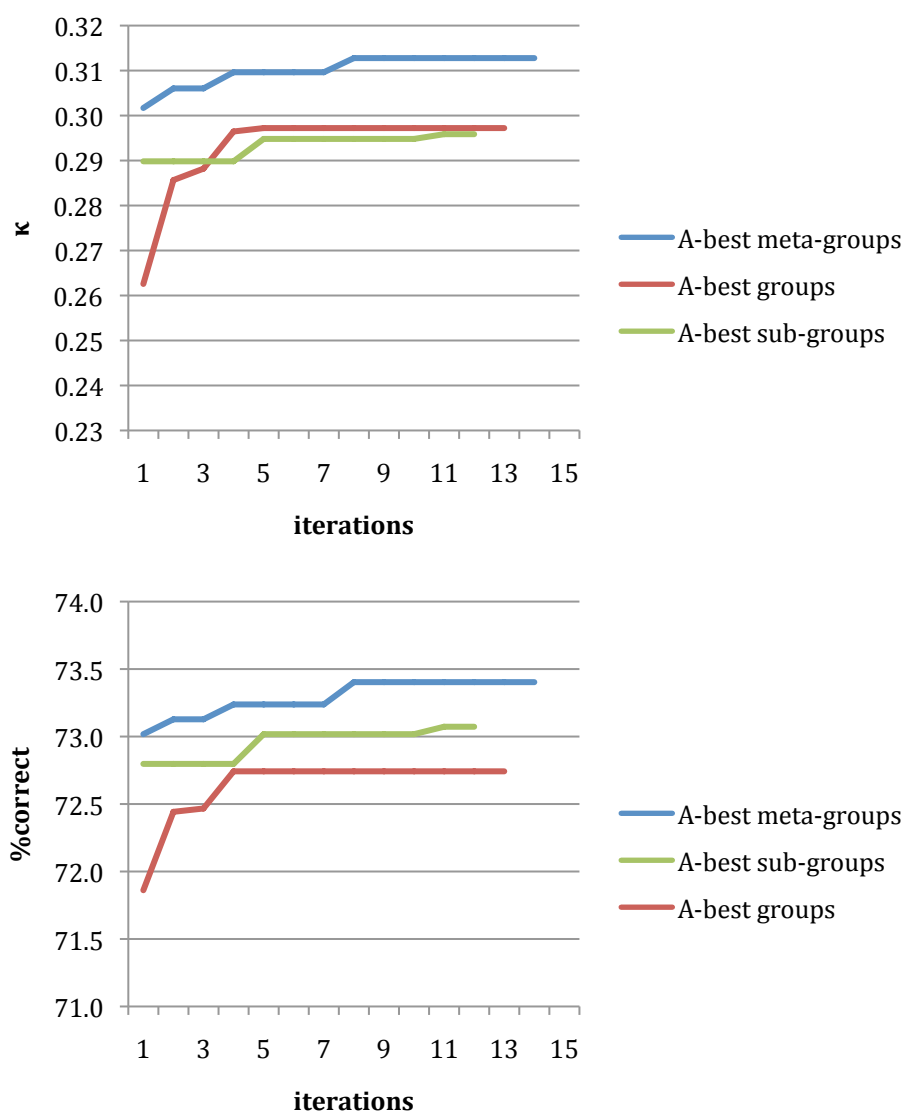


Figure 4.27: *A-best* in RFW+20 with *persistence* = 75 during testing on the training set T for DIALOGUE UNSEARCHED with meta-groups, groups and sub-groups for $A = \kappa$ and $A = \%correct$.

achieved greater accuracy on the final subset returned by RFW+, in part due to the initial difference in S 's accuracy.

Figure 4.29 shows that during testing on the holdout set O a different categorization achieved the greatest accuracy on each data set. Moreover, for each

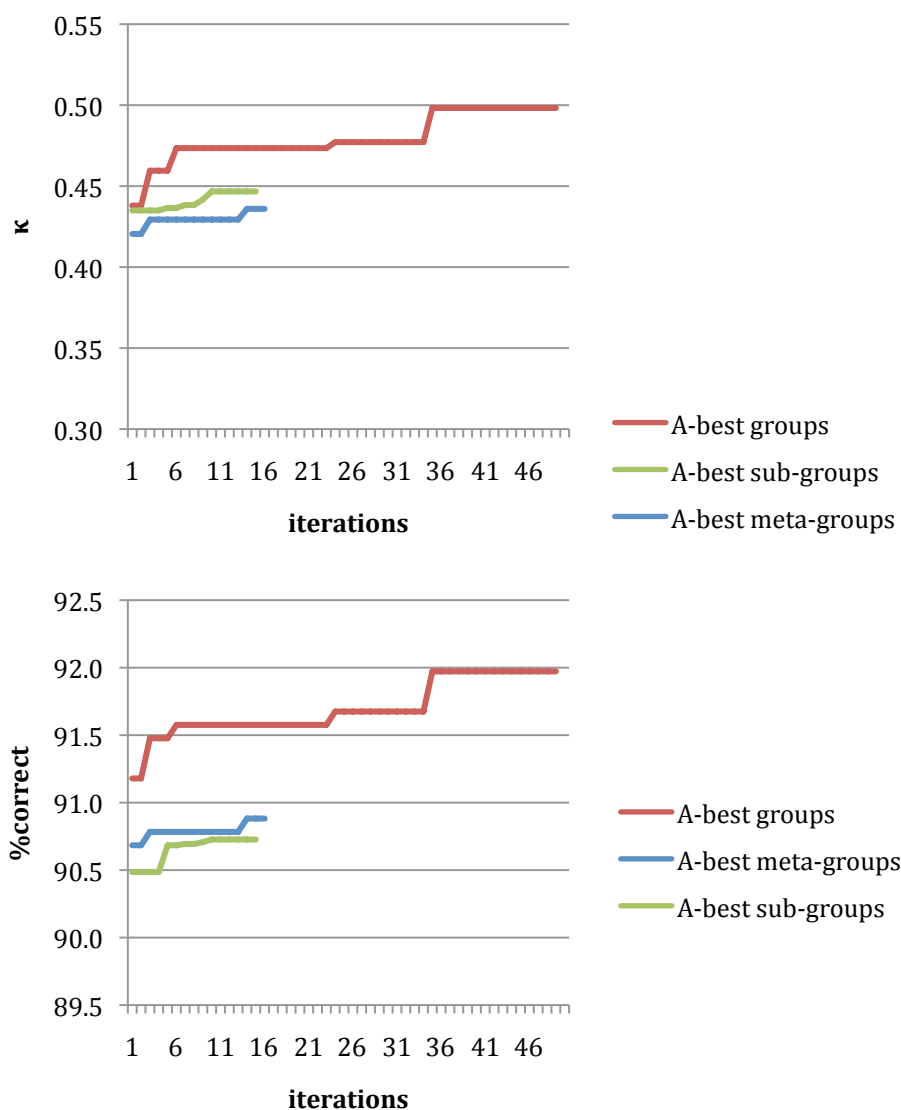


Figure 4.28: *A-best* in RFW+20 with *persistence* = 75 during testing on the training set T for DIALOGUE SEARCHED with meta-groups, groups and sub-groups for $A = \kappa$ and $A = \%correct$.

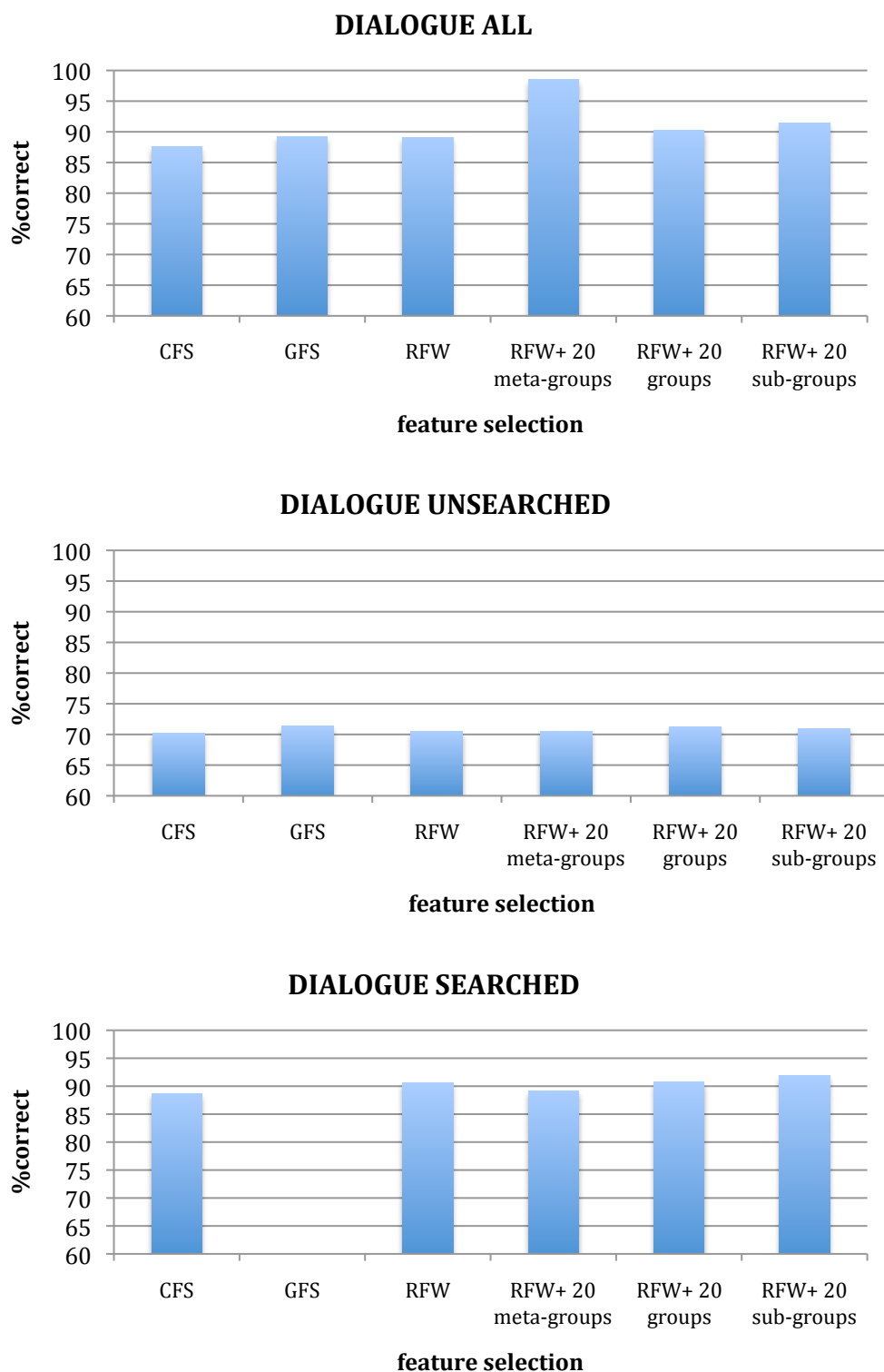


Figure 4.29: %correct averaged over 10 runs in testing on the holdout set O for feature selection methods on the three data sets from the full dialogue experiment. GFS selected no features on DIALOGUE SEARCHED.

data set, the most accurate categorization on the holdout set O was different from the most accurate categorization observed on the training set T . This is the reason why we

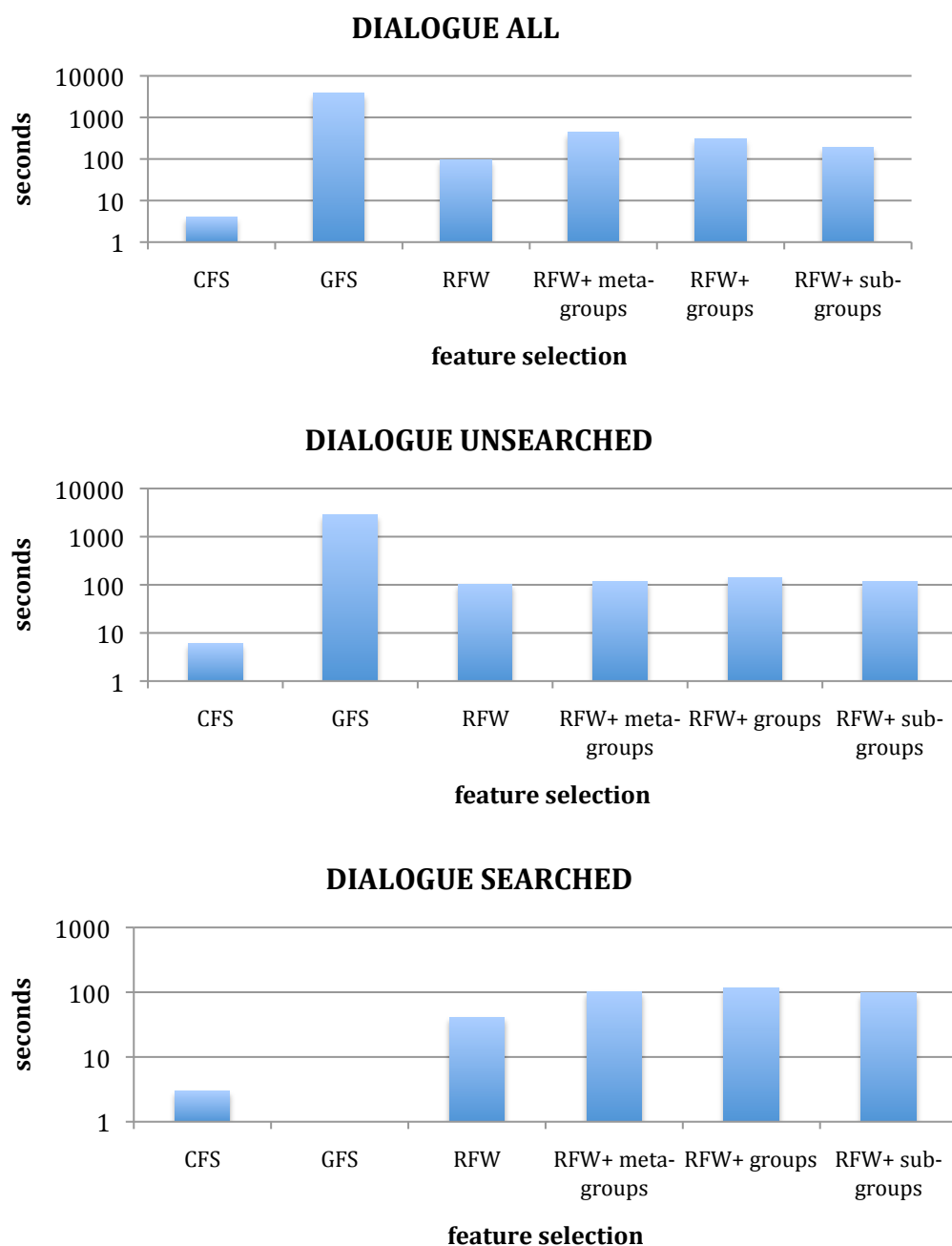


Figure 4.30: Execution time comparison in seconds, on a logarithmic scale, averaged over 10 runs in training on the training set T for feature selection methods on the three data sets from the full dialogue experiment. GFS selected no features on DIALOGUE SEARCHED.

ultimately chose not to commit to a granularity too coarse or too fine. Our results indicate that a categorization of medium granularity is a reasonable choice for the dialogue data, and that, across categories in this domain, RFW+ can improve upon RFW. RFW+ does take more time than RFW. Figure 4.30, however, indicates that its additional time is not dramatic. That difference could be further reduced if only a single subset size were supplied in R , as discussed in Section 4.5.

4.6 Summary

RFW is a general feature selection method that balances the qualities of filters and wrappers. It explores the feature subset space $P(F)$ to learn a feature relevance criterion that is tailored to the data and the learner. Unlike most wrappers, however, it can rank features well at low computational cost. RFW consistently performs as well as the state-of-the-art filter CFS and wrapper GFS, and sometimes it outperforms them. It is slower than a filter, but can be orders of magnitude faster than traditional wrappers.

RFW+ extends RFW with knowledge about the provenance of features along the SLU pipeline. Under the assumption that some representative features of each SLU stage should participate in error detection and recovery, RFW+ can improve upon RFW on the dialogue data sets. RFW+ outperforms CFS and GFS where RFW did not (on DIALOGUE ALL and on DIALOGUE UNSEARCHED), and it continues to outperform CFS and GFS where RFW already had done so (on DIALOGUE SEARCHED). When the subset selected by RFW is already well distributed across

feature groups, RFW+ does not modify it further. The next chapter exploits three models of wizard error detection and recovery trained on features selected by RFW+ to improve CheckItOut.

Chapter 5

Improving the baseline system

This chapter brings together the data collection effort of Chapter 3 and the feature selection results of Chapter 4. Equipped with a rich corpus and a feature selection algorithm (RFW+) guided by domain knowledge, the work described in this chapter exploits them to improve the CheckItOut baseline system (described in Section 1.3). We modeled our best wizards from the full dialogue experiment (described in Section 3.2) to produce three models of error detection and recovery (identified in Section 4.1) from features selected by RFW+. Section 5.1 discusses the three wizard models trained on features selected by RFW+. Section 5.2 discusses how we modified CheckItOut to obtain a wizard-improved version, *CheckItOut+*. Section 5.3 evaluates and compares CheckItOut and CheckItOut+.

5.1 Wizard error detection and recovery models

Recall from our discussion in Section 3.2.2, that WA and WB were the most successful wizards; they had the highest number of correctly identified books per call. WA was persistent in the identification of any single book, with the most database queries and the most questions per book request. WA worked with the caller to identify the book: she asked questions informed by the context of the query results. When the context of the query was not trustworthy, WA led the caller to shift query

type by eliciting specific information (e.g., “Can I have the book number?”). WB, on the other hand, was goal-oriented. He also extensively queried the database in search of a worthy offer, but quickly moved on to the next book when any book request appeared to be problematic.

We combined the best of WA’s and WB’s error detection and recovery strategies into CheckItOut+, a wizard-improved version of CheckItOut. Recall from Section 4.1, that our wizards could make decisions at different stages of SLU, and that we trained three corresponding logistic regression models for three distinct SLU decision points. Figure 5.1, reproduced here from Figure 4.2, illustrates these models. Given WA’s ability to work with the caller to identify a book based on the context of the query results, we trained models A and C on data from calls with WA. Given WB’s ability to recognize when the ASR was too noisy, so that it was better to elicit new

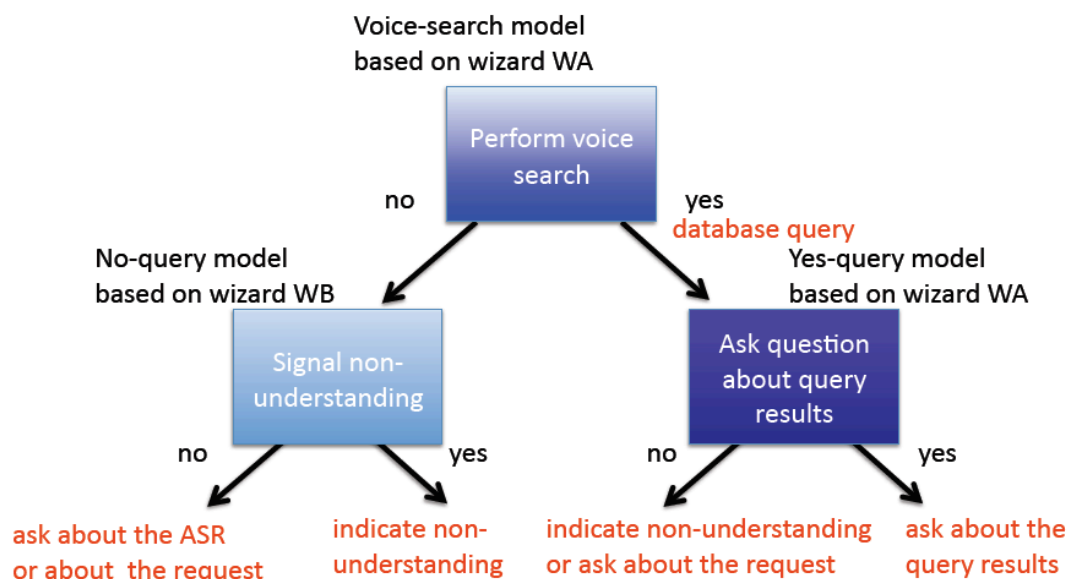


Figure 5.1: (Reproduced from Figure 4.2) Models of wizards’ decision making. The voice-search model decides whether to make a database query or not. The no-query model decides the next action to take given that there has been no database query. The yes-query model decides the next action to take given that there has been a database query. Darker blue indicates later stages in the SLU pipeline. Text in red indicates wizard actions predicted by the models.

ASR by indicating non-understanding or move on, we trained the no-query model on data from calls with WB. Section 5.2 discusses how we combined these three models into a unified error detection and recovery strategy for CheckItOut+. Because we want to simulate only our best wizards, to train only on data from WA and WB, we reduced the DIALOGUE ALL data set (voice-search model) from 16,956 instances for all wizards to 3,161 instances, the DIALOGUE UNSEARCHED data set (no-query model) from 10,901 instances to 714 instances, and the DIALOGUE SEARCHED dataset (yes-query model) from 6,055 instances to 1,159 instances. Thus the resulting models cannot be compared to those trained on data from all wizards in Chapter 4. We did not compare the effects of building models of WA or WB alone because our aim was to obtain a wizard-informed dialogue manager that could improve over CheckItOut (which we did), not to compare WA’s and WB’s strategies when built into a spoken dialogue system.

Table 5.1 provides accuracies and F measures for the three models. We used RFW+ to select features for each model. For each model, we ran RFW+ with 5 heuristic phases with initial subset sizes $R = \{3, 5, 10, 15, 20\}$, and evaluated accuracy with meta-groups, groups, and sub-groups. For the voice-search model, RFW+5 with groups achieved the greatest accuracy and included 11 features. For the no-query model, RFW+3 with groups achieved the greatest accuracy again with 11

Table 5.1: Performance of the three logistic regression models of wizard error detection and recovery trained on features selected by RFW+.

model	# features	# instances	%correct	κ	F
A	11	3161	84.81	0.67	0.88
B	11	714	69.38	0.36	0.75
C	5	1159	67.70	0.34	0.71

features. For the yes-query model, RFW+3 with sub-groups achieved the greatest accuracy with 5 features. Table 5.2 summarizes the features in each model. Features in boldface in Table 5.2 were in the initial subset S that was passed to the heuristic phase of RFW+.

Table 5.2: Features selected by RFW+ for three models of wizard error detection and recovery. Features in boldface are those initially passed to RFW+'s heuristic phase.

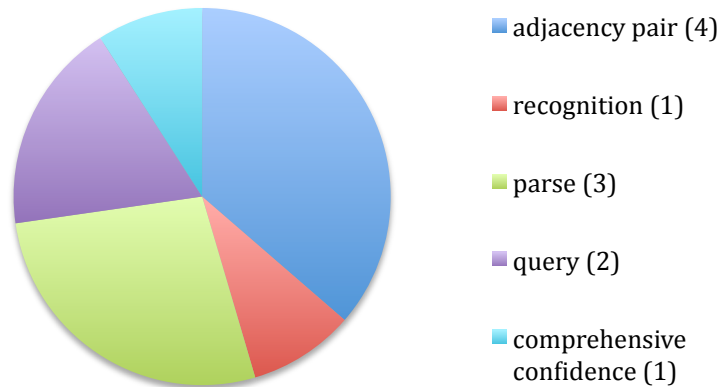
group/sub-group	feature
voice-search model RFW+ 5 (11 features)	
adjacency pair	Number of adjacency pairs in this book request Number of questions asked in this call Whether the prompt that initiated this adjacency pair indicates a non-understanding Whether the prompt that initiated this adjacency pair is an explicit confirmation
recognition	Average word confidence _1
parse	Number of words covered by the best parse _1 Top grammar slot in the best parse tree _1 Number of parses for this recognition hypothesis _1
query	Total number of queries in this book request Number of title queries in this book request
confidence	Average combined confidence A in this book request _1 (see Table 3.14)
no-query model RFW+ 3 (11 features)	
adjacency pair	Number of user utterances in this adjacency pair Number of partial explicit confirmations in this book request Whether the prompt that initiated this adjacency pair is an explicit confirmation
recognition	Average word confidence _1 Maximum word confidence _1 Acoustic model score _1
parse	Whether there was at least one title slot in the best parse _1 Number of words not covered by the best parse _1
query	Number of author queries in this book request Number of title queries in this book request
confidence	Helios confidence _1
yes-query model RFW+ 3 (5 features)	
adjacency pair history	Number of move ons in this call
adjacency pair	Whether a new book request was initiated
recognition history	Average acoustic model score in this call
query	Average R/O score R/O score standard deviation

For the yes-query model, RFW ranked all the query features at the top. RFW+ removed one of the three top-ranked features, and added three more from three underrepresented sub-groups. The removed feature described the maximum R/O score of the candidates returned by the query. After it added three more features, RFW+ could not improve accuracy with any other feature from the least represented sub-group, and therefore halted with five features.

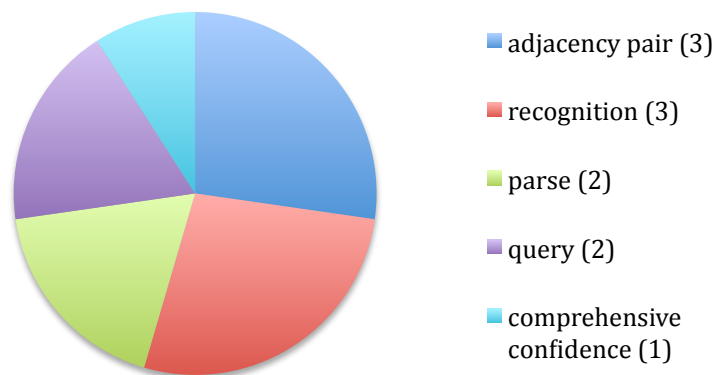
Figure 5.2 illustrates the distribution of the features selected by RFW+ for each model by category. RFW+ for the voice-search and no-query models used categorization by groups, and all 5 groups were represented in the final set of 11 features. For the voice-search model, RFW ranked at the top features from the adjacency pair, parse and query groups, and then RFW+ added features distributed across all five groups. For the no-query model, RFW ranked at the top features from the recognition and query groups, and again RFW+ distributed the remainder across groups.

RFW+ for the yes-query model used sub-groups, and only four sub-groups are represented in the 5-feature subset. Nonetheless, this relatively narrow subset had the highest accuracy of any RFW+ model for this data set. The feature subset selected by RFW+ for the yes-query model clearly indicates how RFW+ can compensate for RFW's randomness. Each query feature is highly relevant to whether the query results match the ASR well, and RFW ranks all query features at the top. Taken all together, however, these features are redundant, while features from other sub-groups may still be relevant. RFW+ repaired the ranking by removing one of the query

Voice-search model



No-query model



Yes-query model

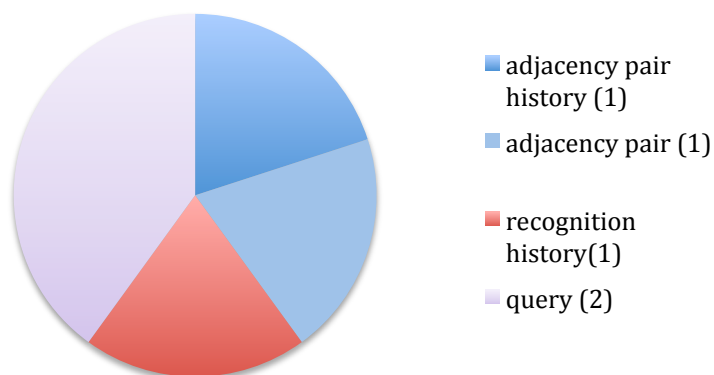


Figure 5.2: Distributions of features across groups (voice-search and no-query models) and sub-groups (yes-query model) for the three wizard models of error detection and recovery.

features initially selected in the three-feature subset, and added features from three of the nine other sub-groups.

5.2 Modifications to CheckItOut

To produce CheckItOut+, we incorporated the three wizard models trained on features selected by RFW+ into CheckItOut. We replaced CheckItOut’s dialogue manager with a new, wizard-guided one. When the ASR has been parsed well and concepts are bound, CheckItOut’s task tree (in Appendix B) suffices to advance the dialogue. Therefore, CheckItOut+ behaves just like CheckItOut as long as the parse is deemed confident by the Helios confidence annotator and a concept can be bound. (See Section 1.3 for details on Helios.) Figure 5.3 illustrates the dialogue managers for CheckItOut and CheckItOut+. The left side of both figures executes the task tree, which is identical in both systems. The agent manager executes the next discourse agent on the agent stack, which enforces depth-first traversal of the task tree. A discourse agent can issue a prompt and bind a concept, optionally as a result of a database query.

When the parse is not confident, as specified by Helios, or when a concept cannot be bound, CheckItOut’s error-handling module, on the right at the top in Figure 5.3, indicates that there has been a non-understanding. The error-handling manager selects the error-handling agent to be executed as a function of the number of consecutive non-understandings. Different error-handling agents issue prompts with different degrees of direction to the caller. After three to at most five consecutive

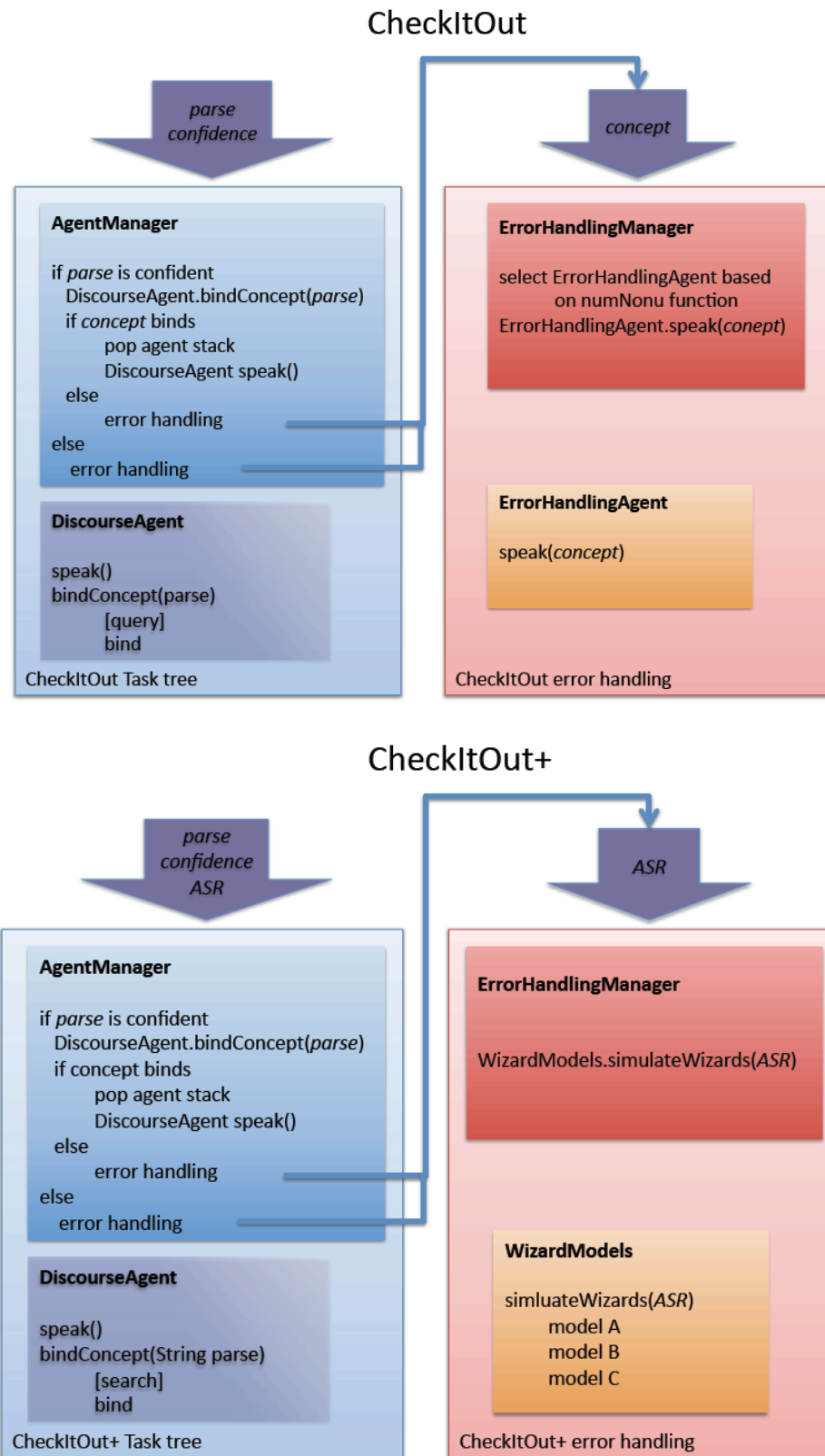


Figure 5.3: The dialogue managers in CheckItOut and CheckItOut+. Task tree execution is on the left and error handling is on the right. CheckItOut+'s task tree execution is identical to CheckItOut's.

non-understandings, CheckItOut will tell the caller that it is having too much difficulty with the call, and simply hang up. This behavior frustrates the callers and can be repaired by wizard models of error detection and recovery.

CheckItOut+ replaces CheckItOut's error handling with the wizard models to decide what to do when the parse is not confident or when a concept could not be bound. The error-handling manager in CheckItOut+ executes the three wizard models instead. Figure 5.4 provides pseudocode for the *wizard module* that combines the three wizard models learned from empirical data and implemented in CheckItOut+. If the voice-search model indicates voice search (line 1), CheckItOut+'s error handling will perform voice search and then execute the yes-query model (line 3).

In our data, wizards make decisions whether or not the ASR could be parsed well. In CheckItOut+, however, the wizard models are invoked only when the ASR is not well parsed. Although the models are also trained on instances where the ASR could

simulateWizards(ASR)

1. **if** voice-search model indicates voice search
 2. **then** VoiceSearch(ASR)
 3. **if** yes-query model responds positively to the query results
 4. **then** suggest the top ranked candidate
 5. **else** elicit specific information
 6. **else if** no-query model indicates non-understanding
 7. **then if** there have been fewer than 3 consecutive
 non- understandings
 8. **then** indicate non-understanding
 9. **else** move on
 10. **else** elicit specific information
-

Figure 5.4: Pseudocode for the wizard module that combines the three wizard models in CheckItOut+.

be parsed, our results (discussed next) indicate that this does not hurt the models' performance, and that the wizard models can lead to the successful identification of a book.

In CheckItOut+, voice search queries the database three times with the ASR, once each for title, author, and catalogue number. Then it selects the returned candidate with the highest R/O score. (It breaks ties by selecting the query with lowest R/O score standard deviation.) Based on the decision of the yes-query model, CheckItOut+ either proposes the top candidate from the voice search results (e.g., "Would you like *Pride and Prejudice* by Jane Austen?"), or elicits specific information from the caller (lines 4 - 5). Recall from the discussion in Section 3.2.2, that when WA was unconfident in the voice search results, she would elicit specific information from the caller to change the type of book request. Following WA's strategy, CheckItOut+ elicits a book request by a request type not just tried by the caller, with preference for catalogue number. For example, if voice search for titles had the highest R/O score, it will ask for the catalogue number 60% of the time and for the author 40% of the time. We chose a 40% - 60% split to emulate WA's observed slight preference for requests by catalogue number.

If the voice-search model determines that there should be no voice search, CheckItOut+ will execute the no-query model to determine whether a non-understanding has occurred (Figure 4.5, line 6). If so, CheckItOut+ will execute a non-understanding policy similar to CheckItOut's (line 7). Rather than prematurely end the call after several consecutive non-understandings like CheckItOut, CheckItOut+ moves on instead, as WB would do (line 9). (Recall that the caller may

return to that book request later in the call.) If the no-query model indicates that there has not been a non-understanding, again following WA's strategy to elicit specific information, CheckItOut+ will either ask for the catalogue number or inquire whether the caller has just asked for an author (line 10).

Figures 5.5 and 5.6 provide examples of sub-dialogues with CheckItOut+ in which the wizard module triggers. In both figures, red ASR could not be parsed confidently, and blue prompts are issued by the wizard module. Figure 5.5 illustrates successful examples of voice search. In the first example, the ASR .DUDLEY. SAFARI could not be parsed confidently (line 2). Instead of asking the caller to repeat, as

Example 1

- 1 **CheckItOut+:** Ok, let's order some books. You can ask for a title, author or book number. If I get it wrong just say: that is not what I said.
- 2 **Caller:** .DUDLEY. SAFARI
- 3 **CheckItOut+:** *Is the title *Deadly safari*?*
- 4 **Caller:** YES
- 5 **CheckItOut+:** Got it! What's the next book?

Example 2

- 1 **CheckItOut+:** What's the next book?
 - 2 **Caller:** .FOR. .NONDRIVERS. .SCHULLER. CHAPMAN
 - 3 **CheckItOut+:** *Is the author Fern Schumer Chapman?*
 - 4 **Caller:** YES
 - 5 **CheckItOut+:** *Would you like *Motherland beyond the holocaust a mother-daughter journey to reclaim the past* by FERN SCHUMER CHAPMAN?*
 - 6 **Caller:** YES
-

Figure 5.5: Two sub-dialogues with CheckItOut+ in which a book is identified from a wizard offer of a voice search candidate. Prompts in blue are issued by the wizard module. ASR in red could not be parsed confidently.

Example 1

- 1 **CheckItOut+:** What's the next book?
- 2 **Caller:** DUST .TO. DUNE
- 3 **CheckItOut+:** *Is the title Dust to dust?*
- 4 **Caller:** NO
- 5 **CheckItOut+:** Sorry, I misunderstood. Can you repeat the title please?
- 6 **Caller:** GOING THE .OTHER. ARETHA THAT
- 7 **CheckItOut+:** *Can you please give me the catalogue number?*
- 8 **Caller:** THREE ONE NINE NINE EIGHT
- 9 **CheckItOut+:** *Gorbachev heretic of the Kremlin, got it! What's the next book?*

Example 2

- 1 **CheckItOut+:** Sorry, I misunderstood. Can you repeat the author please?
 - 2 **Caller:** .ANDRES. MORTIMER
 - 3 **CheckItOut+:** Did you say John Mortimer?
 - 4 **Caller:** NO
 - 5 **CheckItOut+:** Sorry, I misunderstood. Can you repeat the author?
 - 6 **Caller:** .ANDREW. .VACHSS. NO DIDN'T
 - 7 **CheckItOut+:** *Can you give me the book title please?*
 - 8 **Caller:** DIANA HER TRUE STORY IN ROME
 - 9 **CheckItOut+:** *Diana her true story in her own words, got it! What's the next book?*
-

Figure 5.6: Two sub-dialogues with CheckItOut+ in which a book is identified after the wizard module elicited a change in request type from the caller. Prompts in blue are issued by the wizard module. ASR in red could not be confidently parsed.

CheckItOut would have done, CheckItOut+'s error handling executed the wizard module, which performed voice search on the ASR. As a result of voice search, the wizard module offered the top-ranked candidate title (line 3), which was indeed the correct title. In the second example of Figure 5.5, the ASR .FOR. .NONDRIVERS. .SCHULLER. CHAPMAN could not be parsed confidently (line 2), and the wizard module

offered the correct author from among the voice search candidates (line 3). Upon confirmation of the author (line 4), the task tree reclaimed execution and offered a book by that author (line 5). Here again, CheckItOut would have simply asked the caller to repeat the book request.

Figure 5.6 illustrates successful examples of a change in request type elicited by the wizard module. In the first example of Figure 5.6, the wizard module first attempted voice search on titles, but could not identify the correct book (lines 2 – 4). The second time the ASR could not be parsed (line 6), the wizard module asked for the catalogue number instead (line 7), and identified the book. CheckItOut+ gives the caller a sense that it is working with her to identify the book. CheckItOut, in this case, would have asked the caller to repeat both times. In the second example of Figure 5.6, a few consecutive failed attempts to identify the author (lines 1 – 6) are followed by the wizard module's request for the title (line 7). This change in request type was successful, and the ASR for the title could be parsed well (line 8). Instead, CheckItOut would have simply asked the caller to repeat even though it had already misunderstood the author twice in a row.

5.3 Evaluation

5.3.1 Data collection with CheckItOut and CheckItOut+

To evaluate CheckItOut+ and to establish a baseline with CheckItOut, two separate data collections were conducted with different subjects, except for one caller who participated in both. In both data collections, 10 callers, 5 male and 5 female, were instructed to make 50 calls each. All subjects were recruited through email and flyers at Hunter College and Columbia University and received monetary compensation. Both systems used the same language model as that used in the full-dialogue data collection of Section 3.2. As in the full-dialogue wizard experiment, callers accessed a web page to obtain a scenario with a patron identity and four books to order. Here too, books in a scenario were randomly selected, and callers were instructed to distribute the request types evenly. Callers were surveyed online after the first, 23rd, and 47th calls. All caller surveys were the same; they elicited user satisfaction measures and allowed the callers to make comments. For the baseline with CheckItOut, 562 dialogues were collected, in spite of the instructions to make 50 calls each. With CheckItOut+, 505 dialogues were collected in all. All differences reported in the next section are statistically significant ($p < 0.0001$) unless otherwise noted.

5.3.2 Comparative results

Results indicate that CheckItOut+ was more successful at ordering four books than CheckItOut was. As discussed in the Section 5.2, when CheckItOut incurs a series of

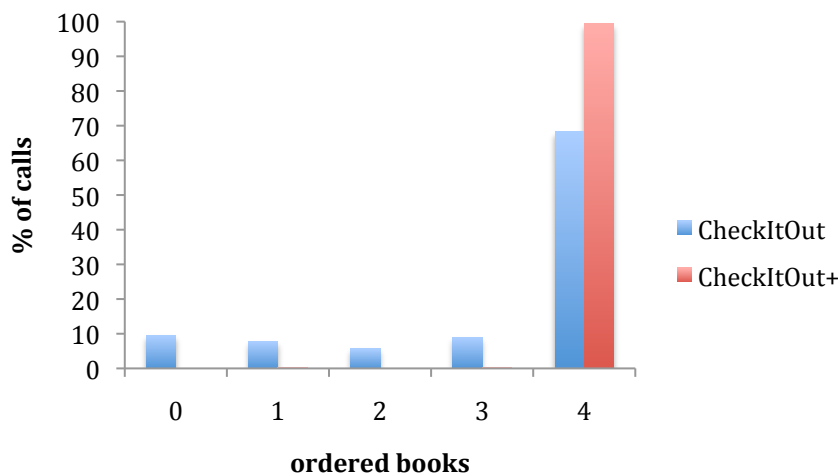


Figure 5.7: Percent of calls in which the two systems ordered zero to four books.

consecutive non-understandings, eventually it exhausts its error handling strategies and abandons the call. CheckItOut+'s richer error handling strategy does not do so. Figure 5.7 illustrates how often a call to each system ordered up to four books. In 502 of its 505 calls, CheckItOut+ completed its task, that is, it ordered four books.

An ordered book is not always correct, however. In both systems, if the ASR parses to a title or to a catalogue number, the matching book is ordered and implicitly confirmed (e.g., “*Pride and Prejudice*, got it. What’s the next book?”) If the book

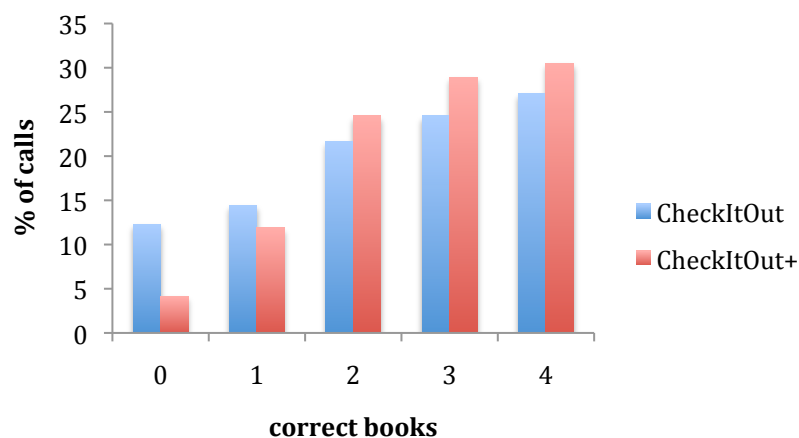


Figure 5.8: Percent of calls in which the two systems ordered zero to four correct books.

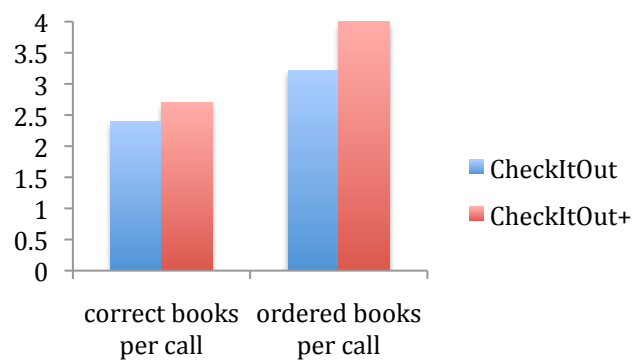


Figure 5.9: Average number of correct books and ordered books per call with the two systems.

was not correct, the caller had been instructed to signal the misunderstanding with “That is not what I said.” If the system correctly parses the signal from that speech, it acknowledges its mistake and removes the book from the order. If, however, the caller’s reply to the misunderstanding is not parsed, the book remains ordered incorrectly. Figure 5.8 illustrates how often a call to each system ordered up to four correct books. CheckItOut ordered zero or one correct book more often; CheckItOut+ ordered two, three and four correct books more often. Figure 5.9 illustrates the average number of correct books and ordered books per call.

Misunderstandings can be a problem for CheckItOut+. When the caller’s reply to a misunderstanding cannot be parsed, CheckItOut’s strategy is to signal non-

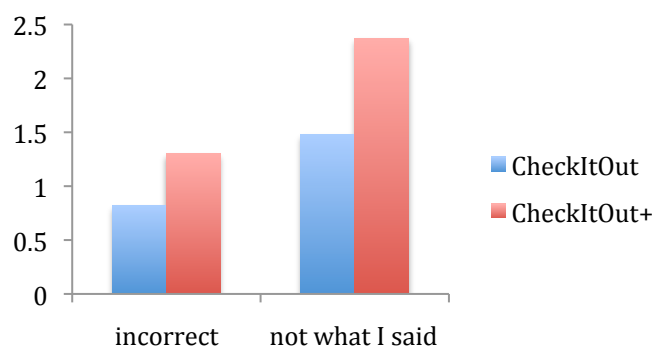


Figure 5.10: Average number of incorrect books and caller-repair attempts (number of times the caller said “that’s not what I said”) per call with the two systems.

understanding, and the caller may then repeat “That’s not what I said.” With CheckItOut+, however, if the wizard module decides to perform voice search, it will mistake the unparsed noisy ASR for a book request, and the caller does not get a second chance to repair the original misunderstanding. In this situation, often callers repeated “That’s not what I said” a few more times, but the system was already processing a new book request and the repair attempt on the previous book failed. CheckItOut+, therefore, incurred more caller-repair attempts, as illustrated in Figure 5.10. Recall too that CheckItOut+ ordered more books per call, and therefore ordered a larger number of incorrect books per call on average, also illustrated in Figure 5.10. Henceforward, $\mu (\pm \sigma)$ denotes data with mean μ and standard deviation σ .

Figure 5.11 illustrates how often the wizard module triggered in CheckItOut+. Figure 5.12 depicts a simplified execution flow in CheckItOut+’s dialogue manager until a book is ordered. The wizard module in CheckItOut+ triggered $4.72 (\pm 3.94)$ times per call. Of these, $3.76 (\pm 3.35)$ resulted in a voice search candidate offer, and

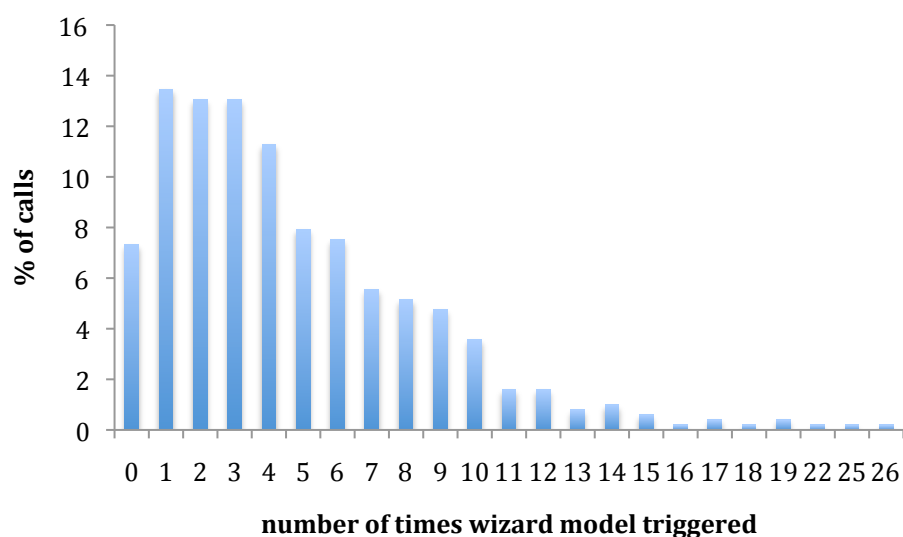


Figure 5.11: Percent of calls in which the wizard module triggered from 0 to its maximum of 26 times.

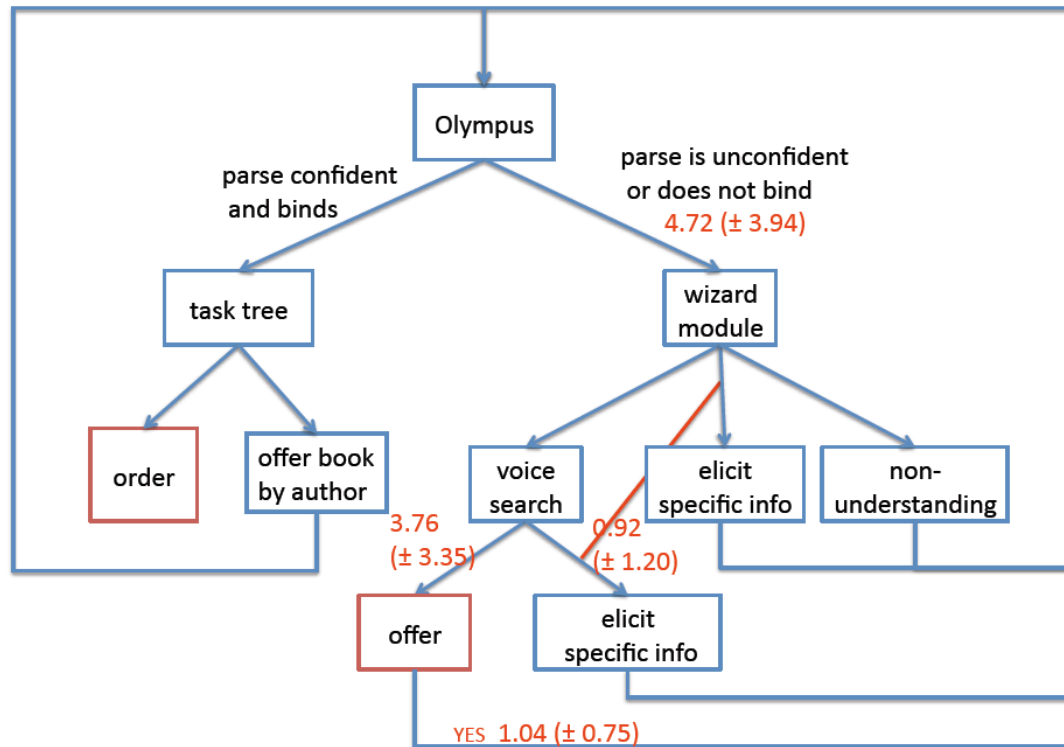


Figure 5.12: A simplified execution flow of CheckItOut+'s dialogue manager up until a book is ordered. Arrows are annotated with the number of times they are executed in each call, reported as $\mu (\pm \sigma)$, where μ is the mean and σ the standard deviation.

1.04 (± 0.75) of these wizard-initiated offers was confirmed by the caller as correct. The wizard module was, therefore, responsible for at least 38% (1.04 / 2.70) of the books correctly ordered by CheckItOut+. The wizard module elicited a different request type 0.92 (± 1.20) times per call. The response so elicited was then pipelined back into the system and could have been handled by either the task tree or the wizard module once again to identify a book. In this way, the wizard module could have been indirectly responsible for more than 38% of all correctly ordered books.

Bear in mind that each time the wizard module triggered in CheckItOut+, CheckItOut would have either asked the caller to repeat or have given up on the call. The fact that the offer of a voice search candidate was correct only about one time in

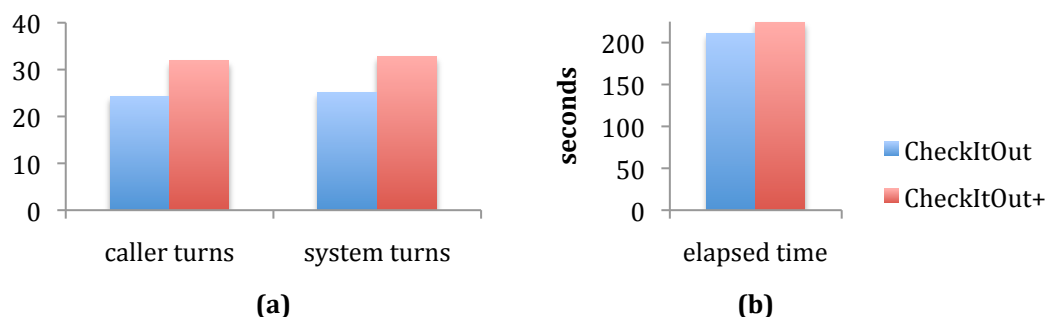


Figure 5.13: Average number of caller turns and system turns per call a), and number of seconds per call b) with the two systems.

four is not a fatal flaw. The dialogue continued constructively, and the caller had the impression that the system was working with her to identify the book. (See, for example, Example 1 in Figure 5.6.)

The wizard module models WA's collaborative approach: it works with the caller to identify the book despite the noisy ASR. CheckItOut+ either offers a voice search candidate to the caller, or asks for specific information (e.g., the book's author or catalogue number) to change the book request type, with the expectation that it will elicit better ASR. Moreover, during the experiment, CheckItOut+ never prematurely ended a call, almost always ordered four books, and ordered more correct books on average. For all these reasons, dialogues with CheckItOut+ generally incurred higher overall costs. Nonetheless, CheckItOut+ was also able to order more books and interpret utterances that would have otherwise gone unparsed.

Figure 5.13(a) illustrates the number of caller and system turns per call, and Figure 5.13(b) illustrates the time per call in seconds. Given that CheckItOut+ ordered more books, the difference in elapsed time ($p < 0.0175$) is notably low, only 13 additional seconds on average. Moreover, if we consider the number of ordered books and the number of correct books, CheckItOut+ had better throughput and

identified correct books faster. CheckItOut+ took 56.01 seconds per ordered book on average, compared to CheckItOut's 65.57. CheckItOut+ took 82.95 seconds per correct book on average, compared to CheckItOut's 87.89.

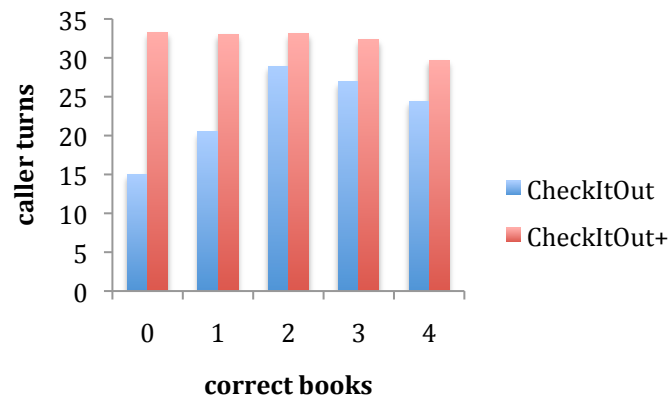


Figure 5.14: Average number of caller turns in calls with zero to four correct books.

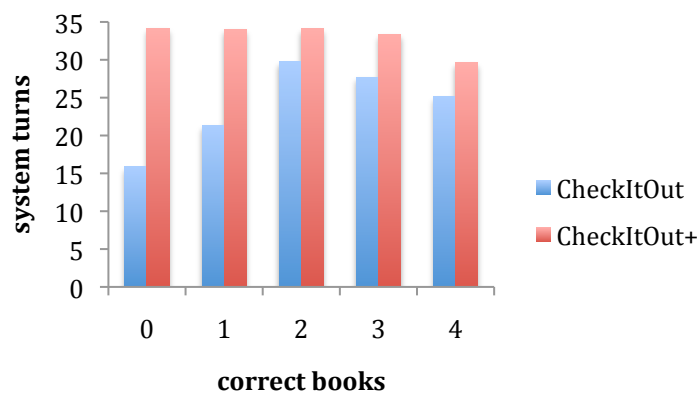


Figure 5.15: Average number system turns in calls with zero to four correct books.

Figures 5.14 and 5.15 break down caller turns and system turns across calls with zero to four correct books. These figures illustrate that CheckItOut+ worked equally hard to identify books even when the ASR was so poor that no or few books were ordered correctly. Callers exchanged a comparable number of turns with CheckItOut+, regardless of the number of books correctly ordered because the system strove to identify all four books. CheckItOut, instead, simply abandoned the call. Bear in mind, however, that, as illustrated earlier in Figure 5.8, CheckItOut+ identified zero or one correct book far less often than CheckItOut did. On calls with better ASR quality, more correct books were ordered and costs, such as the number of system and user turns per ordered book, decreased.

When misunderstandings occur, more than four book requests can be received in a call. Figure 5.16 illustrates that the two systems processed on average the same number of book requests per call ($p = 0.8904$). This also explains the relatively small difference in overall elapsed time. Figure 5.16 also illustrates the distribution of book requests by request type. Recall that callers had been instructed to distribute their

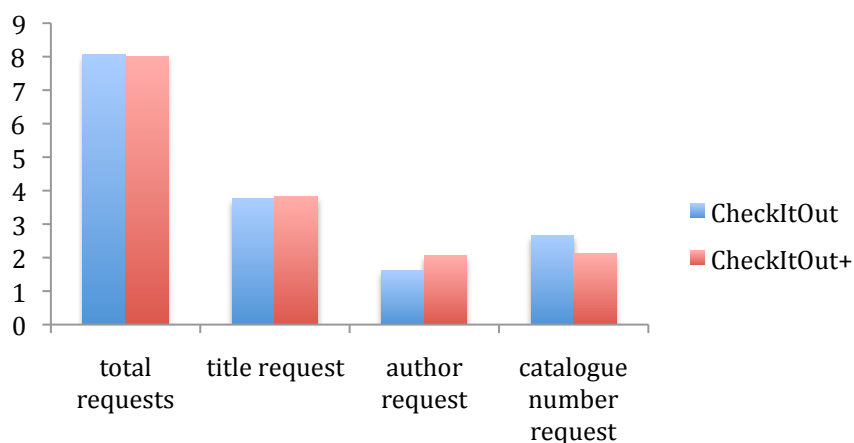


Figure 5.16: Total number of book requests per call, and their distribution by request type for both systems.

requests evenly across request types: one by title, one by author, one by catalogue number, and one by any of the three types. The two systems processed, on average, the same number of requests by title ($p = 0.6485$). CheckItOut+'s overall task was actually more difficult, because it encountered more requests by author and fewer requests by catalogue number. (Catalogue numbers are generally better recognized than author names are.) Author requests are always explicitly confirmed in both systems (e.g., "Did you say Jane Austen?") and require an additional user turn and an additional system turn for the explicit confirmation. Author requests also incur higher cost because of the multiple books offers for any single author request. Each author offer requires at least an additional system turn and an additional user turn. CheckItOut+ made 8.33 such author offers per call on average, against CheckItOut's 4.06.

Figure 5.17 breaks down book requests across calls with zero to four correct books. On calls with zero or one correct book, CheckItOut+ processed more book requests because it still tried to order all four books, while CheckItOut simply

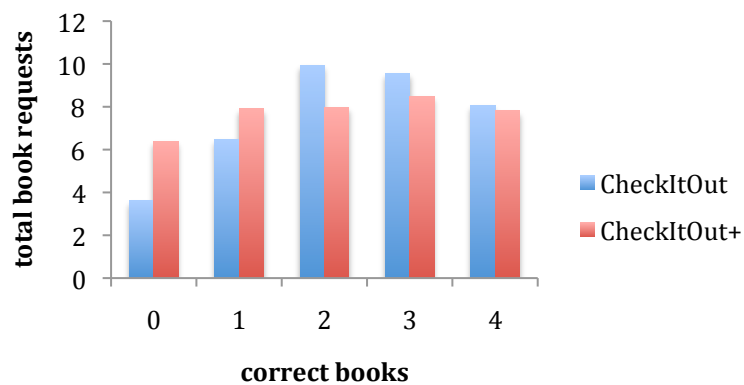


Figure 5.17: Average number of book requests in calls with zero to four correct books.

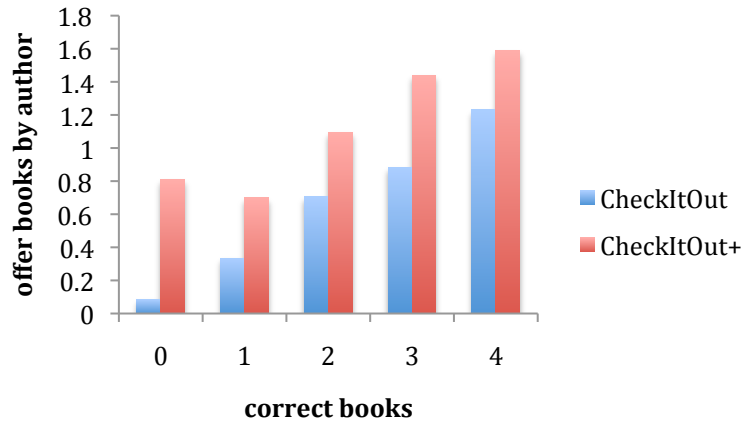


Figure 5.18: Average number of book offers by author in calls with zero to four correct books.

abandoned the call. On calls with two or three correct books, CheckItOut processed more book requests because it incurred more misunderstandings with its simple strategy: ask the caller to repeat. CheckItOut+ received explicit confirmations to voice search offers instead. Recall that CheckItOut had fewer calls with two, three, and four correct books, and therefore incurred fewer misunderstandings overall, as per the earlier discussion of Figure 5.10. Figure 5.18 breaks down book offers by author across calls with zero to four correct books. As previously discussed, CheckItOut+ processed more requests by author, and therefore it also offered more books by author for every number of correct books. Callers were more comfortable with CheckItOut+'s improved ability to identify books and resorted less often to the better-recognized catalogue number requests, which they used with CheckItOut.

CheckItOut+ engages the caller; it offers books from the voice-search results or asks questions (e.g., “Who is the author of this book?”), rather than ask the caller to repeat as CheckItOut does. CheckItOut+ rarely *offers direction*, that is, rarely dictates what the caller should say. CheckItOut, in contrast, gradually increases the amount of

direction it provides as consecutive non-understandings accumulate (e.g., “Would you like *Pride and Prejudice* by Jane Austen? Please say yes or no.”) CheckItOut offered direction 1.75 times per call on average; CheckItOut+ did so 0.07 times.

Caller satisfaction was gauged through the caller surveys. Recall that three surveys were administered to each caller during the data collection. Table 5.3 lists the questions in the surveys. Callers were automatically redirected from the scenario webpage to the survey. Despite instructions to take all three surveys, we collected only 25 of the 30 surveys with CheckItOut, and 29 surveys with CheckItOut+. Callers checked answers on a Likert scale, converted for analysis to numbers from one to five. The remainder of this discussion references questions by their labels in Table 5.3, and compares the callers’ answers to survey questions for the two systems. Given the small number of data points, however, an unpaired t test averaged over all surveys indicated a statistically significant difference ($p < 0.01$) only for question D. Given our experimental conditions, more than three surveys would have not been well tolerated by the callers. A study with fewer calls per session and greater

Table 5.3: Survey questions to elicit caller satisfaction measures.

Label	Question
A	The system responded correctly
B	I found the system’s voice easy to understand
C	The system was friendly
D	I had to pay close attention while using the system
E	The system made the task easy for me
F	I found talking with the system annoying
G	The system interacted with me in an irritating way
H	It was clear to me what the system could do
I	It was clear to me what the system could not do
J	The system was helpful
K	I would use a system like this to borrow library books by phone

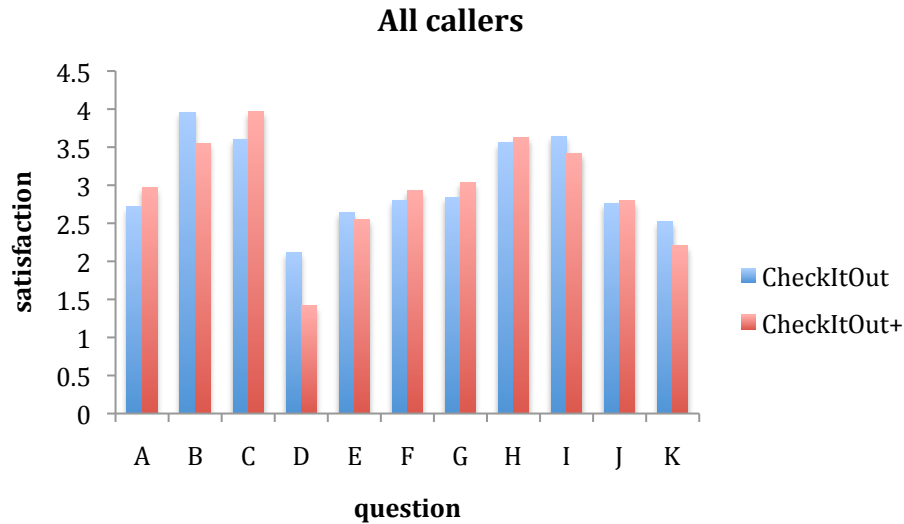


Figure 5.19: Satisfaction metrics from answers to the questions in Table 5.3. Satisfaction is averaged over all surveys from all callers.

compensation could have requested more surveys and, perhaps, observed significant differences across questions. Figure 5.19 shows the callers' responses as numeric values averaged across all surveys by all callers. For questions D, F, and G, the scale was inverted so that, for all questions, higher numbers indicate greater satisfaction.

The only statistically significant difference indicates that callers found that they had to pay less attention with CheckItOut than they did with CheckItOut+ (question D). This is probably due to CheckItOut+'s greater involvement with the caller through questions. CheckItOut+ engages the caller in a collaborative process to identify books and overcome noisy ASR. This requires that callers pay more careful attention to the system's requests. Dialogues with CheckItOut are straightforward; it asks for the next book, and it asks the caller to repeat if it does not understand. Dialogues with CheckItOut+ are more complex, and involve a greater variety of questions.

5.3.3 A unified evaluation

To analyze which cost and success measures most influenced user satisfaction, we used PARADISE (PARAdigm for Dialogue System Evaluation), a general framework for the evaluation and comparison of spoken dialogue systems' performance (Walker, et al. 1997). PARADISE postulates that overall performance can be derived as a weighted linear combination of task-success measures and dialogue-cost measures, with user satisfaction as the dependent variable. Table 5.4 summarizes the cost and success measures collected for the two systems, labels them for reference in the remainder of this discussion, and reports their average and standard deviation.

Taken all together, responses to the questions in Table 5.3 provided a reliable scale ($\alpha = 0.97$) (Cronbach 1951). Therefore we averaged each caller's answer to them to gauge an *overall satisfaction* measure to use as the dependent variable. Given the small number of data points, we selected the cost and success measures that, once normalized, had statistically significant correlation ($p = 0.001$) with overall satisfaction. There were six such measures for CheckItOut (CORR, ORD, DIR, UTT, REQ, TSLOT), and three for CheckItOut+ (CORR, INC, REQ). Then we derived from a stepwise linear regression a *performance equation* for each system to predict overall satisfaction from the highly-correlated measures whose inclusion produced a significant additional explanation to the variance ($p = 0.001$). Those equations are reported in Table 5.5.

Table 5.4: Task-success and dialogue-cost measures per call, over all calls for the two systems with mean μ and standard deviation σ .

Task-success measures		CheckItOut	CheckItOut+
		$\mu \pm \sigma$	$\mu \pm \sigma$
Correctly ordered books	CORR	2.40 \pm 1.34	2.70 \pm 1.14
Ordered books	ORD	3.22 \pm 1.35	4.00 \pm 0.17
Dialogue-cost measures			
Incorrectly ordered books	INC	0.82 \pm 1.03	1.30 \pm 1.14
Caller-repair attempts (i.e. “that is not what I said”)	REP	1.48 \pm 2.24	2.37 \pm 2.53
User turns	USRT	24.28 \pm 11.44	31.85 \pm 14.19
System turns	SYST	25.08 \pm 11.42	32.84 \pm 14.20
Distinct user utterances (more than one utterance in the same user turn)	UTT	25.16 \pm 11.90	31.85 \pm 14.19
Distinct system prompts (more than one prompt in the same system turn)	PRMPT	41.57 \pm 20.76	38.75 \pm 16.46
Elapsed time	TIME	210.93 \pm 103.44	223.96 \pm 100.58
Title offers by author	AUTOFF	0.76 \pm 1.08	1.29 \pm 1.41
Direction offers	DIR	1.75 \pm 1.80	0.07 \pm 0.29
Book identifications (e.g., “ <i>Pride and Prejudice</i> , got it!”)	ID	4.06 \pm 2.53	8.33 \pm 3.87
Time outs (the system hangs up because there is no caller response)	TIMEOUT	0.02 \pm 0.39	0.00 \pm 0.00
Total book requests	REQ	8.05 \pm 4.86	8.00 \pm 3.87
Title requests	TITREQ	3.76 \pm 3.19	3.82 \pm 2.46
Author requests	AUTREQ	1.62 \pm 1.75	2.05 \pm 1.94
Catalogue number requests	NUMREQ	2.67 \pm 1.91	2.12 \pm 1.70
Title slots in the parses	TSLOT	4.12 \pm 3.55	4.03 \pm 2.78
Author slots in the parses	ASLOT	1.74 \pm 1.87	2.11 \pm 2.03
Catalogue number slots in the parses	NSLOT	2.74 \pm 2.05	2.59 \pm 1.70

Table 5.5: Performance equations on cost and success measures for CheckItOut (Ch) and CheckItOut+ (Ch+) on data from all callers.

System	Performance equation	R ²
Ch	0.102 (CORR) – 0.074 (DIR) + 2.837	0.1455
Ch+	-0.091 (INC) + 3.052	0.0867

Inspection of the performance equations indicates that for CheckItOut the predominant factors that influenced satisfaction were a task success measure (the number of correctly ordered books, CORR) and a cost measure (the amount of direction provided by the system, DIR). For CheckItOut+, the predominant factor was a success measure (the number of incorrectly ordered books, INC). (Recall that the amount of direction in CheckItOut increased with the number of consecutive non-understandings.) These performance equations suggest that task success influenced caller satisfaction, and that cost also had an impact in CheckItOut, albeit a smaller one. Callers to both systems cared about the number of books correctly (or incorrectly) ordered.

We offer another way to understand user satisfaction, not based on PARADISE, but on the nature of the questions themselves. Although some were statistically significant, the magnitude of the correlations of cost and success measures to user satisfaction was not great. Originally question K in the caller surveys was intended to elicit an overall satisfaction measure (i.e., whether the callers would use the system or not). Question K did not significantly correlate with any CheckItOut+ cost or success measures. Nonetheless, survey questions A through J did correlate well with the overall satisfaction metric elicited by question K. Questions C, F, and G described the callers' emotions (how they felt about the system). Questions A, B, D, E, H, I, and J

Table 5.6: Performance equations on caller experience and emotion measures for CheckItOut (Ch) and CheckItOut+ (Ch+) on data from all callers.

System	Performance equation	R ²
Ch	2.094 (EXP) + 0.231 (EM) - 4.553	0.6397
Ch+	1.836 (EXP) - 0.583 (EM) - 1.208	0.5203

described the callers' experiences (how they interacted with the system). We averaged the answers to these two kinds of questions (EM and EXP, respectively) and modeled their relative influence on overall satisfaction, as measured by question K. This time, stepwise regression produced the equations in Table 5.6. The callers' answers to questions grouped as EXP drove callers' overall satisfaction more than their answers to question grouped as EM did. As one would expect, both were significantly correlated with overall satisfaction ($p = 0.001$). For CheckItOut, the highest correlation with question K among the cost and success measures was that of CORR = 0.0304. The correlations of EXP and EM with question K were 0.7908 and 0.5187, respectively. For CheckItOut+ the highest correlation among the cost and success measures was ASLOT = -0.0661. The correlations of EXP and EM with question K were 0.6764 and 0.1594, respectively.

In summary, CheckItOut+ achieved greater task success, with more ordered books and more correct books per call. On a per-book basis, CheckItOut+ also had higher throughput. Callers, however, were equally satisfied with the performance of both systems. They perceived the systems' responses to noisy ASR similarly, whether they were CheckItOut's simple requests to repeat followed by eventual hang-ups, or CheckItOut+'s collaborative process of questions and offers, even though the latter

achieved greater task success. Callers were generally overwhelmed by the level of noise, which is unusual in commercial spoken dialogue systems. Their perception of the system's difficulty in understanding, due to the high word error rate, prevented them from recognizing CheckItOut+'s increased task success, although task success did influence their level of satisfaction. The next chapter summarizes the thesis and contributions of this dissertation and discusses future work.

Chapter 6

Conclusion

Spoken dialogue between people and machines is desirable and increasingly popular. Dialogue interaction is natural to people: it requires no training, it is suitable to small devices, it facilitates multitasking, and it increases accessibility. Because spoken dialogue is easy for people, human expectations for such interaction are high. People expect to interrupt themselves, cough, ponder, and mumble, and still be understood. Although most commercial systems incur low speech recognition error, they conduct dialogue with little flexibility and minimal complexity. Next-generation spoken dialogue systems must contend with error, if they are to handle complex language and flexible dialogue interactions.

The thesis of this work is that feature selection is essential to data-driven error detection and recovery for robust and adaptive spoken dialogue systems.

In particular, this research investigates how feature selection can exploit knowledge about the provenance and interaction of system features and their relevance to different sources of error. The research described in this dissertation demonstrates that people have strategies they use in the presence of noise, and that those strategies are feasible for successful dialogue with noisy ASR. Such strategies can be learned from dialogue system features selected by RFW+, a knowledge-guided feature selection algorithm. When implemented within a spoken dialogue system, they improve task success.

6.1 Summary of contributions and future work

The principal contributions of this research include:

- An experiment that studied how people made decisions given the same input and output as a spoken dialogue system. This *Wizard of Oz* study presented a person with a speech recognition string, and gave her access to the book database and the ability to select the next prompt spoken by the system. It produced a large corpus of 913 dialogues to be released to the research community. The corpus was used to produce models of error detection and recovery.
- An experimental feature selection framework that runs comparison experiments between feature selection algorithms. The experiments compare the accuracy of models learned on different sets of features to determine feature selection quality.
- Two novel feature selection algorithms: RFW, a randomized weight learning algorithm, and RFW+, an extension of RFW guided by domain knowledge.
- The identification of sets of features for error detection and recovery at different SLU processing levels.
- Empirical results that demonstrate the feasibility of error detection and recovery strategies for successful dialogue despite noisy ASR.
- Empirical results that demonstrate how data-driven models of error detection and recovery strategies can improve overall task success.

6.1.1 Rapid tailored feature selection

RFW is a feature selection wrapper that learns feature weights through the random sampling and evaluation of feature subsets. Unlike filters, wrappers are tailored to a dataset and a learner, albeit with greater computational costs. Empirical results indicate that RFW generally performs as well as, and sometimes better than, the state-of-the-art baseline wrapper and is at least an order of magnitude faster. Empirical results also indicate that RFW generally performs as well as, and sometimes better than, the state-of-the-art baseline filter, but slower. To learn feature weights, RFW tests moderately-many random subsets of features and records how those subsets performed. Through random exploration and weight learning, RFW strikes a good balance between the desirable properties of filters and wrappers; it avoids the bias of a filter's pre-defined feature relevance metric, and it is faster than traditional wrappers. When the data is well understood and a filter selects good feature subsets, RFW is unnecessary. When more exploration of possible feature subsets is required, however, RFW adroitly balances computational cost and simplicity, while it proves consistently accurate and scalable.

As most approximation algorithms do, RFW requires parameter settings. We experimented with several values for the halting parameter *persistence*, and generally noticed no empirical difference in accuracy. Although a larger value sometimes produced an improvement over a baseline that a smaller value did not, it is unclear whether the improvement was a result of the more intensive exploration of possible feature subsets, or a byproduct of random exploration. To answer this question, future work will compare larger *persistence* sizes with n RFW runs that use $persistence/n$

and retain only the best ranking. An important quality of RFW is that it achieves accuracy comparable to search-intensive wrappers at low computational cost. The parameter *persistence* should preserve this property and support learning good feature weights, but avoid overfitting. Ultimately the parameter should be a function of the number of features in the data set, or the algorithm should halt when learning converges. Future work will also experiment with randomization over features as well as samples of the training set. Another open question is whether our empirical results with random subset sizes in $[min, max]$ hold only for data sets of sizes comparable to the ones on which we experimented. These results are, however, supported by similar results in a different domain (Petrovic and Epstein 2009).

An important issue not addressed by RFW is the parameter $|S_{top}|$, how many of the features top-ranked by RFW to include in the training of the final model. For experimental purposes, we arbitrarily tested a reasonable range of subset-size reductions for the data sets on which we experimented. We found that RFW produced a good ranking of the features and that subsets of top-ranked features of many sizes often trained classifiers with comparable performance. The size of the initial feature set in the data sets we tested ranged from 23 to 168 features (except for ISOLET's 617 features). On the reduced ISOLET data with C4.5, RFW performed as well as the baseline filter and wrapper, and with a larger *persistence* it outperformed the wrapper. On the full ISOLET data set, however, the wrapper did not complete in 24 hours, and the filter achieved better performance than it had on the reduced data set, but with a larger subset (192 features). Although the filter outperformed RFW, we had limited RFW to the same subset sizes on both the reduced and the full data sets, and

performance remained comparable. This suggests that the subset sizes tested for RFW may have been too small for ISOLET.

To determine the size of the final subset, in an additional evaluation phase, RFW could evaluate increasing subset sizes of top-ranked features, until accuracy significantly degrades, and then select the smaller subset with greatest accuracy. Alternatively, RFW could select the top-ranked features with higher accuracy and size closest to some user-specified preference. Such a preference allows the user to influence dimensionality reduction without making an arbitrary choice. As yet another alternative, when domain knowledge is available, it can be used to guide the selection of the final subset and determine its size. RFW+'s heuristic phase is, de facto, an additional evaluation phase that uses domain knowledge to determine the size of the final subset, given RFW's ranking F' , and a user-specified preference for a single subset size.

RFW+ extends RFW with knowledge about the provenance of features along the SLU pipeline. Under the assumption that some representative features of each SLU stage should participate in error detection and recovery, RFW+ can improve upon RFW on the dialogue data sets. It outperforms the filter and the wrapper on dialogue data sets where RFW did not, and it continues to outperform them on the dialogue data set where RFW already had done so. When the subset selected by RFW is already well distributed across feature categories, RFW+ does not modify it further. Empirical results show that, although the final subset size would be an issue, many different subset sizes of features highly weighted by RFW support equally strong

learning. A user can guess an approximate subset size, and let RFW+ refine RFW's top-ranked features to determine the size of the final subset.

On one dialogue data set, RFW+ did not achieve much improvement over RFW during testing, nor are the improvements it achieved during training over the initial set of features top-ranked by RFW extraordinary. Nonetheless, it never did worse. Recall that results with RFW illustrate that many different subsets of high-ranked features train classifiers of comparable performance. The differences in accuracy observed throughout our experiments are not large, although, where noted, they are often statistically significant. RFW+ can improve on RFW, but is not guaranteed to do so. Nonetheless, its ability to determine the size of the final subset is important. Moreover, RFW+ incurs little additional overhead, and therefore retains RFW's desirable property of producing a subset tailored to the data and the learner faster than a typical wrapper would.

Feature categorization is domain-dependent, and its granularity is an open question. The medium granularity of feature groups achieved the greatest improvement over the initial subset on all three data sets. The interdependence of the granularity of the categorization and the halting condition in RFW+ merits further investigation. Future work will explore a definition for least-represented category in proportion to the size of the category itself. RFW+ could also be further modified to select the least-represented category with roulette-wheel selection.

6.1.2 Wizard-derived expertise improves task success, task correctness, and throughput

Using features selected by RFW+, we derived three models of our best wizards' error detection and recovery actions. Two models were trained on data from one wizard, and the third model on data from another wizard. To train only on data from these two wizards, we reduced the data set sizes dramatically. As a result, the accuracy of the three models decreased somewhat from the accuracy of the models trained on data from all the wizards. On one data set, however, the accuracy of the model that decided whether to ask a question about the query results decreased dramatically. The wizard on whom it was trained was the most communicative and had asked the most questions per call. Her questioning behavior was more complex and less predictable than the more simple approaches of most other wizards. That questioning behavior was also the most effective, however, and therefore the one we chose to model. Exact reproduction of the wizard's actions was neither feasible nor essential; our goal was to capture her overall questioning strategy.

Our results indicate that CheckItOut+ was more successful at ordering books than CheckItOut was. CheckItOut+ almost always ordered four books, and it ordered more correct books than CheckItOut. On a per-book basis, CheckItOut+ also ordered books faster. Misunderstandings can be a problem for CheckItOut+, however. When the caller's reply to a misunderstanding could not be parsed, CheckItOut's strategy was to signal non-understanding, and the caller could then repeat "That's not what I said." With CheckItOut+, however, if the wizard module decided to perform voice search, it would mistake the unparsed noisy ASR for a book request, and the caller did not get a

second chance to repair the original misunderstanding. This behavior could be easily ameliorated through a voice search test of the incoming ASR against the string “That’s not what I said,” as well as against the book holdings. If the R/O score for the caller repair request were higher, CheckItOut+ could explicitly confirm whether the caller had actually just attempted a repair.

Furthermore, CheckItOut+ still relies on a model of discourse that assumes communication has occurred by the mere act of speaking. A more realistic model of discourse assumes that a contribution to the discourse has occurred only when it has been *grounded*, that is, both parties have established the mutual belief that the intent of the speaker was correctly perceived by the listener (Clark and Schaefer 1989).

A wizard offer from the voice search results is more constructive than a request to repeat, even if the offer is unsuccessful. For example, in the dialogue of Figure 6.1, the first author offer after voice search was unsuccessful. It was not out of place, however. An author offer gave the caller the impression that the system was at least aware that an author was intended. CheckItOut, instead, would have asked the caller to repeat herself twice.

CheckItOut+: What's the next book?
Caller: JEAN .FROISSART. ARE NINE
CheckItOut+: Is the author Jean Froissart?
Caller: NO
CheckItOut+: Sorry, I misunderstood. Can you repeat the author please?
Caller: ERIC JEROME DICKEY
CheckItOut+: did you say ERIC JEROME DICKEY?
Caller: DIDN'T
CheckItOut+: Sorry, I didn't catch that.
CheckItOut+: did you say ERIC JEROME DICKEY?
Caller: YES
CheckItOut+: Would you like *Between Lovers* by Eric Jerome Dickey?
Caller: NO
CheckItOut+: Would you like *Thieves Paradise* by Eric Jerome Dickey?
Caller: YES
CheckItOut+: What's the next book?

Figure 6.1: A sub-dialogue with CheckItOut+ with an unsuccessful offer from the voice search results. ASR in red could not be parsed confidently. Blue prompts were issued by the wizard module.

Although CheckItOut+ achieved greater task success with more ordered books, more correct books per call, and higher throughput on a per-book basis, it must still contend with an unusually high word error rate. CheckItOut+ demonstrates that it is possible to have successful dialogue, despite noisy ASR. Next-generation spoken dialogue systems should be able to handle large vocabularies, complex conversational topics, and a flexible style of interaction – all qualities that make the interaction prone to understanding error.

6.2 Concluding remarks

The book-ordering application is challenging, with its large vocabulary and the broad range of topics that book titles address. Moreover, both CheckItOut and CheckItOut+ intentionally handle high levels of noise. (We estimated WER at 50%.) A system that can be successful at its task with such high WER can only improve its performance as speech recognition technology addresses more complex dialogue interactions. Error detection and recovery do not stand in opposition to error prevention. It is, however, unlikely that speech technology can entirely prevent all errors in a system that participates in human-like dialogue interactions. Error detection and recovery are essential to the next generation of robust, flexible, and complex spoken dialogue systems.

People tailor their communication to their interlocutor, and no individual style of communication is adequate (or inadequate) a priori (Newman-Norlund, et al. 1989). Callers have little tolerance for error within a system, and they can justify the effort required to engage in collaborative problem solving with a system only if they recognize its usefulness. CheckItOut+'s "question and offer" strategy may have been perceived as error-ridden and slow, although it did indeed improve task success and identified books faster than CheckItOut did. Nonetheless, ordering books by telephone did not appear to be an essential service for our callers. In contrast, patrons of the Heiskell library would be more likely to value such service highly and find it more essential than our callers did. People also welcome collaboration and problem solving with other people. Should next-generation computer systems become more skilled at dialogue and participate in more complex, flexible, human-like

conversation, people may in turn become more willing to engage in mutual collaboration with a machine toward a common goal.

Many fundamental features of spoken dialogue systems are task-independent. A spoken dialogue system must accept speech recognition and produce some semantic representation of the recognized string. Most also rely on a knowledge base. The approaches and results reported in this dissertation should therefore be applicable to other applications where a rich knowledge base can provide context. Although many aspects of our dialogues are task-specific, many others are task-independent. Our ability to predict a wizard's tolerance for noise in the ASR as an influence on making a query or asking a question bodes well for other Olympus/RavenClaw applications, which share the same system features. The ability to describe an object in more than one way was important in our wizards' recovery strategies, and that should be true of other applications as well.

RFW is a general feature selection method. It is easy to implement, and can be used in any domain with any application. RFW+ is domain-specific; it can be used to select features on data derived from dialogue systems, provided that the features are categorized relative to their provenance in the SLU process. RFW+ is not confined to pipelined spoken dialogue systems. Any asynchronous spoken dialogue system could provide feature categorizations relative to different processing modules (e.g., (Allen, et al. 2001, Epstein, et al. 2010, Raux and Eskenazi 2007, Skantze and Gustafson 2009)). The selected features could then be used to implement decision-making modules that work asynchronously to make decisions. Feature selection is particularly

important for them as well because it is not known a priori which features within a complex system should be referenced by any individual decision-making model.

People can process high levels of noise and ambiguity during dialogue with little effort. The research described in this dissertation demonstrates that, when people are restricted to the same input and actions as those of a dialogue system and must contend with understanding errors, they rely on context and questions to disambiguate. Those human strategies can be learned from dialogue system features alone and, when implemented within a spoken dialogue system, those strategies improve task success. It is not known a priori which system features better support data-driven models of error detection and recovery. This research demonstrates that a good feature ranking tailored to a learner and a data set can be obtained through the random sampling and evaluation of feature subsets to learn feature weights. Moreover, that ranking can be obtained with relatively low computational cost. This research also demonstrates that knowledge about the provenance of features along all stages of spoken language processing can further refine the feature selection process with little additional cost.

Appendix A

Development

This thesis included the following development tasks:

Implemented the initial CheckItOut grammar and NLG templates (Perl)

Implemented the initial CheckItOut task reasoner (Perl)

Designed, implemented and maintained the CheckItOut task tree (C++)

Designed and implemented XFF to extend Weka (Java)

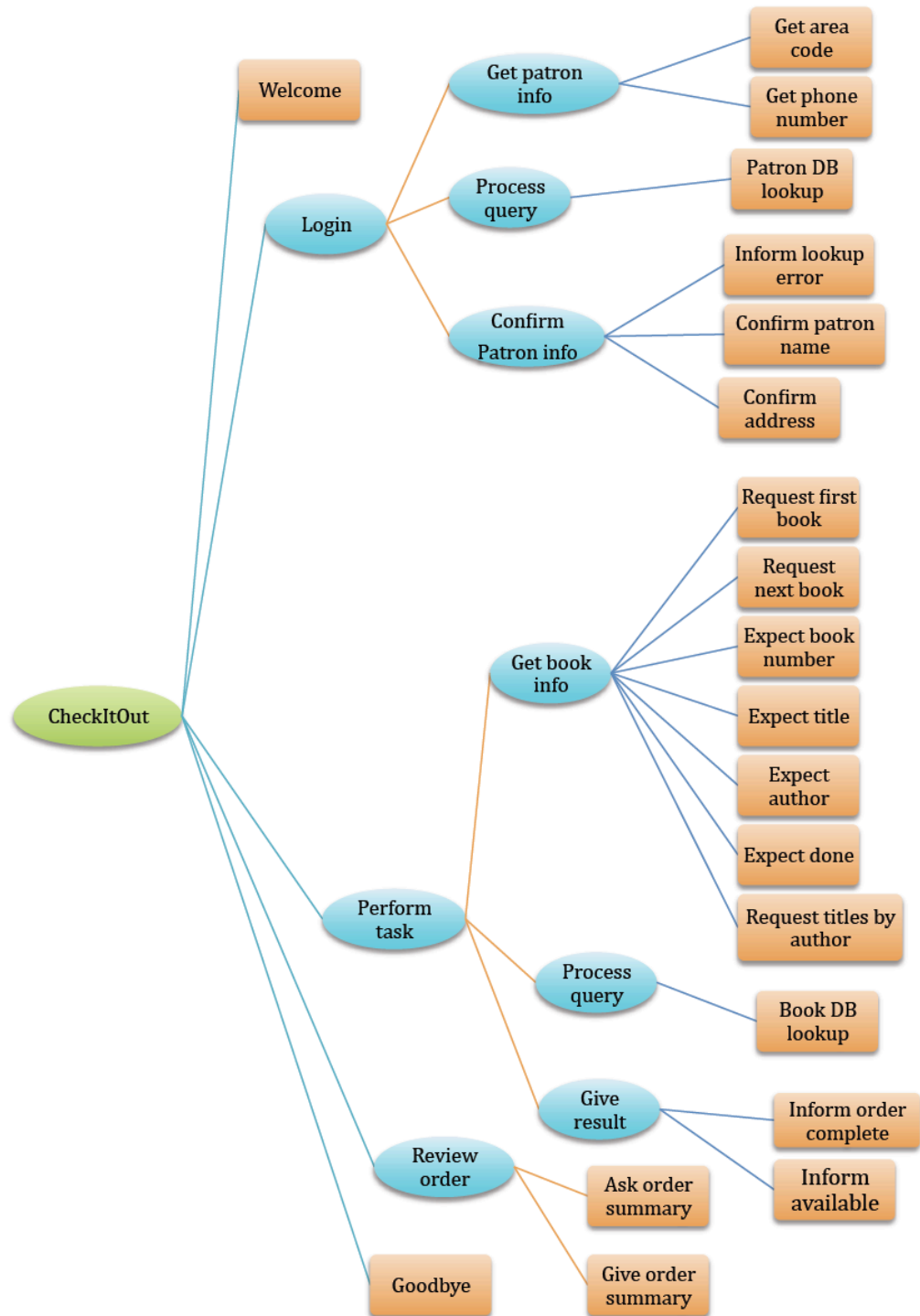
Designed and implemented the wizard module and GUI for the full dialogue experiment (Java)

Implemented log parsing and data pre-processing for feature extraction (Python)

Designed and implemented CheckItOut+'s dialogue manager (Java)

Appendix B

CheckItOut's task tree



Appendix C

Prompts in the full dialogue experiment

During a call, the wizard could prompt the user with the following:

Basic actions

Next book?

Book_title, got it! Next book?

OK, we now have four books. Would you like a summary of your order?

book_title, got it! We now have four books. Would you like a summary of your order?

Order summary as a list of titles and authors of the books ordered in the call

Thanks for calling. Have a nice day!

Auxiliary actions

Are you still there?

Did you say you want to end this call?

I'm sorry, but we already ordered that book. Next?

I'm sorry, but I cannot order all your books in this call. I will hang up now.

Please call again.

Sorry I must have misunderstood you, let's try again. Next book?

Non-understanding

Can you repeat the book request please?

Sorry, I didn't catch that.

I'm sorry; I'm still having trouble understanding you.

I'm still having trouble. Let's try the next book and come back to this one later.

Questions about the ASR

Did you speak the word ___ ?
 Does it begin with ___ ?
 Does it end with ___ ?
 How many words are there?
 What's the first word?
 What's the last word?

Questions about the book request

What is the book about?
 Did you ask for a title?
 Did you ask for an author?
 Can I have the RC number?
 What is the author's name?
 What is the author's first name?
 What is the author's last name?
 Is the author female?

Questions about the query results

Does the title contain the word ___ ?
 Does the title contain the word ___ or the word ___ ?
 Is the title ___ ?
 Is the title ___ or ___ ?
 Is the author ___ ?
 Is the author ___ or ___ ?
 Is the author's first name ___ ?
 Is the author's first name ___ or ___ ?
 Is the author's last name ___ ?
 Is the author's last name ___ or ___ ?
 Is the catalogue number ___ ?

Appendix D

Datasets from the UCI repository

Data set	Description
CARDIOTOCOGRAPHY	Fetal cardiograms (CTG's) automatically processed with their respective diagnostic features. The CTG's were classified by three expert obstetricians, and a consensus classification label assigned to each of them.
ROBOT NAVIGATION	Robot wall-following task. Features are raw values from measurements of 24 ultrasound sensors, labeled with correct robot steering direction (e.g., Slight-Right-Turn).
STEEL PATES FAULTS	A data set of steel plates' faults, classified into seven different types.
BREAST CANCER (W,D)	Features are computed from a digitized image of a fine needle aspirate of a breast mass. They describe characteristics of the cell nuclei present in the image, and are labeled as benign or malignant.
COVERTYPE	Dataset for the prediction of forest cover type from cartographic variables only (i.e., no remotely sensed data). The actual forest cover type for a given observation (30 × 30 meter cell) was determined from US Forest Services (USFS) Region 2 Resource Information System (RIS) data. Labels indicate seven possible classifications.
MUSK (VERSION 2)	All possible low-energy conformations of 102 molecules described by features of shape and conformation. Molecules are labeled as musk or non-musk.
ISOLET	150 subjects spoke each letter of the alphabet twice. Features include spectral coefficients, contour features, sonorant features, pre-sonorant features, and post-sonorant features. Instances are labeled with the corresponding letter of the alphabet.

Appendix E

Experimental results with RFW

The assessment of a feature selection method with respect to a data set and a learner is reported here according to three criteria: speed, size, and accuracy. *Speed* is the time the feature selection method requires to produce its feature subset on the training set T . *Size* is the number of features in the subset selected by that method. *Accuracy* gauges the learner's correctness on the holdout set O .

Results for C4.5 appear in Tables E.1 through E.3 for logistic regression in Tables E.4 and E.5, and for SVM in Tables E.6 through E.8. Each table reports on speed, size, and accuracy as measured by %correct and κ . A statistically significant difference in accuracy from CFS and/or GFS ($p < 0.05$) is in boldface.

A + in the comparison column indicates that RFW outperformed the corresponding baseline, a – indicates that it underperformed it. For example, in Table E.1 on DIALOGUE SEARCHED, RFW5 performed as well as both CFS and GFS, while RFW10, RFW15, and RFW20 outperformed CFS and performed as well as GFS.

Table E.1: Results for feature selection with a C4.5 learner on data from the full dialogue experiment. Two accuracy metrics are reported as mean \pm standard deviation. Time is reported to the nearest second. Boldface indicates that RFW significantly differs ($p < 0.05$) from CFS, GFS, or both. + indicates that RFW outperformed a baseline; - that it underperformed it.

data set	FS	comparison		κ	%correct	time	subset size
		CFS	GFS				
DIALOGUE ALL	CFS	NA	NA	0.74 ± 0.04	87.52 ± 2.18	8	9
	GFS	NA	NA	0.81 ± 0.03	91.36 ± 1.55	2904	9
	RFW5	+	-	0.76 ± 0.04	88.70 ± 1.62	151	5
	RFW10	+	-	0.74 ± 0.05	88.05 ± 2.21	151	10
	RFW15	+	-	0.75 ± 0.03	88.23 ± 1.68	151	15
	RFW20	+	-	0.78 ± 0.03	90.09 ± 1.48	151	20
DIALOGUE CORRELATED	CFS	NA	NA	0.99 ± 0.01	99.59 ± 0.25	7	9
	GFS	NA	NA	0.98 ± 0.01	99.03 ± 0.34	2894	12
	RFW5		+	0.99 ± 0.01	99.59 ± 0.25	118	5
	RFW10		+	0.99 ± 0.01	99.62 ± 0.28	118	10
	RFW15		+	0.99 ± 0.01	99.62 ± 0.28	118	15
	RFW20		+	0.99 ± 0.01	99.59 ± 0.25	118	20
DIALOGUE UNSEARCHED	CFS	NA	NA	0.22 ± 0.05	68.99 ± 2.06	5	8
	GFS	NA	NA	0.33 ± 0.05	71.36 ± 1.87	809	15
	RFW5	+		0.34 ± 0.04	71.03 ± 2.04	79	5
	RFW10	+		0.32 ± 0.05	70.21 ± 2.40	79	10
	RFW15	+		0.36 ± 0.08	72.63 ± 2.96	79	15
	RFW20	+		0.34 ± 0.06	71.36 ± 1.94	79	10
	RFW20 -200	+	+	0.39 ± 0.06	73.79 ± 2.30	371	20
DIALOGUE SEARCHED	CFS	NA	NA	0.26 ± 0.14	88.70 ± 1.93	4	13
	GFS	NA	NA	0.46 ± 0.13	92.07 ± 1.54	382	7
	RFW5			0.21 ± 0.18	90.49 ± 2.19	81	5
	RFW10	+		0.36 ± 0.15	91.08 ± 1.49	81	10
	RFW15	+		0.42 ± 0.13	91.28 ± 1.46	81	15
	RFW20	+		0.49 ± 0.12	91.58 ± 2.19	81	20

Table E.2: Results for feature selection with a C4.5 learner on seven UCI data sets. Two accuracy metrics are reported as mean \pm standard deviation. Time is reported to the nearest second. Boldface indicates that RFW significantly differs ($p < 0.05$) from CFS, GFS, or both. + indicates that RFW outperformed a baseline; - that it underperformed it.

data set	FS	comparison		κ	%correct	time	subset size
		CFS	GFS				
CARDIOTOC- OGRAPHY	CFS	NA	NA	0.69 ± 0.08	83.28 ± 4.66	1	10
	GFS	NA	NA	0.86 ± 0.07	92.46 ± 3.50	126	10
	RFW5			0.82 ± 0.10	90.56 ± 5.12	4	5
	RFW10	+		0.85 ± 0.07	91.99 ± 3.72	4	10
	RFW15	+		0.83 ± 0.08	91.05 ± 4.28	4	15
	RFW20	+		0.84 ± 0.07	91.53 ± 3.86	4	20
ROBOT NAVIGATION	CFS	NA	NA	0.97 ± 0.01	98.07 ± 1.01	2	6
	GFS	NA	NA	0.97 ± 0.01	98.07 ± 1.01	317	8
	RFW5	-	-	0.95 ± 0.02	96.60 ± 1.56	15	5
	RFW10			0.95 ± 0.02	96.51 ± 1.48	15	10
	RFW15			0.95 ± 0.02	96.60 ± 1.50	15	15
	RFW20			0.96 ± 0.01	97.43 ± 0.84	15	20
STEEL PLATES FAULTS	CFS	NA	NA	0.38 ± 0.12	56.19 ± 7.96	5	14
	GFS	NA	NA	0.46 ± 0.10	61.87 ± 7.63	146	8
	RFW5			0.44 ± 0.12	60.34 ± 7.61	12	5
	RFW10			0.42 ± 0.11	58.76 ± 8.11	12	10
	RFW15			0.43 ± 0.12	60.03 ± 8.39	12	15
	RFW 20			0.41 ± 0.11	58.47 ± 6.35	12	20
BREAST CANCER	CFS	NA	NA	0.78 ± 0.16	89.24 ± 8.39	0	10
	GFS	NA	NA	0.80 ± 0.17	90.45 ± 8.43	48	3
	RFW5			0.76 ± 0.18	87.80 ± 9.23	1	5
	RFW10			0.76 ± 0.18	87.80 ± 9.23	1	10
	RFW15			0.75 ± 0.17	87.73 ± 8.49	1	15
	RFW20			0.77 ± 0.16	88.56 ± 8.42	1	20
COVERTYPE	CFS	NA	NA	0.58 ± 0.04	71.00 ± 2.64	3	13
	GFS	NA	NA	0.72 ± 0.05	80.55 ± 3.43	7830	26
	RFW5	+	-	0.64 ± 0.05	75.10 ± 3.69	52	5
	RFW10	+		0.69 ± 0.04	78.50 ± 3.12	52	10
	RFW15	+		0.70 ± 0.05	79.00 ± 2.24	52	15
	RFW20	+		0.70 ± 0.05	78.70 ± 3.28	52	20
MUSK (VERSION 2)	CFS	NA	NA	0.71 ± 0.13	97.19 ± 1.18	5	15
	GFS	NA	NA	0.64 ± 0.10	96.13 ± 0.97	3911	11
	RFW5	+		0.81 ± 0.11	98.03 ± 1.14	71	5
	RFW10			0.79 ± 0.12	97.65 ± 1.45	71	10
	RFW15	+		0.80 ± 0.12	97.88 ± 1.37	71	15
	RFW20			0.65 ± 0.13	96.59 ± 1.14	71	20
	RFW20 -100	+	+	0.81 ± 0.10	98.10 ± 0.82	91	20
ISOLET	CFS	NA	NA	0.69 ± 0.09	70.79 ± 8.79	198	133
	GFS	NA	NA	0.67 ± 0.09	68.19 ± 8.50	77381	38
	RFW10			0.59 ± 0.08	61.12 ± 7.56	456	10
	RFW15			0.66 ± 0.05	67.22 ± 4.61	456	15
	RFW20			0.64 ± 0.06	65.61 ± 5.73	456	20
	RFW30			0.65 ± 0.07	66.88 ± 6.43	456	30
	RFW20 -200		+	0.77 ± 0.08	77.81 ± 7.82	816	20

Table E.3: Results for feature selection on the full COVERTYPE and ISOLET data sets with a C4.5 learner. Two accuracy metrics are reported as mean \pm standard deviation. Time is reported to the nearest second. Boldface indicates that RFW significantly differs ($p < 0.05$) from CFS, GFS, or both. + indicates that RFW outperformed a baseline; - that it underperformed it.

data set	FS	comparison		κ	%correct	time	subset size
		CFS	GFS				
full COVERTYPE	CFS	NA	NA	0.52 ± 0.01	80.20 ± 0.28	2070	17
	GFS	NA	NA	Did not complete in 24 hours			
	RFW5	+	+	0.85 ± 0.01	93.03 ± 0.14	37644	5
	RFW10	+	+	0.86 ± 0.01	93.48 ± 0.34	37644	10
	RFW15	+	+	0.87 ± 0.01	93.93 ± 0.26	37644	15
	RFW20	+	+	0.88 ± 0.01	94.12 ± 0.23	37644	20
full ISOLET	CFS	NA	NA	0.77 ± 0.04	78.32 ± 3.63	316	192
	GFS	NA	NA	Did not complete in 24 hours			
	RFW10	-	+	0.66 ± 0.05	67.09 ± 5.01	2186	10
	RFW15	-	+	0.68 ± 0.05	69.40 ± 4.62	2186	15
	RFW20	-	+	0.69 ± 0.04	69.98 ± 3.96	2186	20
	RFW30	-	+	0.70 ± 3.01	71.01 ± 3.01	2186	30
	RFW30 -100			+	0.74 ± 0.03	75.56 ± 3.05	3206

Table E.4: Results for feature selection with logistic regression on data from the full dialogue experiment. Two accuracy metrics are reported as mean \pm standard deviation. Time is reported to the nearest second. Boldface indicates that RFW significantly differs ($p < 0.05$) from CFS, GFS, or both. + indicates that RFW outperformed a baseline; - that it underperformed it.

data set	FS	comparison		κ	%correct	time	subset size
		CFS	GFS				
DIALOGUE ALL	CFS	NA	NA	0.74 ± 0.04	87.52 ± 2.18	8	9
	GFS	NA	NA	0.77 ± 0.02	89.14 ± 1.10	3767	16
	RFW5			0.75 ± 0.02	88.58 ± 1.19	96	5
	RFW10			0.76 ± 0.02	88.97 ± 0.97	96	10
	RFW15			0.76 ± 0.03	88.90 ± 1.25	96	15
	RFW20			0.76 ± 0.02	89.10 ± 1.10	96	20
DIALOGUE CORRELATED	CFS	NA	NA	0.99 ± 0.01	99.59 ± 0.25	7	9
	GFS	NA	NA	0.97 ± 0.01	98.76 ± 0.36	2875	4
	RFW5		+	0.99 ± 0.01	99.59 ± 0.25	215	5
	RFW10		+	0.99 ± 0.01	99.65 ± 0.30	215	10
	RFW15		+	0.99 ± 0.01	99.65 ± 0.30	215	15
	RFW20		+	0.99 ± 0.01	99.65 ± 0.30	215	20
DIALOGUE UNSEARCHED	CFS	NA	NA	0.32 ± 0.05	70.21 ± 2.52	5	8
	GFS	NA	NA	0.29 ± 0.08	71.42 ± 3.09	3254	16
	RFW5			0.31 ± 0.08	70.60 ± 3.15	102	5
	RFW10			0.30 ± 0.04	70.04 ± 2.29	102	10
	RFW15			0.29 ± 0.05	69.60 ± 2.18	102	15
	RFW20			0.32 ± 0.04	70.43 ± 1.85	102	20
DIALOGUE SEARCHED	CFS	NA	NA	0.26 ± 0.14	88.70 ± 1.93	4	13
	GFS	NA	NA	No features selected			
	RFW5	+	+	0.24 ± 0.20	91.18 ± 1.70	41	5
	RFW10	+	+	0.37 ± 0.16	91.58 ± 1.16	41	10
	RFW15	+	+	0.40 ± 0.14	91.77 ± 1.62	41	15
	RFW20			0.36 ± 0.15	90.58 ± 2.06	41	20

Table E.5: Results for feature selection with logistic regression on seven UCI data sets. Two accuracy metrics are reported as mean \pm standard deviation. Time is reported to the nearest second. Boldface indicates that RFW significantly differs ($p < 0.05$) from CFS, GFS, or both. + indicates that RFW outperformed a baseline; - that it underperformed it.

data set	FS	comparison		κ	%correct	time	subset size
		CFS	GFS				
CARDIOTOC- OGRAPHY	CFS	NA	NA	0.84 ± 0.07	91.52 ± 3.73	1	10
	GFS	NA	NA	0.64 ± 0.12	80.00 ± 6.79	45	4
	RFW5		+	0.83 ± 0.09	90.83 ± 4.87	10	5
	RFW10		+	0.85 ± 0.08	91.78 ± 4.40	10	10
	RFW15		+	0.86 ± 0.07	92.24 ± 3.82	10	15
	RFW20		+	0.83 ± 0.09	90.59 ± 4.82	10	20
ROBOT NAVIGATION	CFS	NA	NA	0.97 ± 0.01	98.07 ± 1.01	2	6
	GFS	NA	NA	0.95 ± 0.02	96.60 ± 1.68	179	4
	RFW5			0.96 ± 0.02	97.24 ± 1.43	67	5
	RFW10			0.96 ± 0.01	97.52 ± 0.87	67	10
	RFW15			0.96 ± 0.01	97.43 ± 0.95	67	15
	RFW20			0.96 ± 0.01	97.34 ± 0.91	67	20
STEEL PLATES FAULTS	CFS	NA	NA	0.38 ± 0.12	56.19 ± 7.96	5	14
	GFS	NA	NA	0.41 ± 0.12	58.25 ± 8.61	2654	14
	RFW5			0.34 ± 0.11	53.60 ± 7.61	77	5
	RFW10			0.40 ± 0.14	58.00 ± 9.64	77	10
	RFW15			0.41 ± 0.15	58.27 ± 10.69	77	15
	RFW20			0.38 ± 0.13	56.21 ± 9.09	77	20
BREAST CANCER	CFS	NA	NA	0.78 ± 0.16	89.24 ± 8.39	0	10
	GFS	NA	NA	0.76 ± 0.14	88.41 ± 7.39	30	6
	RFW5			0.78 ± 0.16	89.39 ± 8.08	3	5
	RFW10			0.83 ± 0.14	91.06 ± 7.43	3	10
	RFW15			0.80 ± 0.15	90.23 ± 7.82	3	15
	RFW20			0.78 ± 0.16	89.32 ± 8.24	3	20
COVERTYPE	CFS	NA	NA	0.58 ± 0.04	71.00 ± 2.65	3	13
	GFS	NA	NA	0.70 ± 0.04	79.10 ± 2.73	43437	26
	RFW5		+	0.64 ± 0.06	75.40 ± 4.32	510	5
	RFW10		+	0.66 ± 0.06	76.55 ± 4.11	510	10
	RFW15		+	0.67 ± 0.05	77.05 ± 3.16	510	15
	RFW20		+	0.67 ± 0.05	77.05 ± 3.16	510	20
MUSK (VERSION 2)	CFS	NA	NA	0.71 ± 0.13	97.19 ± 1.19	5	15
	GFS	NA	NA	0.63 ± 0.12	95.83 ± 1.21	8764	24
	RFW5			0.58 ± 0.15	96.13 ± 1.21	123	5
	RFW10			0.55 ± 0.16	95.52 ± 1.44	123	10
	RFW15			0.61 ± 0.12	95.98 ± 1.18	123	15
	RFW20			0.63 ± 0.14	96.28 ± 1.21	123	20
ISOLET	CFS	NA	NA	0.69 ± 0.09	70.79 ± 8.55	198	133
	GFS	NA	NA	Did not complete in 24 hours			
	RFW	-		Did not complete in 24 hours			

Table E.6: Results for feature selection with SVM on data from the full dialogue experiment. Two accuracy metrics are reported as mean \pm standard deviation. Time is reported to the nearest second. Boldface indicates that RFW significantly differs ($p < 0.05$) from CFS, GFS, or both. + indicates that RFW outperformed a baseline; - that it underperformed it.

data set	FS	comparison		κ	%correct	time	subset size
		CFS	GFS				
DIALOGUE ALL	CFS	NA	NA	0.26 ± 0.14	88.70 ± 1.93	8	9
	GFS	NA	NA	Selected no features			
	RFW5		+	0.28 ± 0.01	89.59 ± 0.51	89	5
	RFW10		+	0.18 ± 0.01	89.39 ± 0.93	89	10
	RFW15		+	0.27 ± 0.01	89.09 ± 1.13	89	15
	RFW20		+	0.21 ± 0.03	89.49 ± 0.68	89	20
DIALOGUE CORRELATED	CFS	NA	NA	0.99 ± 0.01	99.59 ± 0.25	7	9
	GFS	NA	NA	0.99 ± 0.01	99.67 ± 0.26	2357	7
	RFW5			0.99 ± 0.01	99.59 ± 0.25	1821	5
	RFW10			0.99 ± 0.01	99.65 ± 0.19	1821	10
	RFW15			0.99 ± 0.01	99.59 ± 0.25	1821	15
	RFW20			0.99 ± 0.01	99.59 ± 0.25	1821	20
DIALOGUE UNSEARCHED	CFS	NA	NA	0.32 ± 0.05	70.21 ± 2.52	5	8
	GFS	NA	NA	0.18 ± 0.07	67.95 ± 1.59	2233	3
	RFW5			0.21 ± 0.06	70.54 ± 2.54	598	5
	RFW10		+	0.26 ± 0.05	70.82 ± 1.65	598	10
	RFW15			0.24 ± 0.05	70.49 ± 2.59	598	15
	RFW20			0.26 ± 0.05	70.43 ± 2.64	598	20
	RFW10 -200	+	+	0.29 ± 0.07	71.15 ± 3.33	1062	10
	RFW15 -200	+	+	0.29 ± 0.08	71.15 ± 3.47	1062	15
	DIALOGUE SEARCHED	CFS	NA	NA	0.44 ± 0.06	75.16 ± 2.62	4
	GFS	NA	NA	Did not complete in 24 hours			
	RFW5	+	+	0.62 ± 0.03	82.33 ± 1.41	654	5
	RFW10	+	+	0.63 ± 0.03	83.16 ± 1.49	654	10
	RFW15	+	+	0.73 ± 0.03	87.49 ± 1.59	654	15
	RFW20	+	+	0.73 ± 0.04	87.40 ± 1.72	654	20

Table E.7: Results for feature selection with SVM on seven UCI data sets. Two accuracy metrics are reported as mean \pm standard deviation. Time is reported to the nearest second. Boldface indicates that RFW significantly differs ($p < 0.05$) from CFS, GFS, or both. + indicates that RFW outperformed a baseline; - that it underperformed it.

data set	FS	comparison		κ	%correct	time	subset size
		CFS	GFS				
CARDIOTOC- OGRAPHY	CFS	NA	NA	0.64 ± 0.07	89.52 ± 3.73	1	10
	GFS	NA	NA	0.57 ± 0.09	76.46 ± 5.39	42	3
	RFW5	+		0.79 ± 0.06	89.16 ± 3.54	30	5
	RFW10	+	+	0.85 ± 0.05	91.76 ± 2.55	30	10
	RFW15	+	+	0.84 ± 0.06	91.52 ± 3.19	30	15
	RFW20	+	+	0.87 ± 0.07	93.17 ± 3.91	30	20
ROBOT NAVIGATION	CFS	NA	NA	0.97 ± 0.01	98.07 ± 1.01	2	6
	GFS	NA	NA	0.96 ± 0.02	97.15 ± 1.33	2427	16
	RFW5		-	0.93 ± 0.02	95.50 ± 1.59	168	5
	RFW10			0.95 ± 0.03	96.79 ± 2.08	168	10
	RFW15			0.96 ± 0.02	97.43 ± 1.60	168	15
	RFW20			0.96 ± 0.01	97.34 ± 0.91	168	20
STEEL PLATES FAULTS	CFS	NA	NA	0.38 ± 0.12	56.19 ± 7.96	5	14
	GFS	NA	NA	0.36 ± 0.15	53.65 ± 10.78	939	14
	RFW5			0.45 ± 0.12	60.59 ± 8.60	186	5
	RFW10			0.43 ± 0.13	59.26 ± 8.60	186	10
	RFW15			0.42 ± 0.16	58.72 ± 10.72	186	15
	RFW20			0.43 ± 0.14	59.53 ± 9.31	186	20
BREAST CANCER	CFS	NA	NA	0.78 ± 0.16	89.24 ± 8.39	0	10
	GFS	NA	NA	0.76 ± 0.09	88.41 ± 4.56	46	7
	RFW5			0.82 ± 0.14	91.06 ± 7.43	9	5
	RFW10			0.82 ± 0.14	91.06 ± 7.43	9	10
	RFW15			0.82 ± 0.14	91.06 ± 7.43	9	15
	RFW20			0.80 ± 0.15	90.30 ± 7.81	9	20
COVERTYPE	CFS	NA	NA	0.58 ± 0.04	71.00 ± 2.64	3	13
	GFS	NA	NA	0.52 ± 0.06	66.45 ± 4.35	15392	24
	RFW5	+	+	0.65 ± 0.05	75.95 ± 3.69	328	5
	RFW10	+	+	0.65 ± 0.05	75.80 ± 3.63	328	10
	RFW15	+	+	0.66 ± 0.05	76.55 ± 3.30	328	15
	RFW20	+	+	0.71 ± 0.03	79.95 ± 2.17	328	20
MUSK (VERSION 2)	CFS	NA	NA	0.71 ± 0.13	97.19 ± 1.19	5	15
	GFS	NA	NA	Selected no features			
	RFW5	-	+	0.32 ± 0.12	94.84 ± 0.69	99	5
	RFW10		+	0.55 ± 0.14	95.52 ± 1.32	99	10
	RFW15		+	0.65 ± 0.14	96.59 ± 1.14	99	15
	RFW20		+	0.65 ± 0.14	96.59 ± 1.14	99	20
ISOLET	CFS	NA	NA	0.69 ± 0.09	70.78 ± 8.79	198	133
	GFS	NA	NA	Did not complete in 24 hours			
	RFW10	-	+	0.51 ± 0.07	53.04 ± 6.55	5740	10
	RFW15		+	0.55 ± 0.08	57.22 ± 8.13	5740	15
	RFW20		+	0.56 ± 0.09	57.54 ± 8.78	5740	20
	RFW30		+	0.58 ± 0.08	59.47 ± 7.97	5740	30

Table E.8: Results for feature selection on the full COVERTYPE and ISOLET data sets with SVM. Two accuracy metrics are reported as mean \pm standard deviation. Time is reported to the nearest second. Boldface indicates that RFW significantly differs ($p < 0.05$) from CFS, GFS, or both. + indicates that RFW outperformed a baseline; - that it underperformed it.

data set	FS	comparison		κ	%correct	time	subset size
		CFS	GFS				
full COVERTYPE	CFS	NA	NA	0.52 ± 0.01	80.20 ± 0.28	2070	17
	GFS	NA	NA	Did not complete in 24 hours			
	RFW5	+	+	0.57 ± 0.52	85.00 ± 17.48	9963	5
	RFW10	+	+	0.55 ± 0.44	85.50 ± 12.57	9963	10
	RFW15	+	+	0.55 ± 0.44	85.50 ± 12.57	9963	15
	RFW20	+	+	0.65 ± 0.47	90.00 ± 12.90	9963	20
full ISOLET	CFS	NA	NA	0.77 ± 0.04	78.32 ± 3.63	316	192
	GFS	NA	NA	Did not complete in 24 hours			
	RFW10	-	+	0.49 ± 0.02	57.73 ± 2.75	6028	10
	RFW15	-	+	0.62 ± 0.05	63.12 ± 4.09	6028	15
	RFW20	-	+	0.62 ± 0.05	63.56 ± 4.74	6028	20
	RFW30	-	+	0.62 ± 0.04	63.76 ± 4.13	6028	30

Appendix F

Experimental results with RFW+

The assessment of a feature selection method with respect to three spoken dialogue data sets and a learner is reported here according to three criteria: speed, size, and accuracy. *Speed* is the time the feature selection method requires to produce its feature subset on the training set T . *Size* is the number of features in the subset selected by that method. *Accuracy* gauges the learner's correctness on the holdout set O .

Results for C4.5 appear in Table F.1, for logistic regression in Table F.2, and for SVM in Table F.3. Each table reports on speed, size, and accuracy as measured by %correct and κ . Any statistically significant difference in accuracy from CFS and/or GFS ($p < 0.05$) is in boldface. A + in the comparison column indicates that RFW outperformed the corresponding baseline. For example, in Table F.1 with meta-groups, RFW+20 outperformed both baselines, while all other RFW+s outperformed CFS and performed as well as GFS.

Table F.1: Results for RFW and RFW+ with logistic regression on the holdout set O for DIALOGUE ALL. Two accuracy metrics are reported as mean \pm standard deviation. Time is reported to the nearest second. Boldface indicates that RFW or RFW+ significantly differs ($p < 0.05$) from CFS, GFS, or both. + indicates that RFW or RFW+ outperformed a baseline; - that it underperformed it.

category	FS	comparison		κ	%correct	time	subset size
		CFS	GFS				
NA	CFS	NA	NA	0.74 ± 0.04	87.52 ± 2.18	8	9
	GFS	NA	NA	0.77 ± 0.02	89.14 ± 1.10	3767	16
NA	RFW5			0.75 ± 0.02	88.58 ± 1.19	96	5
	RFW10			0.76 ± 0.02	88.97 ± 0.97	96	10
	RFW15			0.76 ± 0.03	88.90 ± 1.25	96	15
	RFW20			0.76 ± 0.02	89.10 ± 1.10	96	20
META-GROUPS	RFW+3	+		0.76 ± 0.03	88.94 ± 1.53	436	30
	RFW+5	+		0.79 ± 0.01	90.59 ± 0.74	436	16
	RFW+10	+		0.79 ± 0.03	90.26 ± 1.24	436	40
	RFW+15	+		0.79 ± 0.03	90.47 ± 1.34	436	32
	RFW+20	+	+	0.79 ± 0.02	98.47 ± 1.20	436	32
GROUPS	RFW+3			0.75 ± 0.03	88.55 ± 1.37	313	15
	RFW+5	+		0.78 ± 0.03	90.09 ± 1.29	313	15
	RFW+10	+		0.78 ± 0.03	90.00 ± 1.33	313	16
	RFW+15	+		0.78 ± 0.03	89.94 ± 1.44	313	20
	RFW+20	+		0.79 ± 0.02	90.27 ± 0.82	313	25
SUB-GROUPS	RFW+3	+		0.75 ± 0.03	88.49 ± 1.35	189	18
	RFW+5	+		0.80 ± 0.03	90.94 ± 1.48	189	15
	RFW+10	+		0.80 ± 0.03	91.00 ± 1.54	189	17
	RFW+15	+	+	0.80 ± 0.02	90.91 ± 1.18	189	20
	RFW+20	+	+	0.81 ± 0.01	91.41 ± 0.71	189	24

Table F.2: Results for RFW and RFW+ with logistic regression on the holdout set O for DIALOGUE UNSEARCHED. Two accuracy metrics are reported as mean \pm standard deviation. Time is reported to the nearest second. Boldface indicates that RFW or RFW+ significantly differs ($p < 0.05$) from CFS, GFS, or both. + indicates that RFW or RFW+ outperformed a baseline; - that it underperformed it.

category	FS	comparison		κ	%correct	time	subse t size
		CFS	GFS				
NA	CFS	NA	NA	0.32 ± 0.05	70.21 ± 2.52	5	8
	GFS	NA	NA	0.29 ± 0.08	71.42 ± 3.09	3254	16
NA	RFW5			0.31 ± 0.08	70.60 ± 3.15	102	5
	RFW10			0.30 ± 0.04	70.04 ± 2.29	102	10
	RFW15			0.29 ± 0.05	69.60 ± 2.18	102	15
	RFW20			0.32 ± 0.04	70.43 ± 1.85	102	20
META- GROUPS	RFW+3			0.28 ± 0.06	70.54 ± 3.18	116	18
	RFW+5			0.27 ± 0.07	70.15 ± 2.43	116	11
	RFW+10			0.29 ± 0.07	71.04 ± 2.87	116	13
	RFW+15	+		0.27 ± 0.08	71.64 ± 2.28	116	16
	RFW+20			0.29 ± 0.07	70.54 ± 3.17	116	24
GROUPS	RFW+3			0.29 ± 0.04	70.72 ± 1.49	142	10
	RFW+5	+		0.31 ± 0.07	71.59 ± 2.41	142	17
	RFW+10			0.29 ± 0.06	70.68 ± 1.59	142	17
	RFW+15	+		0.30 ± 0.05	71.37 ± 1.33	142	17
	RFW+20	+		0.32 ± 0.06	71.26 ± 2.13	142	24
SUB- GROUPS	RFW+3			0.26 ± 0.08	69.71 ± 2.14	119	8
	RFW+5			0.27 ± 0.09	70.26 ± 3.59	119	8
	RFW+10			0.27 ± 0.09	70.26 ± 2.62	119	10
	RFW+15			0.25 ± 0.10	70.60 ± 3.32	119	16
	RFW+20	+		0.28 ± 0.05	70.93 ± 2.33	119	23

Table F.3: Results for RFW and RFW+ with logistic regression on the holdout set O for DIALOGUE SEARCHED. Two accuracy metrics are reported as mean \pm standard deviation. Time is reported to the nearest second. Boldface indicates that RFW or RFW+ significantly differs ($p < 0.05$) from CFS, GFS, or both. + indicates that RFW or RFW+ outperformed a baseline; - that it underperformed it.

category	FS	comparison		κ	%correct	time	subset size
		CFS	GFS				
NA	CFS	NA	NA	0.26 ± 0.14	88.70 ± 1.93	4	13
	GFS	NA	NA	No features selected			
NA	RFW5	+	+	0.24 ± 0.20	91.18 ± 1.70	41	5
	RFW10	+	+	0.37 ± 0.16	91.58 ± 1.16	41	10
	RFW15	+	+	0.40 ± 0.14	91.77 ± 1.62	41	15
	RFW20		+	0.36 ± 0.15	90.58 ± 2.06	41	20
META-GROUPS	RFW+3		+	0.18 ± 0.18	89.29 ± 2.02	103	6
	RFW+5		+	0.28 ± 0.17	89.89 ± 1.80	103	16
	RFW+10	+	+	0.40 ± 0.09	90.98 ± 2.69	103	14
	RFW+15	+	+	0.31 ± 0.26	90.99 ± 3.36	103	18
	RFW+20		+	0.27 ± 0.12	89.20 ± 1.09	103	22
GROUPS	RFW+3	+	+	0.36 ± 0.04	90.64 ± 2.46	116	7
	RFW 5	+	+	0.37 ± 0.20	90.68 ± 2.92	116	10
	RFW+10	+	+	0.33 ± 0.01	90.48 ± 2.15	116	11
	RFW+15	+	+	0.38 ± 0.01	90.69 ± 1.36	116	20
	RFW+20	+	+	0.35 ± 0.03	90.69 ± 2.92	116	25
SUB-GROUPS	RFW+3	+	+	0.29 ± 0.19	89.97 ± 2.02	97	5
	RFW+5	+	+	0.38 ± 0.04	91.08 ± 2.63	97	5
	RFW+10	+	+	0.36 ± 0.03	90.98 ± 2.44	97	11
	RFW+15	+	+	0.39 ± 0.06	91.38 ± 1.93	97	20
	RFW+20	+	+	0.37 ± 0.06	91.89 ± 2.75	97	24

Glossary

%correct: the percent of instances in a data set correctly classified by a model.

A: an accuracy metric used to evaluate a model. Examples include the model's kappa coefficient κ or *%correct*, the percent of correctly classified instance.

A-best: the maximally observed accuracy at any given iteration in RFW and RFW+.

A(L,D,F): the accuracy of the model trained by a learner *L* using the features *F* on the data *D* using 10-fold cross validation, and averaged over the 10 runs.

adjacency pair: in a system-initiative spoken dialogue system, the portion of the dialogue that begins with a system prompt and ends just before the next system prompt.

ASR: see automated speech recognition.

automated speech recognition: automated speech recognition, a spoken dialogue system's transcription of the speech signal into text.

backward elimination: in a wrapper feature selection algorithm, a procedure that begins with the initial feature set *F*, and removes features from it as long as *F*'s accuracy does not decrease.

CFS: correlation-based feature selection, a filter that searches best-first to identify features that are individually highly correlated with the classification label, but minimally correlated with one another.

CheckItOut: a spoken dialogue system implemented for this research that handles book orders by telephone. CheckItOut is built upon the Olympus/RavenClaw dialogue system architecture.

comprehensive confidence features: dialogue system variables that combine two or more confidence features.

confidence features: dialogue system variables that describe a confidence value over other dialogue system variables.

dialogue act: a semantic representation of the system's response.

DIALOGUE ALL: a data set produced by this research from data collected in the full dialogue experiment that includes all instances, and excludes query features.

dialogue manager: the dialogue system component that determines the system's response, usually based upon the semantic representation of the caller's speech, the current context of the dialogue, and possible task constraints.

DIALOGUE SEARCHED: a data set produced by this research from data collected in the full dialogue experiment that includes only instances where there has been a database query, and includes all features.

DIALOGUE UNSEARCHED: a data set produced by this research from data collected in the full dialogue experiment that includes only instances where there has been no database query, and excludes query features.

DM: see dialogue manager.

disfluencies: interruptions in the smooth flow of speech, such as filled pauses, repetitions, mid-word interruptions or false starts.

F: the initial feature set in a feature selection algorithm.

F': a ranking of the features by their weight produced by RFW.

F measure: a measure of model accuracy that accounts for both precision and recall.

feature category: some partition of the feature set (e.g., groups, meta-groups, sub-groups).

feature groups: a feature categorization of medium granularity used in this research to associate system features to the SLU stages in which their values are bound.

feature meta-groups: a feature categorization of coarse granularity used in this research to associate system features to the SLU stages in which their values are bound. Feature meta-groups lump together those feature groups that pertain to the interpretation of the speech signal.

feature-relevance criterion q : in feature selection, a metric that assesses an individual feature's intrinsic desirability for inclusion in a feature subset.

feature selection: An algorithm that, given an initial set of features F and a dataset D , seeks to identify some subset $S \subseteq F$ that is predictive for D .

feature sub-groups: a feature categorization of fine granularity used in this research to associate system features to the SLU stages in which their values are bound. Feature sub-groups separate history features from the group of features to which they pertain.

filter: a feature selection algorithm that selects features independently from the learner.

forward selection: in a wrapper, a procedure that begins with a feature subset $S = \emptyset$, and adds to S features from the initial feature set, as long as S 's accuracy increases.

full-dialogue experiment: a Wizard of Oz study on full dialogue interactions conducted by this research and described in Section 3.2.

GFS: greedy forward selection, a wrapper that performs greedy search on the space of possible feature subsets. At each step it adds the feature that maximizes the accuracy of a learner trained on that feature subset.

Helios: a confidence annotation component built within the Olympus dialogue system architecture, and used to select among competing parses.

history features: dialogue system variables that describe the cumulative value of other system features during a dialogue interaction.

κ : the kappa coefficient that measures agreement among classifiers.

L: see learner.

L₁ regularized linear models: a linear model that applies a constraint on the sum of the coefficients, effectively removing some of the features from the model.

learner: any machine-learning algorithm.

least-represented category: the feature category with the fewest features in a subset *S*.

MICA parser: a large-coverage dependency grammar.

misunderstanding: a situation in which the system produced a representation that does not correspond to what the speaker said.

mixed initiative: a style of dialogue interaction in which the system permits callers to interrupt and volunteer information.

move on: a recovery action in CheckItOut that asks the caller to request the next book and come back to this one later.

natural language generation: a system's natural language representation of a dialogue act issued by the dialogue manager.

NLG: see natural language generation.

no-query model: a wizard model that predicts whether the wizard asks a question or signals non-understanding given that there has been no database query. It is trained on the DIALOGUE UNSEARCHED data set.

non-understanding: a situation in which the system fails to obtain a representation of what the speaker said.

Olympus: a spoken dialogue system architecture created at Carnegie Mellon University, designed to support research in conversational agents.

P(F): in feature selection, the space of feature subsets.

parse: the product of semantic analysis; the semantic structure of the recognized string.

persistence: the termination parameter in RFW.

q: see feature-relevance criterion.

query features: dialogue system variables that describe a database query.

R^2 : The degree to which a statistical model accounts for variability in a data set.

R/O score: the Ratcliff/Obershelp pattern recognition that measures the similarity between two strings as the number of matching characters divided by the total number of characters in the two strings.

RavenClaw: a dialogue management framework built within the Olympus dialogue system architecture.

recognition features: dialogue system variables that describe the recognized string.

recovery action: a dialogue system's action in response to understanding error.

RFW: randomized feature weighting, a novel feature selection algorithm contributed by this research. It uses random sampling of feature subsets in a wrapper framework and evaluates those subsets to learn feature weights. It returns a ranking of the features by weight.

RFW+: randomized feature weighting with knowledge, an extension of RFW that uses knowledge about the provenance of features along the SLU pipeline to select from the features top-ranked by RFW.

S: a feature subset in a feature selection algorithm.

S_{top}: The features top-ranked by RFW.

SDS: see spoken dialogue system.

semantic analysis: the process that determines the semantic content of the recognized string.

SLU: see spoken language understanding.

spoken dialogue system: a system intended to engage effectively in dialogue with people.

spoken language understanding: spoken language understanding, a dialogue system's extraction of a meaningful internal representation of the speech signal.

system initiative: a style of dialogue interaction in which the path of the dialogue is fully controlled by the system.

task tree: a task-dependent hierarchical plan used in dialogue management that specifies possible dialogue states.

text-to-speech: a spoken dialogue system's production of an audio waveform from the prompt generated by the natural language generation component.

title experiment: a pseudo Wizard of Oz study on single turn exchanges for book requests conducted by this research and described in Section 3.1.

TTS: see text-to-speech.

underrepresented category: a feature category that has relatively few features in a feature subset *S*.

voice search: to query a database with the ASR string, the automated speech recognition's transcription of the speech signal.

voice-search model: a wizard model that predicts whether the wizard performed voice search or not. It is trained on the DIALOGUE ALL data set.

WER: word error rate, a common metric of the performance of a speech recognizer that measures the average number of incorrect words in the transcription of the speech signal.

Wizard of Oz study: a study in which subjects interact with what they believe to be a fully automated computer system, unaware that a person is involved in the system's operation to some extent.

WOz: see Wizard of Oz study.

wrapper: a feature selection algorithm that uses a learner to evaluate the prediction quality of subsets of features on a given data set.

yes-query model: a wizard model that predicts whether the wizard offers a book from the voice search results or asks a question, given that there has been a database query. It is trained on the DIALOGUE SEARCHED data set.

XFF: an experimental framework for feature selection developed by this research, that supports the experimentation with, and evaluation of feature selection algorithms.

Bibliography

- Allen, J., Ferguson, G. and Stent, A. An architecture for more realistic conversational systems. In *Proceedings of the Sixth International Conference on Intelligent User Interfaces*, Santa Fe, NM, 2001.
- Almuallim, H. and Dietterich, T. G. Learning With Many Irrelevant Features. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, 547-552, 1991.
- Alpaydin, E. *Introduction to Machine Learning*. The MIT Press, 2004.
- Bangalore, S., Boullier, P., Nasr, A., Rambow, O. and Sagot, B. MICA: a probabilistic dependency parser based on tree insertion grammars application note. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North America Chapter of the Association for Computational Linguistics*, 185-188, Boulder, Colorado, 2009.
- Bohus, D. *Error Awareness and Recovery in Task-Oriented Spoken Dialog Systems*. Ph.D. Thesis. Computer Science Department, Carnegie Mellon University, 2007.
- Bohus, D. and Rudnicky, A. I. Integrating multiple knowledge sources for utterance-level confidence annotation in the CMU Communicator spoken dialog system. Technical Report No. CS-190, Carnegie Mellon University, 2002.
- Bohus, D. and Rudnicky, A. I. RavenClaw: Dialog Management Using Hierarchical Task Decomposition and an Expectation Agenda. In *Proceedings of the Eight European*

- Conference on Speech Communication and Technology (Eurospeech 2003)*, 597-600, Geneva, Switzerland, 2003.
- Bohus, D. and Rudnicky, A. I. A Principled Approach for Rejection and Threshold Optimization in Spoken Dialogue Systems. In *Proceedings of the Sixth Annual Conference of the International Speech Communication Association (Interspeech 2005)*, Lisbon, Spain, 2005a.
- Bohus, D. and Rudnicky, A. I. Sorry, I didn't Catch That! - An Investigation of Non-understanding Errors and Recovery Strategies. In *Proceedings of the Sixth Special Interest Group on Discourse and Dialogue Workshop (SIGdial 2005)*, Lisbon, Portugal, 2005b.
- Bohus, D. and Rudnicky, A. I. The RavenClaw dialog management framework: Architecture and systems. *Computer Speech and Language*, 23(3): 332-361, 2009.
- Bontempi, G. and Meyer, P. E. Causal filter selection in microarray data. In *Proceedings of the 27th International Conference on Machine Learning (ICML 2010)*, 95-102, Haifa, Israel, 2010.
- Breiman, L. Random forests. *Machine Learning*, 45: 5-32, 2001.
- Bulyko, I., Kirchhoff, K., Ostendorf, M. and Goldberg, J. Error-correction detection and response generation in a spoken dialogue system. *Speech Communication*, 45(3): 271-288, 2005.
- Carlson, R., Hirschberg, J. and Swerts, M. Error handling in spoken dialog systems. *Speech Communication*, 45(3): 207-209, 2005.

- Caruana, R. and Freitag, D. Greedy Attribute Selection. In *Proceedings of the 11th International Conference on Machine Learning (ICML 1994)*, 28-36, New Brunswick, NJ, 1994.
- Chappelle, O., Scholkopf, B. and Zien, A. *Semi-supervised learning*. The MIT Press, 2006.
- Chiang, D., Knight, K. and Wang, W. 11,001 New Features for Statistical Machine Translation. In *Proceedings of the 10th Annual Conference of the North American Chapter of the Association for Computational Linguistics - Human Language Technologies (NAACL-HLT 2009)*, 2180-226, Boulder, Colorado, 2009.
- Clark, H. and Schaefer, E. Contributing to discourse. *Cognitive Science*, 13: 259-294, 1989.
- Clarkson, P. and Rosenfeld, R. *The CMU Statistical Language Modeling (LSM) Toolkit*. (http://www.speech.cs.cmu.edu/SLM_info.html).
- Cohen, J. A coefficient of agreement for nominal scales. *Educational and Psychological Measurements*, 20(1): 37-46, 1960.
- Cronbach, L. J. Coefficient alpha and the interanal structure of tests. *Psychometrika*, 16(3): 297-334, 1951.
- Demsar, J. Statistical Comparisons of Classifiers over Multiple Data Sets. *Journal of Machine Learning Research*, 7: 1-30, 2006.
- Dix, A., Finlay, J., Abowd, G. D. and Beale, R. *Human-Computer Interaction*. Prentice Hall, 3rd Edition, 2003.
- Epstein, S. L., Gordon, J. B., Passonneau, R. J. and Ligorio, T. Toward spoken dialogue as mutual agreement. In *Proceedings of the Association for the Advancement of*

- Artificial Intelligence Workshop on Metacognition for Robust Social Systems*, Atlanta, Georgia, 2010.
- Esseghir, M. A. Effective Wrapper-Filter hybridization through GRASP Schemata. In *Proceedings of the Fourth Workshop on Feature Selection in Data Mining*, 45-54, Hyderabad, India, 2010.
- Fodor, I. K. (2002). A Survey of Dimension Reduction Techniques, Lawrence Livemore National Lab.
- Frank, A. and Asuncion, A. *UCI Machine Learning Repository*. 2010. (<http://archive.ics.uci.edu/ml>). Irvine, CA: University of California, School of Information and Computer Science.
- Gaudel, R. and Sebag, M. Feature Selection as a one-player game. In *Proceedings of the International Conference on Machine Learning (ICML 2010)*, Haifa, Israel, 2010.
- Gordon, J. B. and Passonneau, R. J. An Evaluation Framework for Natural Language Understanding in Spoken Dialogue Systems. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC 2010)*, Valleta, Malta, 2010.
- Guyon, I. and Elisseeff, A. An Introduction to Variable and Feature Selection. *Special Issue on Variable and Feature Selection. Journal of Machine Learning Research*, 3: 1157-1182, 2003.
- Guyon, I., Gunn, S., Nikravesh, M. and Zadeh, L. *Feature Extraction, Foundations and Applications*. Springer, 1st edition, 2006.
- Guyon, I., Weston, J., Barnhill, S. and Vapnik, V. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46(1-3): 389-422, 2002.

- Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P. and Witten, I. H. The WEKA Data Mining Software: An Update. *SIGKDD Explorations*, 11(1), 2009.
- Hall, M. A. Correlation-based Feature Selection for Discrete and Numeric Class Machine Learning. In *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, 359-366, Stanford, Ca, 2000.
- Hollingshead, K. *Formalizing the Use and Characteristics of Constraints in Pipeline Systems*. PhD thesis, Computer Science Department, Oregon Health and Science University School of Medicine, 2010.
- Huggins-Daines, D., Kumar, M., Chan, A., Black, A. W., Ravishankar, M. and Rudnicky, A. I. PocketSphinx: A Free, Real-Time Continuous Speech Recognition System for Hand-Held Devices. In *Proceedings of the Institute of Electrical and Electronics Engineers International Conference on Acoustic, Speech, and Signal Processing (IEEE-ICASSP 2006)* Toulouse, France, 2006.
- Jaiantilal, A. and Grudic, G. Increasing Feature Selection Accuracy for L1 Regularized Linear Models in Large Datasets. In *Proceedings of the Fourth Workshop on Feature Selection in Data Mining*, 86-96, Hyderabad, India, 2010.
- John, G. H., Kohavi, R. and Pfleger, K. Irrelevant Features and the Subset Selection Problem. In *Proceedings of the 11th International Conference on Machine Learning (ICML 1994)*, 121-129, New Brunswick, NJ, 1994.
- Jurafsky, D. and Martin, J. H. *Speech and Language Processing 2 Edition*. Prentice-Hall, 2008.

- Kira, K. and Rendell, L. A. A practical Approach to Feature Selection. In *Proceedings of the Ninth International Workshop on Machine Learning*, 249-256, San Francisco, CA, 1992. Morgan Kaufmann.
- Kohavi, R. and John, G. H. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1-2): 273-324, 1997.
- Kruijff-Korbayova, I., Blaylock, N., Gerstenberger, C., Rieser, V., Becker, T., Kaisser, M., Poler, P. and Schehl, J. An experiment setup for Collecting Data for Adaptive Output Planning in a Multimodal Dialog System. In *Proceedings of the 10th European Workshop on Natural Language Generation (ENLG 2005)*, Aberdeen, 2005.
- leCessie, S. and van Houwelingen, J. C. Ridge Estimators in Logistic Regression. *Applied Statistics*, 41(1): 191-201, 1992.
- Lee, C., Rudnicky, A. I. and Lee, G. G. Let's buy books: finding e-books using voice search. In *Proceedings of the Institute of Electrical and Electronics Engineers Workshop on Spoken Language Technology (IEEE-SLT 2010)*, Berkeley, CA, 2010.
- Levin, E. and Passonneau, R. J. A WOZ variant with contrastive conditions. In *Proceedings of the Interspeech Satellite Workshop, Dialogue on Dialogues: Multidisciplinary Evaluation of Speech-Based Interactive Systems*, 17-21, 2006.
- Ligorio, T., Epstein, S. L. and Passonneau, R. J. Wizards' dialogue strategies to handle noisy speech recognition. In *Proceedings of the Institute of Electrical and Electronics Engineers Workshop on Spoken Language Technology (IEEE-SLT 2010)*, Berkeley, CA, 2010a.

- Ligorio, T., Epstein, S. L., Passonneau, R. J. and Gordon, J. B. What you did and didn't mean. Noise, context and human skill. In *Proceedings of the Annual Meeting of the Cognitive Science Society (CogSci 2010)*, Portland, OR, 2010b.
- Litman, D., Hirschberg, J. and Swerts, M. Characterizing and Predicting Corrections in Spoken Dialogue Systems. *Computational Linguistics*, 32(3): 417-438, 2006.
- Litman, D. and Pan, S. Empirically evaluating an adaptable spoken dialogue system. In *Proceedings of the Seventh International Conference on User Modeling (UM 1999)*, 55-46, 1999.
- Liu, H. and Motoda, H. *Computational Methods of Feature Selection*. Chapman and Hall / CRC Editions, 2008.
- Liu, H., Motoda, H., Setiono, R. and Zhao, Z. Feature Selection: An Ever Evolving Frontier in Data Mining. In *Proceedings of the Fourth Workshop on Feature Selection in Data Mining*, 1-3, Hyderabad, India, 2010.
- Loughrey, J. and Cunningham, P. (2005). Using early-stopping to avoid overfitting in wrapper-based feature selection employing stochastic search. Technical Report TCD-DS-2005-37 Dublin, Trinity College.
- Newman-Norlund, S. E., Noordzij, M. L., Newman-Norlund, R. D., Volman, I. A. C., de Ruiter, J. P., Hagoort, P. and Toni, I. Recipient design in tacit communication. *Cognition*, 111(1): 46-54, 1989.
- Oh, A. H. and Rudnicky, A. I. Stochastic language generation for spoken dialogue systems. In *Proceedings of the Applied Natural Language Processing North American Chapter of the Association for Computational Linguistics Workshop on Conversational Systems (ANLP-NAACL 2000)*, 27-32, Seattle, WA 2000.

- Pado', S., Galley, M., Jurafsky, D. and Manning, C., D. Textual Entailment Features for Machine Translation Evaluation. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, 37-41, Athens, Grece, 2009.
- Passonneau, R. J., Epstein, S. L. and Gordon, J. B. Help Me Understand You: Addressing the Speech Recognition Bottleneck. In *Proceedings of the Association for the Advancement of Artificial Intelligence Spring Symposium on Agents that Learn from Human Teachers*, 119-126, Paolo Alto, CA, 2009a.
- Passonneau, R. J., Epstein, S. L., Gordon, J. B. and Ligorio, T. Seeing what you said: How wizards use voice search results. In *Proceedings of the Sixth Workshop on Knowledge and Reasoning in Practical Dialogue Systems, International Joint Conference on Artificial Intelligence*, Pasadena, CA, 2009b.
- Passonneau, R. J., Epstein, S. L., Ligorio, T. and Gordon, J. B. Embedded wizardry. In *Proceedings of the 12th Special Interest Group on Discourse and Dialogue (SIGdial 2011)*, Portland, OR, 2011.
- Passonneau, R. J., Epstein, S. L., Ligorio, T., Gordon, J. B. and Bhutada, P. Learning about Voice Search for Spoken Dialogue Systems. In *Proceedings of the 11th Annual Conference of the North American Chapter of the Association for Copmutational Linguistics - Human Language Technologies (NAACL-HLT 2010)*, 840-848, Los Angeles, CA, 2010.
- Petrovic, S. and Epstein, S. L. Tailoring a Mixture of Search Heuristics. *Constraint Programming Letters*, 4: 15-38, 2009.
- Platt, J. *Fast Training of Support Vector Machines using Sequential Minimal Optimization*. MIT Press, 1998.

- Quinlan, R. J. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., 1993.
- Ratcliff, J. W. and Metzener, D. Pattern Matching: The Gestalt Approach. *Dr. Dobb's Journal*, 7: 46, 1988.
- Raux, A. and Eskenazi, M. A Multi-layer architecture for semi-synchronous event-driven dialogue management. In *Proceedings of the Institute of Electrical and Electronics Engineers Workshop on Automatic Speech Recognition and Understanding (IEEE-ASRU 2007)*, Kyoto, Japan, 2007.
- Raux, A. and Eskenazi, M. Optimizing Endpointing Thresholds using Dialogue Features in a Spoken Dialogue System. In *Proceedings of the Ninth Special Interest Group on Discourse and Dialogue (SIGdial 2008)*, Columbus, Ohio, 2008.
- Raux, A., Langner, B., Black, A. W. and Eskenazi, M. Let's Go Public! Taking a spoken dialog system to the real world. In *Proceedings of the Sixth Annual Conference of the International Speech Communication Association (Interspeech 2005)*, Lisbon, Portugal, 2005.
- Reunanen, J. Overfitting in making comparisons between variable selection methods. *Journal of Machine Learning Research*, 3: 1371-1382, 2003.
- Rieser, V., Kruijff-Korvayova, I. and Lemon, O. A corpus collection and annotation framework for learning multimodal clarification strategies. In *Proceedings of the Sixth Special Interest Group on Discourse and Dialogue (SIGdial 2005)*, Lisbon, Spain, 2005.
- Rieser, V. and Lemon, O. Using Machine Learning to Explore Human Multimodal Clarification Strategies. In *Proceedings of the Joint Conference of the International*

- Committee on Computational Linguistics and the Association for Computational Linguistics (COLING-ACL 2006)*, 659-666, Sidney, Australia, 2006.
- Rudnicky, A. I. *Logios*. 2008. (<http://www.speech.cs.cmu.edu/tools/>).
- Schatzmann, J., Weilhammer, K., Stuttle, M. and Young, S. A Survey of Statistical User Simulation Techniques for Reinforcement Learning of Dialogue Management Strategies. *The Knowledge Engineering Review*: 1-24, 2006.
- Seneff, S., Lau, R., Pao, C. and Zue, V. Galaxy-II: A Reference Architecture. In *Proceedings of the Fifth International Conference on Spoken Language Systems (ICSLP 1998)*, Sydney, Australia, 1998.
- Sherwani, J., Yu, D., Paek, T., Czerwinski, M. and Acero, A. VoicePedia: Towards speech-based access to unstructured information. In *Proceedings of the Eight Annual Conference of the International Speech Communication Association (Interspeech 2007)*, Antwerp, Belgium, 2007.
- Skantze, G. Exploring human error recovery strategies: Implications for spoken dialog systems. *Speech Communication*, 45(3): 325-341, 2005.
- Skantze, G. and Edlund, J. Early error detection on word level. In *Proceedings of the International Speech Communication Association Tutorial and Research Workshop on Robustness Issues in Conversational Interaction*, Munich, Germany, 2004.
- Skantze, G. and Gustafson, J. Attention and interaction control in a human-computer dialogue setting. In *Proceedings of the Tenth Annual Meeting of the Special Interest Group in Dialogue and Discourse (SIGdial 2009)*, London, England, 2009.
- Stoyanchev, S. and Stent, A. Predicting Concept Types in User Corrections in Dialog. In *Proceedings of the European Chapter of the Association for Computational*

- Linguistics Workshop on the Semantic Representation of Spoken Language (EACL-SRSL 2009)*, 42-49, Athens, Greece 2009.
- Stracuzzi, D. J. and Utgoff, P. E. Randomized Variable Elimination. *Journal of Machine Learning Research*, 5: 1331-1362, 2004.
- Tibshirani, R. Regression Shrinkage and Selection via the Lasso. *Journal of the Royal Statistical Society.*, 58(1): 267-288, 1996.
- Torres, F., Hurtado, L. F., Garcia, F., Sanchis, E. and Segarra, E. Error handling in a stochastic dialog system through confidence measures. *Speech Communication*, 45(3): 211-229, 2005.
- Walker, M. A., Litman, D. J., Kamm, C. A. and Abella, A. PARADISE: A Framework for Evaluating Spoken Dialog Agents. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics (ACL 1997)*, 271-280, Madrid, Spain, 1997.
- Ward, W. and Issar, S. Recent improvements in the CMU spoken language understanding system. In *Proceedings of the Advanced Research Projects Agency Human Language Technology Workshop*, Plainsboro, NJ, 1994.
- Wilcock, G. and Jokinen, K. Adaptivity and Response Generation in a Spoken Dialogue System. J. van Kuppevelt and R. Smith (ed.) *Current and New Directions in Discourse and Dialogue (Text, Speech and Language Technology)*, 213-234. Kluwer Academic Publishers, 2003.
- Williams, J. D. and Young, S. Characterizing Task-Oriented Dialogue using a Simulated ASR Channel. In *Proceedings of of the Fifth Annual Conference of the International Speech Communication Association (Interspeech 2004)*, Jeju Island, Korea, 2004.

- Xiong, D., Zhang, M. and Li, H. Error Detection for Statistical Machine Translation Using Linguistic Features. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, 604 - 611, Uppsala, Sweden, 2010.
- Yao, X., Bhutada, P., Georgila, K., Sagae, K., Arstein, R. and Traum, D. Practical evaluation of speech recognizers for virtual human dialogue systems. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC 2010)*, Valletta, Malta, 2010.
- Yu, L. and Liu, H. Efficient Feature Selection via Analysis of Relevance and Redundancy. *Journal of Machine Learning Research*, 5: 1205 - 1224, 2004.
- Zhao, Z. and Liu, H. Spectral feature selection for supervised and unsupervised learning. In *Proceedings of the 24th International Conference on Machine Learning (ICML 2007)*, 1151-1157, Corvallis, OR, 2007.
- Zollo, T. A study of human dialogue strategies in the presence of speech recognition errors. In *Proceedings of the Association for the Advancement of Artificial Intelligence Fall Symposium on Psychological Models of Communication in Collaborative Systems*, 132-139, Arlington, VA, 1999.