VoiceBrowse: The Dynamic Generation Of Spoken Dialogue from Online Content

Craig Wootton

BSc. (Hons) Computing Science
***with* Dip. Industrial Studies**

Faculty of Computing and Engineering of the University of Ulster

Thesis submitted for the degree of Doctor of Philosophy

October 2008

Declaration

"I hereby declare that with effect from the date on which the thesis is deposited in the Library of the University of Ulster, I permit the Librarian of the University to allow the thesis to be copied in whole or in part without reference to me on the understanding that such authority applies to the provision of single copies made for study purposes or for inclusion within the stock of another library.

This restriction does not apply to the British Library Thesis Service (which is permitted to copy the thesis on demand for loan or sale under the terms of a separate agreement) nor to the copying or publication of the title and abstract of the thesis.

IT IS A CONDITION OF USE OF THIS THESIS THAT ANYONE WHO CONSULTS IT MUST RECOGNISE THAT THE COPYRIGHT RESTS WITH THE AUTHOR AND THAT NO QUOTATION FROM THE THESIS AND NO INFORMATION DERIVED FROM IT MAY BE PUBLISHED UNLESS THE SOURCE IS PROPERLY ACKNOWLEDGED".

Acknowledgements

I wish to acknowledge several people who have helped guide, support and direct me throughout this research.

Firstly, I wish to thank Prof. Mike McTear for sharing his extensive knowledge of the subject area, and for being generous with both his time and involvement. Prof. McTear has been a great teacher, mentor, advisor and friend these last three years, and I found myself both fortunate and privileged to have spent time in learning under his guidance. I wish him all the best for his future retirement, and for forthcoming opportunities he will pursue into this new period of his fruitful career, characteristic of his continuous thirst of discovering new knowledge.

Thanks also to Prof. Terry Anderson, who played the supporting role to Mike in supervising me ever so well. My thanks to Prof. Anderson for his giving of time in meetings, in proof reading, in the little nuggets of advice and ideas he continuously had to offer, and also in the answering of all the little XML and XPATH related problems I burdened him with.

Thanks also to Prof. Sebastian Möller, his team of researchers, and in particular Klaus-Peter Engelbrecht. My thanks to them all for shaping and moulding the evaluation phase of the research with their expertise in the area, and for their help also during the analysis phase.

Thanks to the body of researchers and friends in ‘J26’, who have continuously offered the needed social breaks and coffees during these last three years – the support role that the room plays to both me and to one another cannot be undermined. Thanks also to all

my friends outside of University and in particular those at McQuiston, who have encouraged me and supported me throughout this study.

Thanks also to my 32 evaluation subjects who so willingly gave of their time to help me complete the research.

Lastly, thanks to my family, in particular to my Mum and Dad, who have supported, encouraged, provided for and sustained me during my seven years in full time education. Their love and guidance has been valuable to me, and I thank them for shaping me into the person I am today.

Thanks
Craig

Craig Wootton, University of Ulster

Abstract

Most dynamic spoken dialogue systems operate with specifically structured task or domain knowledge using dialogue management strategies that are either hand-crafted or learned from data. To date this has limited interactions using such systems to a sole domain, so that a dialogue manager that can interact generically with multiple types of content from various structured and unstructured sources has yet to be fully realised.

This thesis describes VoiceBrowse, a dynamically evolving dialogue system that enables users to access online content. Whereas in previous such systems the online content was restricted to a small number of specific websites, the current system is capable of interacting with various unstructured online contents, irrespective of source or type. This is made possible by the inclusion of a novel component known as the Content Manager that provides access to a wide range of online materials using live RSS feeds and real-time API-based techniques. Ideas from information retrieval are incorporated to dynamically select the source(s) that best match the user's requirements.

An evaluation study tests the usability and performance of VoiceBrowse with respect to different user groups interacting with two different versions of the system. Analysis of the data highlights significant differences between the groups and systems, contributing to current dialogue usability research that has been generally restricted to task based dialogues.

Contents

Title Page i
Declaration ii
Acknowledgements iii
Abstract v
Contents vi

List of Figures ix
List of Tables x
List of Code Excerpts xi
Glossary of Technical Terms xii

Chapter 1: Introduction 15
1.1 Research Area and Issues 15
1.2 Research Aims and Objectives 17
1.4 Thesis overview and Outline 18

Chapter 2: Spoken Dialogue Systems: Overview 20
2.1 Spoken Dialogue Systems: Introduction 20
2.2 Spoken Dialogue Systems: Typical Components and Architecture 21
2.3 Spoken Dialogue Systems: Dialogue Manager 22
2.4 Spoken Dialogue Systems: Advantages 28
2.5 Spoken Dialogue Systems: Limitations 30
2.6 Spoken Dialogue Systems: Multimodal Dialogue Systems 32
2.7 Spoken Dialogue Systems: Evaluation 35
2.8 Spoken Dialogue Systems: Usability Considerations 38
2.9 Enabling technologies 41
2.9.1 Enabling Technologies: XML 42
2.9.2 Enabling Technologies: XML Applications 44
2.9.3 Enabling Technologies: Voice technologies 46
2.10 Spoken Dialogue Systems: Advanced Architectures 49
2.10.1 Advanced Architectures: Queen's Communicator 50
2.10.2 Advanced Architectures: JASPIS Architecture 54
2.10.3 Advanced Architectures: CONVERSE Architecture 55
2.10.4 Advanced Architectures: RAVENCLAW Architecture 56
2.10.5 Advanced Architectures: Research Issues 57

2.11 Summary 59
Chapter 3: Advanced Dialogue Research 60
3.1 Spoken Dialogue Systems: Dynamic Dialogue Systems 60
3.1.1 Dynamic Dialogue Systems: Utilising Structured Content 62
3.1.2 Dynamic Dialogue Systems: Utilising Unstructured Online Content 66
3.1.3 Dynamic Dialogue Systems: Research Issues 72
3.2 Adaptive Spoken Dialogue Systems 75
3.2.1 Adaptive Dialogue Systems: Adapting Dialogue 76
3.2.2 Adaptive Dialogue Systems: Introduction to User Modelling 78
3.2.3 Adaptive Dialogue Systems: Adapting Content 80
3.2.4 Adaptive Dialogue Systems: Research Issues 83
3.3 Information Retrieval 85
3.4 Summary 87
Chapter 4: VoiceBrowse Introduction and Architecture 89
4.1 VoiceBrowse: Requirements Derived From Research Gaps 89
4.2 VoiceBrowse: Applications and Users 91
4.3 VoiceBrowse: Use Cases and Example Dialogues 91
4.4 VoiceBrowse: Architecture 96
4.5 VoiceBrowse: Utilising RSS and API feeds 98
4.6 VoiceBrowse: User Manager 102
4.7 VoiceBrowse: Dialogue Manager 103
4.8 VoiceBrowse: Content Manager 105
4.9 VoiceBrowse: Research focus 105
4.10 Summary 106
Chapter 5: VoiceBrowse Content Manager 108
5.1 Content Manager: Introduction 108
5.2 Content Manager: Issues and Concerns 110
5.3 Content Manager: Design and Process 112
5.4 Content Manager: Content Spotter Evaluation and Enhancements 114
5.5 Summary 121
Chapter 6: VoiceBrowse Dialogue Manager 122
6.1 Dialogue Manager: Introduction 122
6.2 Dialogue Manager: Issues and Concerns 124

6.3 Dialogue Manager: Proposed Dialogue Strategies 126
6.4 Dialogue Manager: Handling User Inputs 128
6.5 Dialogue Manager: Handling System Outputs 130
6.6 Summary 132
Chapter 7: VoiceBrowse Implementation 135
7.1 Current Dialogue Technologies 135
7.2 VoiceBrowse Implementation: VoiceXML and Call Flow Diagrams 139
7.3 VoiceBrowse Implementation: Prompt Design 147
7.4 VoiceBrowse Implementation: Content Manager 154
7.5 VoiceBrowse Implementation: Dialogue Manager 161
7.6 VoiceBrowse Implementation: Information Based Dialogues 165
7.7 VoiceBrowse Implementation: Delivery of Online Content 176
7.8 VoiceBrowse Implementation: Task Based Dialogues 183
7.9 VoiceBrowse Implementation: Challenges and Issues Encountered 189
7.10 VoiceBrowse Implementation: Example Dialogues 194
7.11 Summary 200
Chapter 8: VoiceBrowse Evaluation 202
8.1 VoiceBrowse Evaluation: Design 202
8.2 VoiceBrowse Evaluation: Results 206
8.3 VoiceBrowse Evaluation: Discussion 222
8.3.1 Discussion: Hypothesis 1, Hypothesis 4 and Hypothesis 5 223
8.3.2 Discussion: Hypothesis 2 228
8.3.3 Discussion: Hypothesis 3 232
8.4 VoiceBrowse Evaluation: Conclusions 234
Chapter 9: VoiceBrowse Conclusions 238
9.1 VoiceBrowse Conclusions: Summary of Thesis 238
9.2 VoiceBrowse Conclusions: Summary of Research Contributions 239
9.2 VoiceBrowse Conclusions: Future Work 241
References 244
Appendix A: Publications 258
Appendix B: Evaluation Material 259
Evaluation Design 259
Evaluation Schedule 260
Pre-Screening questionnaire 261

Initial Questionnaire....262
Evaluation of the Interaction....263
Evaluation of the System....271
Evaluation scenarios....272
Appendix C: Evaluation Results....274
Interaction Parameters (Quantitative) Results....274
Qualitative (Questionnaire) Results....277

List of Figures

Figure 2.1: Typical Spoken Dialogue System Architecture....22
Figure 2.2: DARPA Architecture....49
Figure 2.3: Queen's Communicator Architecture....51
Figure 2.4: Queen's Communicator Dialogue Manager....52
Figure 2.5: JASPIS Architecture....54
Figure 2.6: CONVERSE Architecture....56
Figure 2.7: RavenClaw Dialogue Manager in RoomLine....57
Figure 3.1: Ritel Architecture....65
Figure 3.2: Customisable Phone Access to Personal Information....69
Figure 3.3: WebTalk Architecture....71
Figure 3.4: Adaptive Place Advisor Architecture....82
Figure 4.1: VoiceBrowse Use Case – System Functionality....92
Figure 4.2: VoiceBrowse Use Case - Available Dialogues Types....92
Figure 4.3: VoiceBrowse Use Case - Information-Based Dialogues....93
Figure 4.4: VoiceBrowse Use Case – Task-Based Dialogues....93
Figure 4.5: VoiceBrowse Dialogue....94
Figure 4.6: VoiceBrowse Architecture....96
Figure 4.7: An Example RSS Feed....99
Figure 5.1: Content Manager....108
Figure 5.2: An Example RSS Feed....112
Figure 5.3: Document Space Creation From RSS Feeds....113
Figure 5.4: Content Spotter Process....114
Figure 5.5: Comparison of Calculations tw1 and tw2....119
Figure 5.6: Comparison of Calculations tw1 and tw3....120
Figure 6.1: Dialogue Manager....123

Figure 6.2: Grammar Creation In The Dialogue Manager.....129
Figure 6.3: Outputting Content To The User.....131
Figure 6.4: VoiceBrowse Workflow.....133
Figure 7.1: VoiceBrowse Environment.....142
Figure 7.2: VoiceBrowse Call Flow Diagram.....143
Figure 7.3: VoiceBrowse Revised Call Flow Diagram.....146
Figure 8.1: Scenario Duration w.r.t. User Group.....211
Figure 8.2: Number <noinput> w.r.t. User Group.....211
Figure 8.3: Number of Barge-Ins w.r.t. User Group.....211
Figure 8.4: Number of Help Requests w.r.t. User Group.....211
Figure 8.5: Number of New Queries w.r.t. User Group.....212
Figure 8.6: Number of <nomatch> w.r.t. User Group.....212
Figure 8.7: Number of Turns per Scenario w.r.t. User Group.....212
Figure 8.8: Overall Rating w.r.t. User Group.....212
Figure 8.9: Efficiency Rating w.r.t. User Group.....213
Figure 8.10: SASSI Annoyance w.r.t. User Group.....213
Figure 8.11: SASSI Cognitive Demand w.r.t. User Group.....213
Figure 8.12: SASSI Likeability w.r.t. User Group.....213
Figure 8.13: SASSI Accuracy w.r.t. User Group.....214
Figure 8.14: Efficiency Rating w.r.t. User Group.....14
Figure 8.15: SASSI Annoyance w.r.t. User Group.....214
Figure 8.16: SASSI Cognitive Demand w.r.t. User Group.....214
Figure 8.17: SASSI Likeability w.r.t. User Group.....215
Figure 8.18: SASSI Accuracy w.r.t. User Group.....215
Figure 8.19: Efficiency Rating w.r.t. User Group.....215
Figure 8.20: SASSI Annoyance w.r.t. User Group.....215
Figure 8.21: SASSI Cognitive Demand w.r.t. User Group.....216
Figure 8.22: SASSI Likeability w.r.t. User Group.....216
Figure 8.23: SASSI Response Accuracy w.r.t. User Group.....216

List of Tables

Table 5.1: Preliminary Experiment Setup.....115
Table 5.2: Preliminary Experiment Results.....116
Table 5.3: Enhanced COSIM Experiment Results.....118
Table 7.1: VoiceBrowse Prompt Design.....149

Table 7.2: VoiceBrowse Pseudo Code For Information Based Dialogues.....166
Table 7.3: Voxeo RecordCall Attribute Specification.....168
Table 7.4: VoiceBrowse Pseudo Code For Outputting Content.....177
Table 7.5: VoiceBrowse Pseudo Code For Task Based Dialogues.....184
Table 7.6: Sample Dialogues With VoiceBrowse.....194
Table 8.1: Usability Category Definitions.....206
Table 8.2: Questionnaire Results.....209
Table 8.3: Interaction Parameters.....210

List of Code Excerpts

Code Excerpt 1: API VoiceBrowse Specification.....155
Code Excerpt 2: VoiceBrowse API Specification.....156
Code Excerpt 3: Function to Produce List of Current Feeds.....157
Code Excerpt 4: Function to Produce Document Space.....159
Code Excerpt 5: Function to Fetch Body of Content From Source URL.....160
Code Excerpt 6: PHP To Save <record> Audio Data.....161
Code Excerpt 7: Function to Start Speech Recognition Process.....162
Code Excerpt 8: Visual Basic.Net Speech Recognition Function.....162
Code Excerpt 9: Function to Add new Dialogue History Record to Database.....164
Code Excerpt 10: Function to Get Previous Dialogue State From History.....164
Code Excerpt 11: ASP.Net Response Statement.....165
Code Excerpt 12: Voxeo Proprietary RecordCall Element.....168
Code Excerpt 13: informationStart.aspx In Closed Version.....169
Code Excerpt 14: Function to Match User Utterance To Available Feeds.....171
Code Excerpt 15: Outputting Matched Feeds to User and Waiting For Input.....174
Code Excerpt 16: Transitions to Informative or Task Based Dialogues.....175
Code Excerpt 17: informationStart.aspx In Open Version.....176
Code Excerpt 18: Speech Recognition On Saved .Wav File.....176
Code Excerpt 19: Cosine Similarity Function of Content Spotter.....179
Code Excerpt 20: Outputting Matched Documents To The user.....179
Code Excerpt 21: Fetching Content Body From URL.....181
Code Excerpt 22: Outputting Main Content Body To User.....182
Code Excerpt 23: Collecting API Parameter From User.....185
Code Excerpt 24: Confirming API Parameter Uttered By User.....186
Code Excerpt 25: Outputting API Results To user.....187

Glossary of Technical Terms

.Net Framework: Software technology that is available with several Microsoft Windows operating systems, which includes a large library of pre-coded solutions to common programming problems and a virtual machine that manages the execution of programs.

Adaptive Dialogue System: A spoken dialogue system that adjusts its outputs and dialogue management to best meet the user's needs.

Application Programming Interface (API): A set of functions, procedures, methods, classes or protocols that an operating system or service provides to support requests made by a software program.

Architecture: The structure of a system, comprising of multiple interacting software components.

Automatic Speech Recognition Engine (ASR): Software that performs speech recognition.

Cosine Similarity (COSIM): A measure of similarity between two vectors of n dimensions by finding the cosine of the angle between them. When used to compare text documents, the vectors represent terms found in the documents, specified by their tf-idf.

Dialogue Knowledge: The rules and information with respect to how the dialogue should be executed between parties, such as verification strategies, available dialogue strategies and initiatives.

Dialogue Manager: Software that implements the dialogue rules written by a developer to encourage interaction with a user. Usually performs the processing in a dialogue system.

Dialogue System: A computer system intended to converse with a human, primarily through natural language.

Domain Knowledge: Structured representation of the content to be utilised by the dialogue manager. Typical specifications includes database format, XML format and ontologies.

Dynamic Dialogue System: A spoken dialogue system that is not static in nature, with grammars and outputs usually created during runtime.

Grounding: Establishing beliefs between parties in a dialogue.

Grammar: A set of word patterns that tells an ASR what utterances are allowed as input.

Information Retrieval: The science of searching for documents, for information within documents and for metadata about documents, as well as that of searching relational databases and the World Wide Web

Interaction Parameters: Quantitative metrics of dialogue, such as dialogue length, word error rate and number of user turns.

Language Generation Engine: Software that creates a natural language utterance to be output to the user that furthers the dialogue.

Language Understanding Engine: Software that accepts text as input and infers the underlying meaning to the utterance.

Microsoft SQL Server: A popular database management system.

Multimodal Dialogue Systems: A dialogue system that can accept more than one mode of input and/or output, such as combining speech with graphics, gesture recognition and emotion recognition.

Named Entity: An atomic element that is found in a sentence, often a Proper Noun.

Narrative Based Dialogue: A dialogue between two or more parties that often does not have a set task to complete, is usually based upon descriptive content such as news and evolves in an opportunistic, unspecified manner.

Online Content: Information held on online source, such as web pages.

Really Simple Syndicate (RSS): A XML based formats used to publish frequently updated works in a standardized specification.

Semantic Web: An evolving extension of the World Wide Web in which the semantics of information and services on the web is defined, making it possible for the web to understand and satisfy the requests of people and machines to use the web content

Spoken Dialogue System: A dialogue system that interacts through voice, accepting spoken inputs from the user and presents output in spoken form.

Speech Recognition: Accepting speech input from a user and transcribing the utterance to written text. Usually seen as the input to a spoken dialogue system, accepting the spoken input by a user and converting this to written text for processing.

Speech Recognition Grammar Specification (SRGS): A W3C standard for specifying speech recognition grammars. Can take a variety of forms, including XML.

Speech Synthesis Mark-up Language (SSML): W3C standard for specifying synthesised content, which provides a standard way of controlling aspects such as pitch, volume, rate etc.

Task Based Dialogue: A dialogue between two or more parties that has a common goal or task to accomplish.

Term Frequency–Inverse Document Frequency (tf-idf): A statistical measure used to evaluate how important a word is to a document in a collection or corpus. The importance increases proportionally to the number of times a word appears in the document but is offset by the frequency of the word in the corpus.

Text-To-Speech Engine (TTS): Converts written text to speech for output. Usually performs the output of a dialogue system, converting the text generation by the language generation engine into spoken output.

Usability: The ease with which people can employ particular software in order to achieve a particular goal.

User model: A representation of the user’s interests and interaction history created to refine dialogue and search results.

VisualBasic.Net: An object-oriented computer language which is an evolution of Microsoft's Visual Basic and is implemented on the Microsoft .NET framework

VoiceXML: The W3C's XML based standard format for specifying interactive voice dialogues between a human and a computer. It allows voice applications to be developed and deployed in a similar way that HTML is used for visual applications.

Voxeo Prophecy: Platform for implementing a VoiceXML system, including ASR and TTS components.

WordNet: A database of words that groups words into sets of synonyms called synsets, provides short, general definitions, and records the various semantic relations between these synonym sets.

XML: The Extensible Mark-up Language is a general-purpose specification for creating custom mark-up languages.

Chapter 1: Introduction

The development of Human-Computer Interfaces to incorporate both speech and natural language technologies has been a goal that is still to be truly realised. Spoken dialogue research is an area of research that seeks to understand and advance work in this area of natural interfaces, integrating the inputs and outputs of Automatic Speech Recognition (ASR) and Text-To-Speech (TTS) technologies with language understanding and dialogue components.

1.1 Research Area and Issues

Recent work and innovation in spoken dialogue systems has started to emerge into the public domain, and some industries and companies have already identified the commercial benefits of such systems. First generation spoken dialogue systems deployed in industry could be thought of as a rudimentary but an effective form of interaction - allowing the completion of tasks such as booking and purchasing services, reviewing bills and payments, and directory assistance.

Academic dialogue research meanwhile endeavours to advance these initial dialogue systems into more natural and flexible systems. Such systems can adapt to meet the particular needs of the current user, evolve 'on-the-fly' to create more dynamic and variable dialogues, and engage in more complex dialogue to help the user accomplish a particular task. Others areas of Artificial Intelligence have also merged with Spoken Dialogue research to further this work even more, such as the utilisation of machine and reinforcement learning techniques to optimise system design and performance.

For these advanced features of dialogue systems to function effectively, it is often the case that dialogue knowledge and domain knowledge are separated from one another –

consequently the representation of the domain knowledge is required to be specifically structured for and accessible to the dialogue manager. Structures such as databases or ontologies, and associated query languages, are often used to store and query the domain knowledge, which the dialogue manager can then utilise when needed.

Due to the domain representation being well defined and specifically for one domain, dialogue managers cannot generically interact with more than one structure or domain. This has led to a current lack of multi-purpose and multi-domain dialogue managers that can interact with various representations of domains. For example, a well defined database for an airline reservation system is created specifically for that domain and purpose, and will therefore be remarkably different in structure to a pizza ordering system.

Contrast this with the Internet and the largely unstructured nature of the online documents available. Although HTML provides a syntactically well defined specification instructing a graphical browser how to render the web page visually, it provides no semantic or presentational information, and each web page can be represented differently with respect to the HTML. This has been a major issue and challenge for dialogue researchers wishing to complement the graphical browser with a dialogue interface to online content - dialogue systems are traditionally created to interact with a specific content type and structure, unlike online content which typically goes beyond such requirements. This is one aspect of the research challenge to be addressed in this dissertation - that of generically accessing online content from various unstructured sources and utilising them in dialogue.

A second aspect of the research is concerned with usability. Usability has been an important issue for many years during the development of the computer and even more

so since the evolution of the graphical user interface. The concept of making a piece of software 'usable' for different users and their needs is highly important also for dialogue engineers, due to the increasing emergence of dialogue systems deployed in industry and the associated public awareness of such systems.

However dialogue systems present a number of additional challenges for usability engineers - inputs have to be constrained to comply with the limitations of speech recognisers and the constraints of specially constructed language models; outputs need to be relevant and meaningful to the user without being cognitively difficult; the functionality of the system needs to be obvious to the user; and error and confirmation strategies must provide an easy way of recovery in the event of mis-understandings and non-understandings.

Usability considerations are made somewhat more complex for information-based dialogues – interactions that aim to provide news and similar information to the user instead of completing a set task. At present there is a current lack of usability studies with regards to information-based dialogues.

1.2 Research Aims and Objectives

The aim of this research is to further the knowledge and contribution of dynamic dialogue systems with regard to browsing the Internet through voice. Objectives to attain this aim can be summarised as:

- Explore the literature on dynamic spoken dialogues.
- Identify and address the main issues and challenges regarding browsing online content through dialogue.

- Develop a dialogue system that interfaces with the Internet, overcoming the shortfalls identified in the literature review.
- Evaluate and measure the performance of the developed dialogue system, assessing its technical qualities and its potential in terms of usability for end users.

It is the thesis of this research that for dialogue systems to interface with the Internet, there are both technical and usability challenges that need to be addressed. The VoiceBrowse system has been designed and implemented to overcome current shortcomings and limitations in these specific areas, which can be summarised as:

- The requirement of specifically structured domain knowledge purposely crafted for each dialogue manager. VoiceBrowse has been developed to make use of on online content from different sources that do not conform to a standard structure.
- That dynamic dialogue systems have traditionally been developed for dialogue in a specific domain. VoiceBrowse has been developed to interact generically in various domains and content types that evolve over time.
- That there has not been the same degree of usability research with respect to opportunistic information-based dialogues when compared to traditional task-based dialogue. VoiceBrowse will further this field of dialogue usability study, specifically for browsing the Internet through a dialogue interface.

1.4 Thesis overview and Outline

The area of spoken dialogue systems will first be introduced and reviewed in Chapter 2. More advanced dialogue research relevant to the aims and goals of this research will be surveyed in Chapter 3. Current limitations and gaps in the literature will be identified

and summarised, leading to a requirement specification of VoiceBrowse presented in Chapter 4.

The VoiceBrowse architecture will then be discussed, outlining its features and components that overcome the shortcomings identified with respect to browsing the Internet through dialogue. Chapters 5 and 6 then present and explain in detail the two components of VoiceBrowse which have been the focus of this research, namely the Dialogue Manager and the Content Manager. The role, purpose and functionality of both are introduced, and the generic nature of the dialogue manager that enables it to interact with a wide range of content types is further explored. The Content Manager is the most novel component of VoiceBrowse, enabling the Dialogue Manager to interact with a range of content types. A detailed description of the Content Spotter is included here, and the Information Retrieval mechanisms and techniques that are used here are also explained.

Chapter 7 discusses the implementation of VoiceBrowse in detail, specifically the technical challenges associated with realising the VoiceBrowse architecture. The technical and usability requirements that were specified in Chapter 2 are then used as a basis for evaluation in Chapter 8, along with a discussion and analysis of the results. Conclusions and a summary of the contributions to knowledge are summarised in Chapter 9, together with suggestions for future research.

Chapter 2: Spoken Dialogue Systems: Overview

The following chapter presents the area of spoken dialogue systems in order to introduce the subject area and establish key issues. A typical architecture and its components will be briefly discussed before specifically looking at the Dialogue Manager in particular. Advantages and limitations of such systems will be considered, along with other areas of importance, such as matters regarding the evaluation and usability of spoken dialogue systems. Enabling technologies will be presented, before an overview of more advanced systems concludes the chapter.

2.1 Spoken Dialogue Systems: Introduction

> *"Speech is the ultimate, ubiquitous interface. It is how we should be able to interact with computers. The question is when should it begin supplementing the keyboard and mouse? We think the time is now."*
>
> (Armstrong 1994)

Being the most natural form of interaction between humans, speech has been, and will be, the dominant mode of human social bonding and information exchange (Huang et al. 2001). Since the 1950s, Artificial Intelligence researchers have sought to bring a speech interface to the computer (Rabiner & Juang 1993). Traditional interfaces, however, between humans and computers, have evolved into a graphical based approach, commonly known as the Graphical User Interface (GUI). Rather than the machines adapting to humans, humans have adapted to a form of interaction suited for machines, and graphical interfaces have become a very usable and widely accepted standard for interaction.

Spoken dialogue systems provide a natural language based alternative to the graphical interface. Jurafsky & Martin (2008) define spoken dialogue systems as programmes that communicate with users in spoken natural language in order to accomplish tasks such as making travel arrangements, or answering questions. It important to make a distinction between spoken dialogue systems and other types of speech based systems, such as voice control, call routing, or voice search systems. Systems such as these are more restricted with respect to dialogue, generally using speech recognition to translate spoken words onto a finite set of options. Spoken dialogue systems include additional components, such as language understanding and dialogue management, collaborating with the user in dialogue to solve a common goal.

Language understanding is a highly challenging area of research. Although there have been great advances in speech recognition technology in previous years it is still very difficult to extract the underlying meaning, or semantics, of what has been spoken. Traditional approaches have been to look solely at the words and syntax of the utterance spoken by the user, but future directions in this area will see this information incorporated with other information, such as prosody, dialogue history and pragmatic salience (Bangalore et al 2006).

2.2 Spoken Dialogue Systems: Typical Components and Architecture

A spoken dialogue system consists of a number of components that need to interact with each other in order for the whole dialogue system to function successfully. The components are from a wide variety of disciplines outside dialogue research, such as speech recognition, natural language understanding, and natural language generation.

A typical spoken dialogue architecture, illustrated in Figure 2.1, includes a speech recognition engine that recognises the input from the user - once captured, the

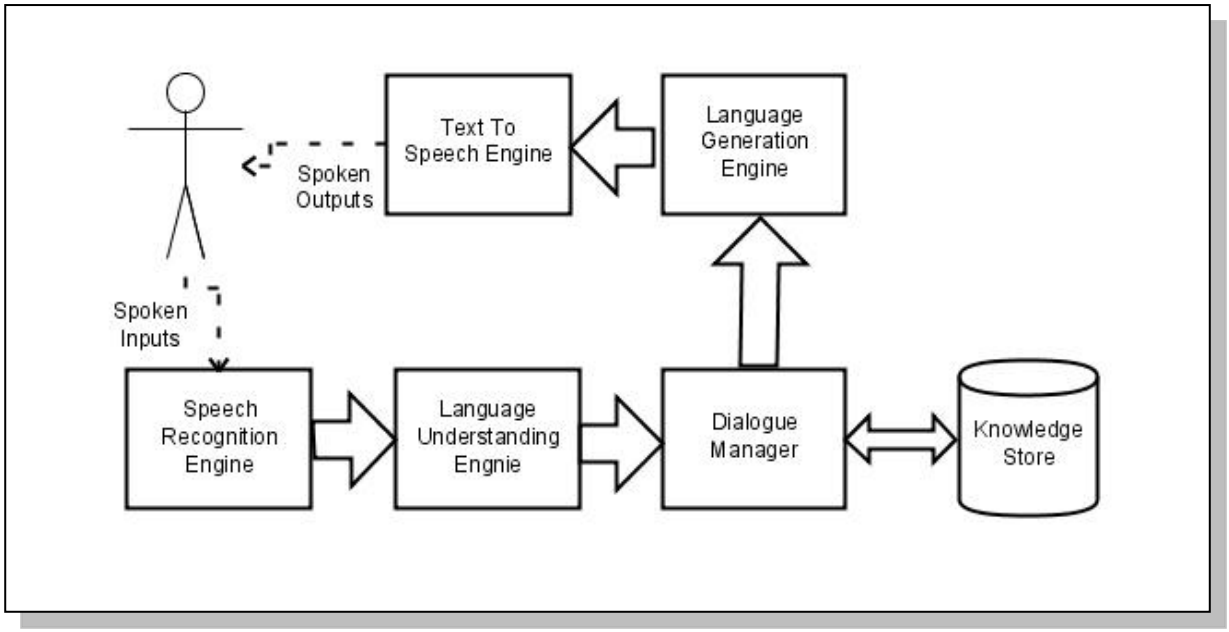


Figure 2.1: Typical Spoken Dialogue System Architecture

recognised words have to be parsed and understood by the system, which is the task of the language understanding engine. It is the role of the dialogue manager to decide what to do next in the interaction, based on the meaning of the words extracted and the current state of the dialogue. More information may be needed from the user before continuing, or perhaps a confirmation is required. This response is taken by a language generation engine that will formulate an appropriate, meaningful sentence to be output to the user. Finally, the text-to-speech engine will take the sentences produced by the language generator and covert them into a spoken form that can be output by the audio player of the system. This particular research project focuses on the dialogue management aspect of the overall architecture.

2.3 Spoken Dialogue Systems: Dialogue Manager

The Dialogue Manager is primarily concerned with ensuring the conversational flow between human and machine, ensuring that the dialogue is coherent with the task to be

accomplished and with the user. Möller (2005) defines the core functions that are to be provided by the dialogue manager;

- The collection of information that is required for the task to be completed.
- Interpretation of complex discourse.
- The organisation of meaningful outputs to the user.
- To provide help to the user when needed.
- The distribution of dialogue initiative.
- To provide relevant feedback to the user, verifying the information understood by the system (grounding).
- Correction strategies for errors and misunderstandings.

Typical spoken dialogue systems are created to accomplish a specific task, such as booking a flight or cinema ticket. It is the dialogue manager that, once given knowledge of this task, will output the appropriate questions to the user to extract the needed information from them, such as departure and destination airports. In some dialogue systems, more complex, natural language inputs are allowed from the user, and the dialogue manager, along with a language understanding component, will interpret and extract the meaning and intent from the user's phrase. Similarly, on the output side of the dialogue, the dialogue manager must arrange and speak the information back to the user in a meaningful and appropriate way. As discussed later with regard to usability (see Section 2.8), there are limitations and constraints to the methods of outputting information to the user. Various help functions should also be available to increase the usability of a dialogue system, and it is the role of the dialogue manager to

record and track the history of the interaction, and offer help when it decides the user requires assistance. This should also be available on demand for the user.

Traditional approaches that have been used to implement these functions include dialogue grammars, plan-based approaches and collaborative approaches (Churcher 1997). A similar classification of implementation approaches provided by McTear defines the categories as finite state, frame based, and agent based dialogue management (McTear 2004a: 91).

Dialogue grammar and finite state based dialogues are the most rigid and inflexible of the different approaches. They can be represented by state transition networks where the nodes represent information that must be elicited from the user, and the transitions are the various paths through the dialogue that are selected based on the semantic interpretation of the user's utterance that is limited to a range of keywords and phrases specified in a Context-Free-Grammar (CFG). Simplistic but effective, they are however very inflexible as the paths through the interaction are fixed for each dialogue.

Frame based dialogues allow for some more flexibility than finite state approaches, allowing the users to over-specify answers and to avoid being tied into a fixed path through the interaction. The dialogue is represented as a hierarchical form containing slots, and each slot must be filled for the dialogue manager to deem a certain task as completed. If the user provides values for more than one slot, then the associated values can be placed into their respective slots. The dialogue, however, is still inflexible, as no form of conversation or true negotiation can take place between user and machine. For this more advanced type of dialogue, agent based control is required.

Drawings on techniques from Artificial Intelligence, such as game theory and planning, agent based and plan based approaches seek to overcome the rigidity and inflexibility

associated with both finite state and frame based approaches. They are based on the observation that humans plan their actions to achieve various goals (Huang et al. 2001; Allen et al. 2001). They permit a more flexible and conversation like interaction, however they can be complex to construct and operate in practice. Alternatively, instead of concentrating on the structure of the task, collaborative based approaches attempt to capture the generic nature of dialogue and the beliefs of both user and system. This approach also requires more sophisticated natural language understanding and interpretation techniques.

Recently, however, statistical approaches have been applied in dialogue management to determine the system's response (Griol et al. 2008). However, the main limitation of these approaches is their need for large corpora of data. Machine learning has also been used in the field of dialogue study (Lemon et al. 2006, Litman et al. 2000; Litman et al. 1999), with transformation-based learning and hidden vector state models two common machine learning paradigms (He & Young 2006). Alternatively, reinforcement learning, as demonstrated in the NJfun system, involves an exploration of the possible system actions when the dialogue is in a particular state, and decides on the best action which will optimise the system's performance with regard to a stated utility function (Singh et al. 2002).

The initiative of the dialogue describes who is leading the conversation, either the user or the system. If the user has the initiative, then he/she can speak freely to the system as it will be more open ended, allowing them to ask questions to the system in an unrestrictive manner. As a consequence however, difficulties can arise when the user asks a question that the system cannot understand, either because the words are out of its vocabulary or out of its particular domain. Furthermore, more complex language

understanding capabilities are required to recognise the concept and intent of the questions uttered by the user, including more advanced strategies for correcting errors in an open ended fashion.

In contrast to user initiative is system initiative, which offers a more inflexible and less open ended form of dialogue. The system simply asks questions, one at a time to the user, who then provides individual answers one at a time. Although at first it may appear to be a disadvantage to have the system simply ask questions to the user, with which they can respond with simple answers, this can be a very effective form of dialogue. The users generally have an idea of what they can say back to the system without it resulting in error, and the correct information is always requested from the user for the task that the system is performing. However, it does leave the dialogue 'closed', leaving the user directed by the system. It can also take longer to complete a dialogue, as the user can generally only provide one piece of information at a time for each system question.

Mixed initiative combines the advantages of both approaches, allowing both user and system to control the dialogue. Typically, such systems initiate the dialogue with user initiative, allowing the user to state their intentions in an open ended manner. If errors are encountered as the dialogue continues, or there is limited progress in completing the task, then the Dialogue Manager will switch to system initiative, and guide the user through the dialogue to completion.

Once the system has received information from the user, it must be confirmed due to the potential errors that occur in speech recognition. The speaker and hearer must constantly establish common ground, that is, the set of things that are mutually believed by both speakers (Jurafsky & Martin 2000; Clark 1996; Traum & Allen 1994). This

concept of grounding is very important in spoken dialogue systems for the completion of the task by the system, and the correctness with which it is completed. Indeed, it could be quite disastrous if the system has misunderstood the user, and booked the user on the wrong flight, for example. Generally there are two grounding techniques available - implicit and explicit verification.

Explicit verification allows for easier error recovery by asking the user explicitly if the utterance was recognised correctly, e.g., "Did you say Boston?", "Ok, you want to fly to New York, Correct?". Like system led initiative however, this will slow down the dialogue if each new value elicited by the user is confirmed. Implicit confirmation can be used in conjunction with the next question to confirm the new value e.g., "Ok, where would you like to fly to from Boston?" Although this would speed up the dialogue it also becomes difficult to correct, as the system would require understanding of a vast range of responses from the user e.g., "No, not Boston", "No, I meant Austin, not Boston", "I would like to fly to New York, but from Austin" etc. In order to cope with input such as this, the system requires a more sophisticated language understanding component.

Equally as important is the area of error handling - it is not always the case that what the user says will be correctly understood by the dialogue system, and it is important to handle errors effectively when they arise. Error handling in a dialogue system is somewhat more complex than in a Graphical User Interface, and can be classified as either misunderstanding or non-understanding (Bohus & Rudnicky 2005a). Misunderstanding errors refer to the system applying the incorrect meaning to a user's input, whereas in non-understanding the system fails altogether to detect any meaningful semantic interpretation of the user's input at all. The latter is often a

consequence of the user expecting the system to perform a task for which it has not been designed. For this reason it is important to devise better strategies for detecting problems in human-machine dialogues and deal with them gracefully (Carlson et al. 2005). Some strategies, derived from human-human interactions, have been used as the basis for error recovery, such as engaging the user in a task based sub dialogue to confirm the system's belief of the user's intentions, and thereafter treating the error as a mis-understanding (Skantze 2005).

Error prevention at each stage of dialogue design is paramount, and various dialogue prompts and strategies should be investigated by the developer to ensure a high number of errors are prevented at all times. Importantly, the initial prompt of a dialogue system can prove decisive between a usable dialogue system, and one which leads to many errors. Research from Raux et al. (2006) supports this by showing a reduction of non-understanding errors of up to a quarter on a dialogue system used over a year in the public domain.

2.4 Spoken Dialogue Systems: Advantages

Given the challenges and issues with spoken dialogue systems described so far, is there a need to replace the traditional graphical interface that has become so widely used? Graphical interfaces have become very usable and effective due to the many years of usability studies and evolutionary designs that have taken place since their inception. However they are not a natural form of interaction, unlike speech. Communicating with machines in natural, conversational language removes many of the interactive barriers that inexperienced users currently face. The dialogue manager will be able to assist with specific problems, with each party collaborating with one another to solve particular goals.

One of the main benefits of a dialogue interface is that of increased accessibility. Accessibility is concerned with how 'usable' or 'available' software is for users, particularly those with disabilities, and it is the focus of much effort from the W3 Consortium to ensure that the Internet and web pages are particularly accessible to all users, despite the disability one may have, be it sight, sound or physical impairment[1]. One major constraint of a GUI is the requirement of using devices such as keyboard and mouse to interact with the system, which presents obvious difficulties for those with disabilities. A VUI (Voice User Interface) removes this need for 'hands on' interaction with the computer, and will allow such users to interact freely with the system using speech. This will also be true for situations when the hands-on approach of graphical interfaces is unsuitable, for example, with 'in-car' systems.

Additional benefits include that a user is not required to have special hardware, such as a computer, to interact with a system, and they do not need to have knowledge of how to use the graphical interface in order to do so. The combined benefit of these two considerations is that the only requirement for interacting with a spoken dialogue system is a microphone and speaker. By replacing website-based front ends to many of their online services with a dialogue system, a user can now simply ring up and talk to the computer, performing a variety of tasks. This would be an attractive alternative for those users who have limited knowledge of technology and graphical interfaces, and who still wish to interact with computers. Commercially this too has a significant benefit in providing services to a new market of customers with minimum computing

[1] Web Accessibility Initiative (WAI) – see http://www.w3.org/WAI/

experience. Dialogue systems utilising a telephone can allow a company to offer their customers a means of interacting with their services at any time.

As discussed further in Section 2.6, spoken dialogue systems also offer an alternative means of interaction on mobile devices and small form computers which have graphical interaction constraints. Providing an opportunity to use speech for input over a less comfortable and usable method, the usability of mobile computer interfaces can be enhanced with a spoken dialogue interface, in addition to gains offered for the usability of traditional graphical interfaces when meeting the needs of different systems and people.

2.5 Spoken Dialogue Systems: Limitations

There are some drawbacks of dialogue systems when compared to both graphical interfaces and human-human interaction. Comparisons with graphical interfaces will be discussed in Section 2.8, however it is also necessary to mention the comparison of spoken dialogue interfaces with human-human interactions, as these are the benchmark that spoken dialogue systems are often designed to achieve.

Humans display a far greater level of intuitive thinking when engaging in conversation, being able to change dynamically their words, phrases and speaking style to adapt to all different kinds of conversational partner. For example, a person would adapt different speaking styles when speaking to a parent than a friend. This is the complete opposite approach to that taken by VoiceXML (see Section 2.9.3) and other implementations of current dialogue systems, where the system is static. Paths through the system are fixed, and the interaction is similar for one user to the next, so the interaction style is independent of the user.

This is also true of dialogue initiative and grounding techniques – all users of a dialogue system are often presented with the same initiative and grounding strategies throughout a dialogue system. What would be desirable is a system that can detect if the user is experienced or a novice, and automatically generate the dialogue based on system or user initiative. This adaptation to individual users would allow more advice and guidance to be given to less experienced users, whilst the more experienced users will be able to use user and mixed initiative to complete the desired task more quickly and efficiently. The same applies to grounding strategies. Implicit and explicit strategies have their advantages and disadvantages. By adapting to different users, the system could present the best strategy.

This lack of adaptation is due to the 'static' hand coding of dialogues by developers. Beveridge & Milward (2004) surmise that, although the static approach allows precise control of what can occur within a dialogue, it is an expensive process in terms of costs and development effort, especially for complex dialogues, where the number of possible paths through the dialogue can be in the hundreds of thousands. It also creates fixed dialogues that are presented to every user, containing no, or very limited, adaptive dialogue.

An approach to overcome this in known as the 'dynamic' dialogue system, discussed further in Section 3.1. Various programming techniques can be utilised to create dynamic dialogues and prompts during the course of the interaction. However this approach suffers from its own limitations: it requires a well-defined and structured representation of the domain knowledge; and they are created purposely for interactions within that specific domain, reducing their extension into other domains.

2.6 Spoken Dialogue Systems: Multimodal Dialogue Systems

Multimodal dialogue systems are dialogue systems that offer additional modes of input and output, such as video output, or handwriting recognition; this allows the user to interact in the most appropriate way best suited for their current environment, and also allows the presentation of output to be generated in the most appropriate manner (López-Cózar & Araki 2005). By increasing the number of modalities that a user can use to interact with a computer, each modality can complement the others to help deliver a higher degree of confidence for the input and remove a lot of recognition errors. As summarised by Ringland & Scahill (2003) users can select the most convenient mode to use for any given circumstance, errors produced in one mode can be corrected using another mode, and multimodality can help interaction with smaller, more mobile devices. Furthermore, the inclusion of numerous modalities is specifically beneficial for mobile devices as users can often find the environment changing around them, and it is usually the case that input on one modality in the current environment might not be the best form of input in the next environment.

Multimodality could include any medium from using graphics and text, to more advanced areas of gestures and emotion recognition and production. This can lead to extra issues, as inputs from more than one modality need to be fused together, so that they provide the system with a complete input from the user. For example, if the user is interacting using multiple modalities, and says “I wish to travel from here to there” whilst pointing from one place on a map to another, the system needs to fuse together the inputs for the origin and destination of the journey, using the combination of map gestures and words spoken. This also presents challenges, as not only does the system now need to recognise more than one input from the user in a correct manner, but also

provide resolutions to interactions that are contradictory – for example, "I wish to hear more about the current sports stories" while pointing at a link to provide more information concerning the ongoing rise in house prices.

The main advantage of multimodal systems is that they can overcome many of the limitations associated with dialogue systems, such as the limited ability of humans to process a large amount of information conveyed through voice. By using multimodal dialogue systems, users can interact with web sites and systems using voice, yet still benefit from the graphical representation of information to overcome the cognitive load associated with informational transfer through voice.

Secondly, multimodal systems are seen as advantageous from an error handling perspective. Sturm & Boves (2005) note three issues concerning errors in a spoken dialogue system to be error reduction, error detection, and error correction. They propose that it is in the area of error correction that multimodal systems can really benefit users, and that, based on a related study (Suhm et al. 2001) multimodal error correction was indeed faster than unimodal correction by re-speaking. A multimodal approach can offer problematic interactions an alternative means of correcting speech recognition errors by simply offering the user an alternative mode of input.

On the other hand small form computers and mobile devices, including PDAs, are restricted in their input capabilities due to their reduced size. By using speech, users can make use of speech inputs and not worry about using a reduced keyboard size or other forms of input.

Additionally, due to the limited processing power available to mobile devices, traditional speech recognition software usually cannot run efficiently. The challenge is therefore how to get dialogue technology to run effectively on mobile devices, and

interact with servers if used as part of a distributed system. One solution is to simply do limited processing on the input signal, and send the unparsed signal to the server via wireless communications, and use the superior processing power available to the server to parse the input string (Ayres & Nolan 2006).

One of the problematic errors when designing multimodal interfaces is that different mobile devices have different input and output capabilities. This is a challenge as the developer cannot specify an interface with any degree of certainty how it will be rendered on a certain device, and if the interaction methods will actually be available. There are standards available for multimodal development, such as X+V (see Section 2.9), or MIRS (Multimodal Interaction and Rendering System), which is a language from an active research project that aims to overcome these interaction difficulties (Mueller et al. 2004).

By utilising other mobile technology, such as wireless communications, dialogue systems on mobile devices are proving to be a very popular research thread. The benefits of adding a spoken dialogue to mobile computing is being demonstrated by many active research projects. Lopez-Cozar et al. (2005) have demonstrated a mobile system where pupils and teachers can request information regarding their studies using multimodal and wireless technology. This is a novel university system that uses wireless communication protocols to interact with a back-end database to build up the grammars for the speech recognition, and also provide information access to the pupils and teaching staff. Similar mobile spoken multimodal systems that allow the user to access information are presented in (Bronsted et al. 2005; Chen et al. 2005).

2.7 Spoken Dialogue Systems: Evaluation

A spoken dialogue system, like any new system, must be subject to a thorough process of testing and evaluation to ensure that all requirements are met, and to achieve a measure of how usable the system is by its targeted users. The results can allow developers to enhance the performance of a dialogue system, and identify those areas on which the effectiveness and satisfaction of the system depend. Quite often these areas are speech recognition or understanding related, illustrated by preliminary evaluations of the DI@L-log system which indicated problems with the open ended implementation of the prompts and grammars. A re-engineered implementation utilising a more focussed approach to dialogue increased the user satisfaction significantly (Black et al. 2005).

Evaluations of spoken dialogue systems are made somewhat more complicated due to the interacting nature of the many different components that make up typical spoken dialogue architectures. Dialogue developers are quite often only interested in the dialogue aspect of the interaction, and the performance of the dialogue manager in the overall architecture. However, since the performance of the dialogue manager depends crucially on the performance of other components, such as the speech recogniser and text-to-speech synthesiser, users may not be aware of the dialogue management aspect specifically, and base their judgements on the quality of those components that are the tangible inputs and outputs in the interaction. Important questions to ask during an evaluation are what you want to evaluate, how you are going to measure it, and the meaningfulness of the results.

A number of evaluation techniques and metrics have emerged, many of which are now standard when evaluating a spoken dialogue system. Two approaches are commonly

used - firstly the system and user behaviour are logged in order to quantify the performance of the systems and its components, and secondly, the entire system is evaluated from a user's point of view, through questionnaires and interviews (Möller 2005a). This combination of qualitative and quantitative metrics is used to undertake three different types of evaluation: performance evaluation measures the performance of the system using quantitative metrics; diagnostic evaluation detects and diagnoses design and implementation errors; and adequacy evaluation measures how well the system fits its purposes and meets users needs (Bernsen et al. 1998: 191). Paek extends the need of well defined evaluation standards so that: an accurate estimation can be made of how well a system meets the goals of the task; comparative judgements between systems can be made; factors or components in the system can be improved; and tradeoffs or correlations between factors can be identified (Paek 2001).

Typical quantitative metrics used, defined as interaction parameters by Möller (2005b), include the dialogue duration, delay length of user's response, number of turns, the length of the prompts, and word error rate. By recording and analysing many of these interaction parameters, developers can judge and assess how their dialogue system is performing, how the system can be optimised, and where re-engineering is needed. Kamm et al. (1999) however discuss certain issues concerning the reliance on interaction parameters as a measure of the quality of dialogue, as often these interaction parameters may contradict one another, leaving developers with the tricky task of untangling the interactions or correlations between parameters.

Furthermore, due to the interactive nature of dialogue, these interaction parameters do not always correspond to the most effective user experience (Lamel et al. 2000). They themselves do not directly measure and record the user's judgement of a system, and

importantly their satisfaction of the system. User satisfaction, ease of use, and quality of output are some of the subjective measures important to developers (Larson et al.. 2005). Currently the only method to achieve this is through subjective measures, most commonly through the use of questionnaires after an interaction with a dialogue system (Hartikainen et al. 2004). A widely accepted questionnaire to record a range of user's opinions on different aspects of a dialogue interaction is the so called 'SASSI' questionnaire (Subjective Assessment of Speech System Interfaces) (Hone & Graham 2000, 2001).

PARADISE (PARAdigm for Dialogue System Evaluation) is a generic evaluation framework that attempts to combine interaction parameters and subjective measures into a single performance evaluation (Walker et al. 1997). To create the interaction logs, dialogue corpora must be collected using controlled experiments, after which the user records their judgements using surveys (Walker et al. 1998). A set of 'cost and success' factors is then treated as a set of independent factors, and multiple regression is then applied which measures each factor's overall importance in the user's satisfaction. PARADISE then allows the comparison of different dialogue strategies by comparing weighted judgement scores achieved using the algorithm.

Recent research interest however has seen the prediction of the usability of a spoken dialogue system from the interaction parameters during testing (Möller et al. 2008). It is believed there is a correlation between certain parameters and the user's satisfaction of a system, for example, the lower the word error rate of the speech recogniser the more satisfied the user will be with the system. The rationale of prediction usability is to help designers in making choices between system versions and lower testing costs at early phases of development (Möller et al. 2006). Möller et al. shows that the reliability

of prediction models relies greatly on the reliability of the interaction parameter used as input to the models (Möller et al. 2007). Callejas & López-Cózar (2008) argue however that user satisfaction is dependent also upon the dialogue management strategy used, and not only the interaction parameters. This suggests that evaluation methods themselves need to be tailored specifically to the type of interactions being analysed.

2.8 Spoken Dialogue Systems: Usability Considerations

An important aspect of any system's development is that of usability. A user's satisfaction with a system often lies not in the technical achievements of its implementation, but on how well they can interact with the system. Usability can be defined in terms of three dimensions: the extent of its effectiveness in doing typical tasks, the efficiency with which the task can be done, and the satisfaction of the user when carrying out the task (ISO 9241-11, 1998). As spoken dialogue systems are a new and unfamiliar interface for many users, it is important for dialogue developers to understand usability, how users currently interact with graphical interfaces, and then consider how interacting with the same task will differ through a dialogue interface (Dybkjær 2005).

Since the evolution of the graphical browser, usability engineering has become an integral aspect of system development, with suitable standards devised after many years of usability evaluations. Nielsen (1993) defines 5 different considerations that must be evaluated to give a measure of the usability of a system. These are learnability, efficiency, memorability, errors and satisfaction, typically measured by having a number of test users interact with the system to perform a number of set tasks. Virzi (1992) and Nielsen & Landauer (1993) have both published influential articles on the

topic of sample sizes for usability testing; with qualitative results collected using data from questionnaires and interviews.

As a consequence of many years of usability studies the graphical interface has evolved from an unnatural paradigm for computer interaction into an effective and satisfactory means for communication with computers. Large bodies of text can be read quickly, suitable icons and graphics represent an appropriate action, the available functionality of a piece of software can be visually inspected by the user, which can be restricted to only the correct options in a given interaction state, and inputs from the user can be easily validated and verified, using constructs such as drop down boxes and JavaScript. However, the mere presence of learnability as a consideration of usability illustrates that there is a learning curve for users to make use of a graphical interface, and graphical interfaces on mobile devices have yet to reach the usability standards of home computers.

The usability considerations of spoken dialogue systems differ from those of graphical interfaces due to the interface not being evident or available for inspection to the user before interaction. Consequently the user may not be aware of the functionality of the dialogue system or how to interact with the system. This can cause further issues for the speech recognition engine, the constraints of which are not obvious to the user. Guidelines to prevent this have been proposed by Suhm (2000), most notably to use the system's prompts effectively to constrain the user's input to the system's speech recognition grammar, making sure that any keywords to be used are easily recognised and not confusable.

Furthermore, whereas graphical interfaces rely on control through screen, keyboard and mouse, spoken dialogue systems use speech which is a perceptually transient rather than

static interface (Dybkjaer & Bernsen 2001). This means that the user must pick up the outputted information by the system the moment it has been provided, or miss it altogether. Human factors and psychological understanding play an important role here ensuring that the user is not overloaded cognitively with information that cannot be perceived visually. Given that humans can remember only five to nine things for around twenty seconds in their working or short term memory, dialogue engineers must be sure to remain within these limits when designing verbal menus and prompts (Weinschenk & Barker 2000).

Although there is potential for a dialogue system to offer a more natural interface than a graphical interface, the many years of usability study and evolution that graphical interfaces have over their dialogue counterparts means that usability issues have been explored more widely and so the interfaces are more mature. For both task based and information based interactions, graphical interfaces offer a very high standard of satisfactory interface. Graphical browsers are very usable for reading large bodies of text, search results can be meaningfully displayed in various forms for user interpretation, data input is quite acceptable through the use of a keyboard, the detection and correction of errors is quite effective and the graphical interface performs independently of current surroundings. However, in dialogue systems in the domain of online browsing, large bodies of text take more time to speak out, search results are difficult to output because of the linear nature of speech, data input can prove troublesome due to speech recognition and language understanding errors, errors can be problematic to detect and correct, and lastly the user must interact in a quiet environment.

2.9 Enabling technologies

As stated previously, typical spoken dialogue architectures utilise various technologies from numerous areas of computing. Enabling technologies for a dialogue manager are themselves vast and varied, and one can further observe that there is a difference in how dialogue managers are implemented in industry for commercial use as opposed to the laboratory for academic research. Industry implementations are usually found to be more standards based, as defined by the W3C 'Voice Browser' group (Froumentin & Ashimura 2006). Formed in 1999 by the World Wide Web Consortium (W3C)[2] to promote the standardisation and specification of voice technologies, it aims to utilise web technologies to further dialogue systems. Entire spoken dialogue systems can be realised using the implementations developed by the Voice Browser group, which include the specifications such as

VoiceXML[3], Speech Recognition Grammar Specification[4] (SRGS) and Speech Synthesis Mark-up Language[5] (SSML), which form part of the W3C Speech Interface Framework.

However, it is usually a trend with research groups to develop and promote their own specifications. This can lead to a gap forming between the two parties as sometimes, but not always, the research is far removed from the standards, so not of appeal to industry. Sometime research groups themselves are unaware of what the actual requirements from industry are for commercial dialogue systems.

---

[2] http://www.w3.org
[3] http://www.w3.org/TR/voicexml21/
[4] http://www.w3.org/TR/speech-grammar/
[5] http://www.w3.org/TR/speech-synthesis/

Pieraccini & Huerta (2005) develops such thoughts highlighting that industry and research often have two, quite conflicting views on how spoken dialogue systems should interact with the user. Larson on the other hand, noted as being a great 'bridge' between these two divided camps, working foremost as part of a large industry corporation, but also being the co-chair of the W3C Voice Browser group, counters this argument by stating that research is the first step in the life cycle of a new technology, eventually leading to its standardisation (Larson 2005a). Larson does have a good point here, as most of the technologies used today often start out as research projects in the laboratory. Take the Hyper Text Transfer Protocol (HTTP), for example. However one must further question this life cycle, as not all research technologies can be adapted as standards, and if they were, then what would the point of standards be in the first place.

Nevertheless, standards have been specified and developed, making up a suite of applications known as the W3C Speech Interface Framework. The standards devised by the W3C Voice Browser group are all applications of the language known as XML. It is these standards that will be used for development of the proposed system. Before reviewing the technologies that enable dialogue however, one must have an appreciation of XML, what it is capable of. Other related technologies will then be briefly introduced.

2.9.1 Enabling Technologies: XML

XML was first introduced by W3C on the 10th February 1998 (Thompson & le Hegaret 2005). Introduced to complement HyperText Mark-up Language (HTML) as a means of specifying websites, XML is primarily concerned with the structure of data, as opposed to the layout and presentation of data. This was seen as a much needed

requirement for the Internet, as HTML was becoming very unorganised and informal, as manufacturers of different web browsers would implement their own different specifications of HTML, leading to incompatibilities between different browsers. HTML developers themselves were also often quite 'lazy' with HTML, as it did not require precise, or strict, formal coding for it to work.

The birth of the Web was very chaotic, and the non-standard modifications and inconsistencies of HTML reflected that chaos (Morrison 2002). XML was introduced therefore to help formalise HTML, and apply structure to data, something which HTML is not that particularly good at. HTML was more equipped to handle the presentation whereas XML is a simple means of describing the data or content. There are no presentation or layout concerns included with XML - it is described as a meta-language, a language that describes other languages. This is one of XML's advantages, there is no limit on what it can be used for, it is truly extensible. Another important goal of the XML language was for the need to create well structured and formalised XML documents for the browser to parse them correctly. This would be moving away from the forgiving days of HTML, and one of the original design goals of XML calls for this explicitly (Bray et al. 2006).

With XML being used to describe data structure, how can one use this technology to present this data? For this to be done, some form of parsing or processing must be done, to transform the XML into another presentation language, such as HTML. This is another great advantage of XML, that the data structure and presentation are separated, unlike its predecessor HTML. Because the structure and presentation are separate, the same structure can be transformed into more than one different presentation medium, simply by changing the template that is doing the transforming.

In conjunction with XML, XSLT (eXtensible Stylesheet Language Transformations) was developed to transform an XML document into another form. XSLT 1.0 appeared as a W3C recommendation first in November 1999 (Fitzgerald 2004). An XSLT document specifies a 'Stylesheet' that can transfer an XML document into another language, for example HTML, PDF, Database, or VoiceXML (see Section 2.9.3).

XSLT is a very powerful language that allows the developer to do lots of interesting things. An XSLT document could transform the content held in a XML structure into a HTML table, neatly showing the information contained with the XML document. However, if the same developer wishes to use the same content, but this time for display on a mobile, another Stylesheet can be created, this time displaying the essential data in a more efficient way for viewing on a smaller screen. It is clear to see the benefits of separating out the presentation knowledge from the content knowledge.

2.9.2 Enabling Technologies: XML Applications

One of the most popular and most used applications born from the introduction of XML meta-language is 'RSS'. RSS, or Really Simple Syndicate, was first introduced in March 1999 as RDF Site Summary (Resource Descriptive Framework), by Dan Libby of Netscape (Anonymous 2006). RSS is a pure text format, and doesn't contain information about how a document should be presented. It simply uses XML to semantically distinguish information, which can then be transformed into whatever way appropriate (Wittenbrink 2005).

RSS allows companies to produce 'feeds' information, which contain just the content. For example the BBC makes the current world news stories available as an RSS feed[6]. As it is just the content that is included with an RSS feed, users rely on news or feed readers to render the RSS feed, and display it in a meaningful form. This allows the user to get all the RSS feeds that they are interested in delivered into the same software package, so that they do not have to navigate around many sites on the World Wide Web. Likewise, it allows web developers not to worry if the user is viewing the content on a personal computer or a mobile phone, as it is the RSS feeder that will be displaying the contents on the screen in an appropriate way, specific for the device that is currently in use.

RSS feeds are just one successful web application that has been made possible due to the emergence and continued use of XML. Similar technologies are 'web services', which allow different applications to communicate with one another using the Internet. A service, such as SOAP, or REST, defines a standard specification, using XML, of requests and responses for data. So a developer, for example, could make a request for information from a weather provider, and include the information on his/her web site. This is similar to how APIs (Application Programming Interface) operate. APIs are created by web developers and companies to allow individuals to interact with their systems (Zirkle 2003) APIs are commonly available for many popular web sites, such as eBay[7], Wikipedia[8] and the BBC[9]. The user can package the information they request

---

[6] feed://newsrss.bbc.co.uk/rss/newsonline_uk_edition/front_page/rss.xml

[7] http://www.ebay.co.uk

[8] http://www.wikipedia.org

[9] http://www.bbc.co.uk

into an API, for example a book search on Amazon.co.uk[10]. The results then come back as an API response, usually in an XML based format. The user can then render these results and display them however they wish.

Mentioned here are just some of the web technologies available for developers to use to build a wide variety of sites, and have also been used to provide the foundation for many dialogue systems.

2.9.3 Enabling Technologies: Voice technologies

Whilst developing the main dialogue specification, VoiceXML (Voice eXtensible Markup Language), the main concern for the W3C Voice Browser group was providing interactions with existing web technologies, including the World Wide Web (Boyer et al. 2000). The Voice XML specification was essential to making Internet content and information accessible via voice and phone (Hocek & Cuddihy 2003; Miller 2002). VoiceXML applications are not seen as replacement interfaces to the Internet, but seek to offer additional access to the same content through a highly accessible medium - the telephone.

VoiceXML is an XML-based mark-up language for distributed voice application first published in 2000, much as HTML is a mark-up language for distributed visual applications (Sharma & Kunis 2002). The similarities with HTML are obvious - firstly, the workflow of VoiceXML interaction is similar as well to the HTML workflow. With VoiceXML, a voice browser is required to fetch the VoiceXML documents from the server, interpret them, verbalise the contents for the caller, and then accept voice input from the user (Larson 2003).

---

[10] http://www.amazon.co.uk

Secondly, VoiceXML allows dialogue designers to develop a frame based dialogue (see Section 2.3) by specifying a form representation of the dialogue, containing fields that must be filled with values elicited by the user. This is comparable to the HTML paradigm of forms and fields that the user can select values from or enter values into. This similar approach is appropriate for web developers who are familiar with HTML, but also on a conceptual level, is quite fitting for end users, as they may already have a visualisation of the web based form for that interaction based on prior experience. Whereas HTML forms contain fields, each of which usually has a question displayed graphically, VoiceXML forms contains fields with <prompts>, which specify the question to be asked to the user. Likewise, HTML has an associated area for the user to enter a value or select from a list of given values, VoiceXML fields have a <grammar> containing the allowable answers that the user can submit.

VoiceXML has some major drawbacks however. Similar to that of web programming, where the content itself is mixed with the syntax of HTML, VoiceXML also mixes both the dialogue and domain knowledge. This is a concern mainly for developers as code can become unmanageable and difficult to maintain, and with regard to adaptive dialogue systems, as dialogue cannot be adapted to users if the domain knowledge is contained within the dialogue. To overcome this, like XML and XSLT, it is common with researchers to separate the domain and dialogue knowledge from one another in the form of a dynamic dialogue system (see Section 3.1).

Additionally, although the commercial potential of VoiceXML has been demonstrated with a number of products deployed based on this standard, the specification has not been produced for research purposes, and lacks some of the more advanced features of dialogue that are appealing to academic researchers. Integration with natural language

understanding and generation components for more advanced language parsing and interaction, further complicated by incompatibilities of language understanding formats and VoiceXML, further adds to the limiting appeal of VoiceXML to academic researchers (Mittendorfer et al. 2001).

Furthermore, due to the large operating constraints of system resources and the number of components required to run a VoiceXML system, such as a web server, VoiceXML Interpreter, speech recognition and TTS engines, ECMAScript interpreter and associated technologies, VoiceXML platforms are normally implemented using a three tier client server architecture, and not installed on embedded devices, not making VoiceXML a desirable solution for such devices (Bühler & Hamerich 2005). However, due to the increasing use of small form computers, with a limited screen size compared to traditional computers, there is a new found motivation to incorporate VoiceXML technology pervasive devices such as PDAs, smart phones and tablet PCs (McTear 2004b).

To realise multimodal dialogues, an additional set of tags has been made available to VXML, and this is the basis of the X+V[11] specification. An acronym for XHTML+VoiceXML, X+V is a specification that allows the creation of a multimodal dialogue by incorporating a smaller set of VoiceXML tags into the XHTML specification. As forms are loaded into the browser, associated VoiceXML <prompts> will be played as focus progresses from input to input. A compatible browser is required, such as Opera[12], to render the X+V pages correctly.

---

[11] http://www.w3.org/TR/xhtml+voice/

[12] http://www.opera.com/

There is a competitor, however to the X+V's specification. SALT, or Speech Application Language Tags, extends existing Web mark-up languages such as XHTML and XML to also enable multimodal and telephony access to the Web. Introduced in 2001 and maintained by the SALT forum[13], the SALT 1.0 specification is also under consideration of the W3C to be made a development standard (Platt 2004).

Both specifications have their respective advantages and disadvantages – X+V utilises the accepted and well developed VoiceXML foundation, but multimodality is provided more as an add on solution rather than being core, whereas SALT has had multimodal considerations core to its development, but is not open source and does require an associated proprietary framework to execute.

2.10 Spoken Dialogue Systems: Advanced Architectures

The spoken dialogue architecture presented in Section 2.2 is typical of many commercial and academic systems, providing the necessary framework for spoken dialogue interaction between human and machine. Common tasks, such as: information retrieval regarding orders or account details; bookings of various items such as concert tickets or flight tickets; or transactions that allow customers to pay for bills and invoices, can be fully supported using this architecture. However, as dialogue research has progressed, more advanced and flexible architectures have been required to realise additional, more advanced functionality. For example, the TRIPS system incorporates components that use artificial intelligence technologies such as planning, allowing more advanced interactions where the dialogue agent collaboratively interacts with the user to solve a problem in the domain of disaster management (Allen et al. 2001).

---

[13] http://www.saltforum.org/

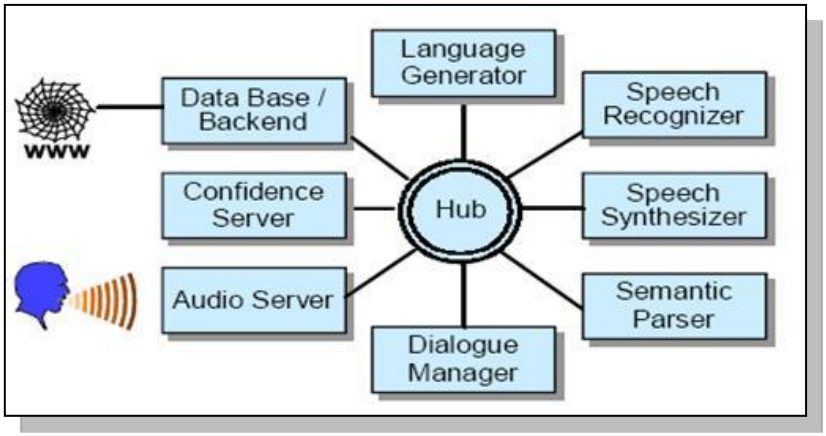


Figure 2.2: DARPA Architecture

(Pellom et al. 2001)

Numerous architectures were investigated during the literature exploration, and only a selected few will be presented here. Specifically, architectures and mechanisms were identified that handle multiple tasks or domains during an interaction. The Queen's Communicator (O' Neill et al. 2005), JASPIS (Turunen 2004) and CONVERSE (Batacharia et al. 1999) all use a form of polling agent or evaluator to weight different options available and select the most appropriate one for the current interaction, while RAVENCLAW (Raux et al. 2005) reinforces the method of separating the task knowledge and dialogue knowledge to promote extensibility and reuse with different task specifications and domains.

2.10.1 Advanced Architectures: Queen's Communicator

Resulting from research based at Queen's University, Belfast, the Queen's Communicator is an evolution of the DARPA Communicator. The goal of the original DARPA Communicator was to develop robust spoken dialogue systems that support complex, conversational interfaces. It had been developed to allow users to call into the system using a phone, and enquire about a number of travel options and itineraries, such as flights, hotels, and car rentals. The DARPA Communicator architecture used a 'hub-and-spoke' architecture, as shown in Figure 2.2, composed of a number of servers that

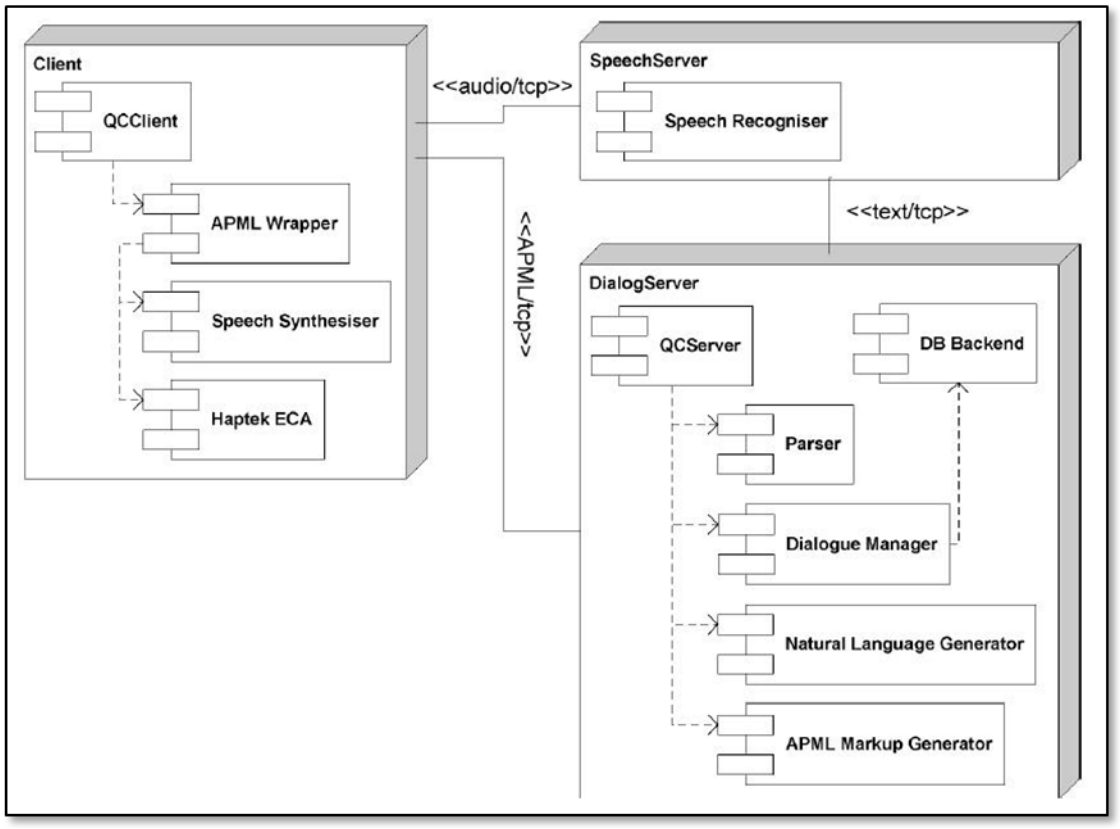


Figure 2.3: Queen's Communicator Architecture
(Hanna et al. 2007)

interact with each other through the DARPA Hub (Pellom et al. 2000). The Queen's Communicator (QC) is an evolution of this system, replacing the dialogue manager with a Java based Object Oriented (OO) version. Based on a client-server paradigm, the architecture is presented as Figure 2.3.

The attraction of object-orientation is that it can be used to separate generic dialogue behaviour from domain-specific behaviour (O' Neill et al. 2005). By implementing the dialogue manager (shown in Figure 2.4) in an OO fashion, the Queen's Communicator allows the advantages associated with OO programming to be brought to the domain of spoken dialogue systems, such as inheritance, extensibility and encapsulation. Implemented as a series of agents (specialised objects performing a particular task) organised as an object hierarchy, a key agent of the dialogue manager is the domain

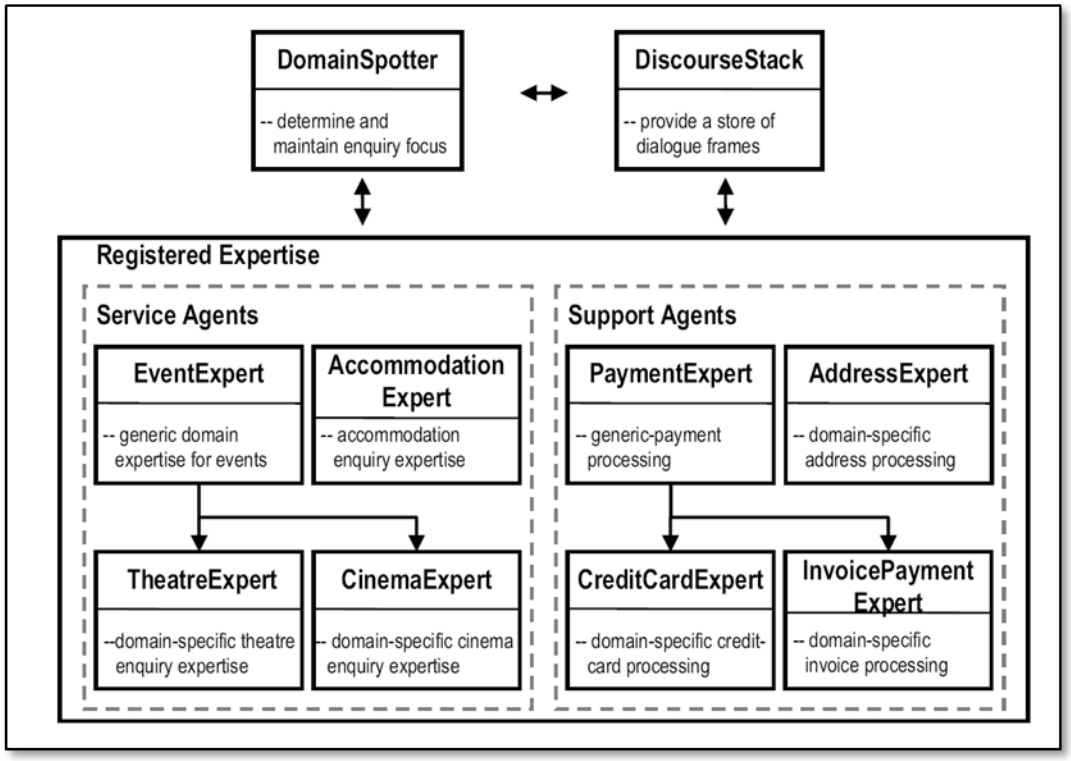


Figure 2.4: Queen's Communicator Dialogue Manager

(Hanna et al. 2005)

spotter (McTear et al. 2005). For example, the Event Expert contains general domain knowledge regarding event based dialogues, such as show date and number of adult tickets. However, experts lower in the hierarchy contain more specific rules associated with the domain represented, such as the Cinema Expert. Each domain that the QC can engage with in dialogue has its own expert and when the user engages the QC in dialogue, it is the task of the domain spotter to decide which of the multiple domain experts should be given control of the dialogue. As each agent is implemented as an agent with a specific task, encapsulation is promoted by confining all the methods and variables related to that agent within the object. This agent acts independently of other agents, and so they are unaffected by changes or alterations to its operation, and vice versa. This also promotes the extensibility of the system, as agents can simply be plugged in or removed from the system without affecting the operation of the other

agents. In real world terms, one could simply develop the rules required for the operation for a concert ticket domain expert, for example. This will then be available to the domain spotter during dialogues for it to allow requests in the domain of concert tickets.

Lastly, and arguably the most appealing for dialogue developers, are the benefits that inheritance has to offer. As the agents are developed as an object hierarchy, higher level dialogue management can be included at the upper most parent level. The domain experts themselves are at the lowest level of the hierarchy. This allows them to inherit all the dialogue management rules as the system is extended, so these same rules do not need to be programmed into each domain expert.

The dialogue flow can also be altered quite easily as the developer simply needs to change the appropriate rules in the high level dialogue manager, as these changes will then be inherited by all the child domain experts. There are further classifications within the hierarchy providing an even greater level of abstraction for the developer. For example, the cinema and theatre experts are both children of the event expert. This avoids duplication of rules amongst children objects and provides an easy mechanism to alter the dialogue should the need arise. The separation of dialogue knowledge from the domain knowledge removes the issues and problems associated with how VoiceXML mixes the two together (see Section 2.9.3). This allows the developer to control the domain and the dialogue separately, providing a more adaptable and dynamic approach to dialogue. This approach of domain spotting can also be found in work by Lee et al., in their Unified Multi-domain Dialogue Manager, however this architecture has yet to be evaluated in terms of extension and reuse (Lee et al. 2006).

2.10.2 Advanced Architectures: JASPIS Architecture

Similar to the QC, JASPIS is based on agents, each of which has a specific job to do (Turunen 2004). Shown in Figure 2.5, these agents are implemented in a modular and distributed system structure, an adaptive interaction coordination model and a shared system context (Turunen et al. 2004). Similar to the QC, evaluators play the part of the domain spotter concept, which selects the best agent most capable of performing the requested operation. The JASPIS architecture also includes managers, which are used to coordinate agents and evaluators.

Adaptive dialogues are possible by selecting the agent that is most appropriate for the current task. This is achievable as agents can be used to model different interaction strategies for the same task (Jokinen et al. 2002). This allows the evaluators to select the most appropriate initiative and grounding strategy, and also presentation mediums, based on its knowledge of the user and the dialogue so far.

Extension of the system is supported once again by the encapsulation of the agents, implemented as small software functions that perform a very specific task. JASPIS[1]

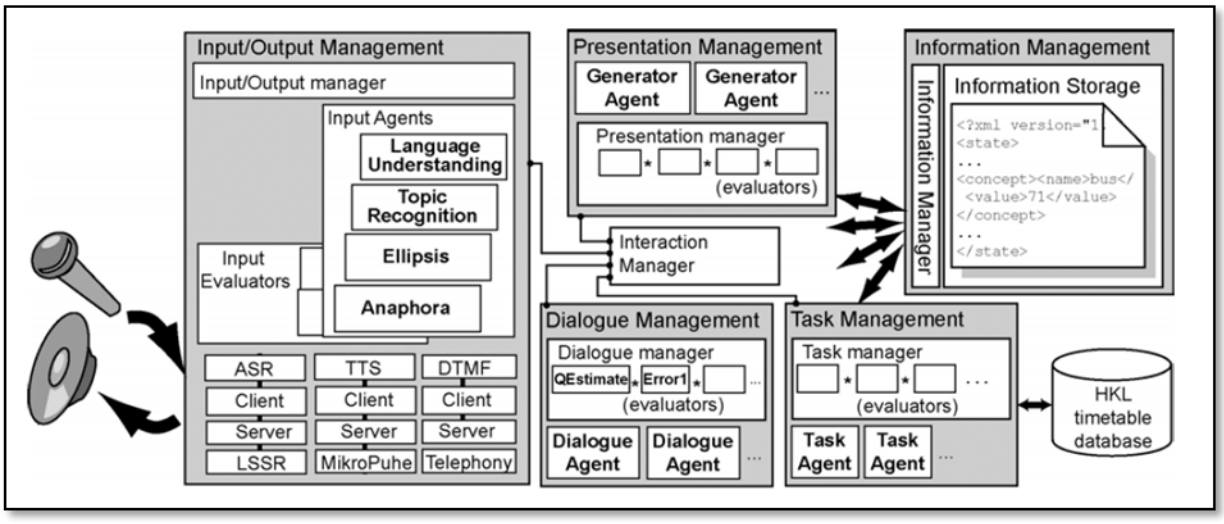


Figure 2.5: JASPIS Architecture

extends the original architecture by implementing a distributed dialogue architecture supporting the realisation of mobile spoken dialogue systems (Turunen et al. 2005).

2.10.3 Advanced Architectures: CONVERSE Architecture

Queen's Communicator and JASPIS have been shown to have two features in common: that they are both architectures created for task based applications; and that they both rely on a specialist component for weighting and choosing between more than one possible way to solve a task. In the case of the Queen's Communicator, a domain spotter is used to decide on the specific domain expert to solve the current interaction, and evaluators are used in JASPIS to decide on the specific agent to best handle the interaction.

Whereas the Queen's Communicator and JASPIS use a top-down approach to this decision making, CONVERSE provides an alternative, bottom up approach of using multi dialogue agents, known as Action Modules (Batacharia et al. 1999). The architecture of CONVERSE is shown in Figure 2.6. Comparable to the aforementioned domain spotter and evaluators is the Where-To-Go module of the CONVERSE architecture. However, in contrast to these modules, Where-To-Go does not apply weighting functions to its children modules to decide the most appropriate one. In CONVERSE it is the job of the Action Modules themselves to apply for their chance to handle the current interaction, similar to the analogy of an auction, where the Action Modules bid for their chance to handle the interaction based on the information they have of the situation, and the Where-To-Go module plays the part of the auctioneer, awarding the highest bidding module the opportunity to handle the interaction once all bidding has ceased.

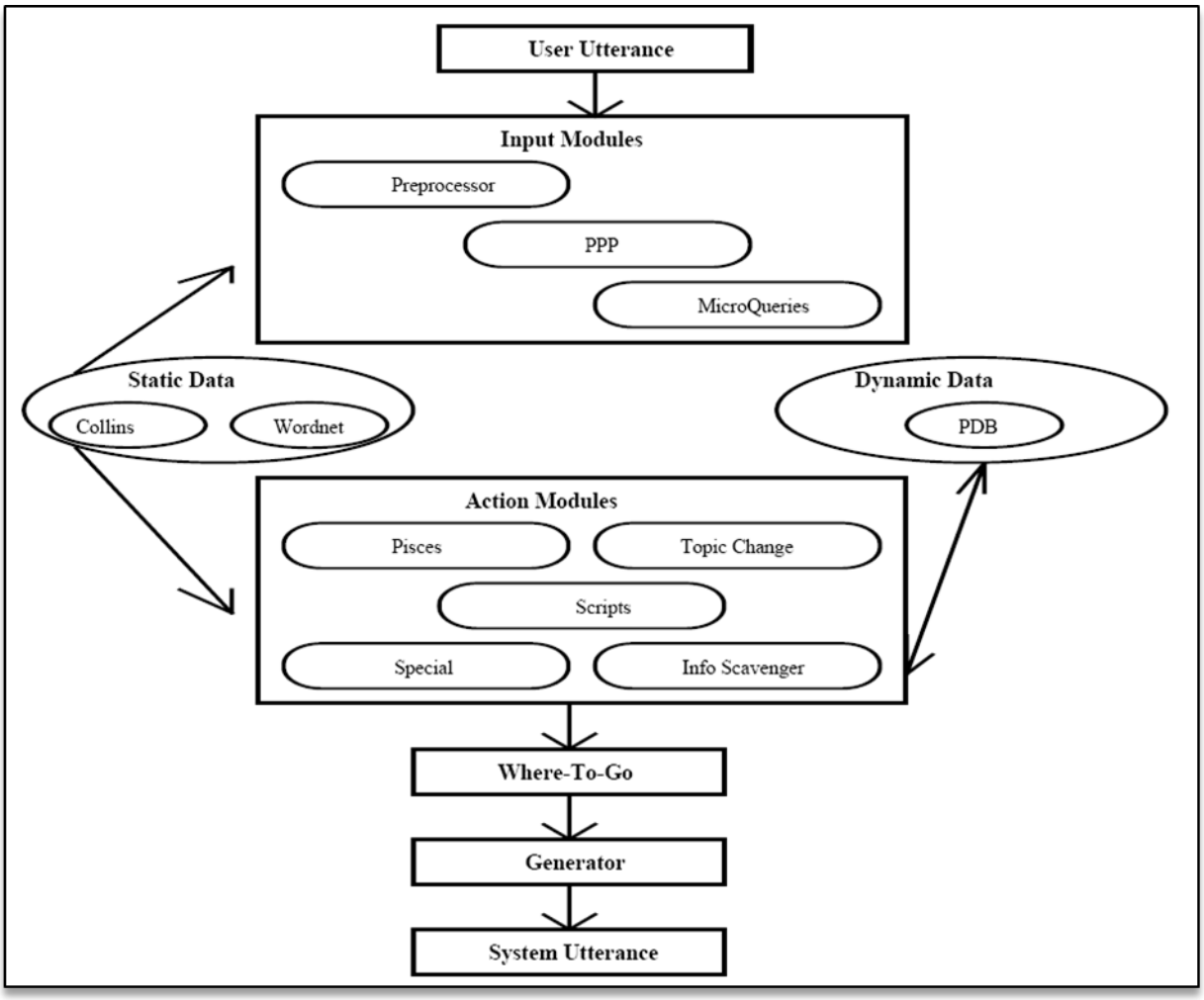


Figure 2.6: CONVERSE Architecture

2.10.4 Advanced Architectures: RAVENCLAW Architecture

The architectures discussed so far have been used to realise entire spoken dialogue systems. RavenClaw however is not an architecture for a system, but for a dialogue manager specifically. It is a task-independent dialogue engine that carries out a dialogue according to a given task specification (Raux et al. 2005). As introduced in the next section, there is a need to separate domain knowledge from dialogue knowledge, and RavenClaw includes a clear separation between task and discourse behaviour specification, allowing the rapid development of dialogue management components for complex, goal-oriented dialogue systems (Bohus & Rudnicky 2003). This is shown in Figure 2.7, where the domain knowledge is represented as a series of hierarchical

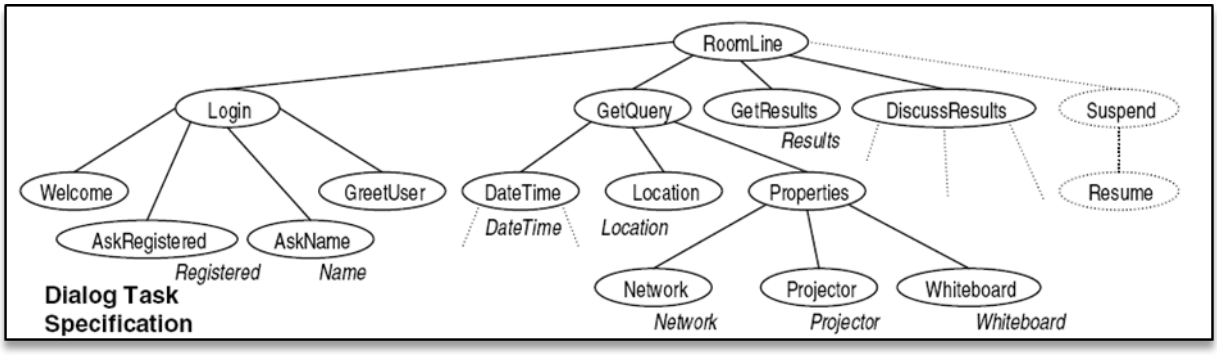


Figure 2.7: RavenClaw Dialogue Manager in RoomLine

objects, allowing reuse for different systems. By separating the task, or domain rules, from the dialogue knowledge, extensibility and reuse are encouraged by reusing the same dialogue knowledge in different domains, demonstrated by the use of RavenClaw in numerous dialogue systems, such as Let's Go! (Raux et al. 2003), ConQuest (Bohus et al. 2007), and RoomLine (Bohus & Rudnicky 2005b).

2.10.5 Advanced Architectures: Research Issues

Advanced features and aspects of spoken dialogue systems need to be realised by an appropriate system architecture. The aims, objectives and the specification of the dialogue system to be implemented should be clearly and fully stated, so that a relevant architecture that will accommodate all the required functionality can be conceptualised. This research aims to further explore dialogue interfaces for browsing online content, and the gaps and limitations of current research are translated into well defined system requirements and described in Chapter 4.

One requirement of VoiceBrowse is that it should be capable of interacting across multiple domains and content types. The Queen's Communicator and the JASPIS architecture both cater for this functionality. In the Queen's Communicator, the Domain Spotter accepts the user's query as its input, and matches this to a suitable Expert that can handle the interaction, for example, an Accommodation Expert if the

query concerned a hotel booking. In JASPIS, evaluators assess the current interaction and choose the most appropriate agent to continue, promoting adaptability within the multimodal dialogue architecture. Agents here are interactive elements that have a specific job to do, such as speech recognition, gesture capture, graphic generation or confirmation handling. This allows the switching of modes for input and output, and other dialogue features such as initiative, so that the system can adapt itself based on its current environment and user. The CONVERSE system has a similar mechanism, but is based upon an auctioneering approach, where each action module bids for their chance to handle the interaction based upon its belief of how well suited it is to do so.

In the case of all three mentioned architectures, the agents or classes handling the interactions specific to a domain or dialogue feature have been created or developed specifically for use in that architecture. This however would not be feasible for a generic dialogue system since each potential website would require its own agent or class to be created with its own knowledge. Moreover, the developer would need to anticipate which websites would be of interest to the user; and effort would be required to translate each website into the particular representation of knowledge used in the specialised agents or classes. Therefore an alternative, novel solution must be developed that will allow the inclusion of many different web sites, with many different content types and knowledge representations, that will dynamically change and evolve during runtime.

Architectures, however, are merely a framework that enables the system to produce the dialogue. It is clear that, for dialogue to occur opportunistically, and to be different with each interaction and for each user, as in the case of browsing the Web through voice, this has to be done dynamically.

2.11 Summary

Research in the area of spoken dialogue systems and, in particular, their potential 'real-world' applications, has attracted increased attention in recent years. This interest in 'real' systems and their growing use by the public has led to a new awareness of the problems associated with such systems, and a continuous striving towards realising true intelligent systems (Carlson et al. 2005). The inclusion of components such as dialogue management and language understanding set spoken dialogue systems apart from other types of speech system, many of which simply recognise the user's words without understanding.

Spoken dialogue systems can offer many advantages to both end-users and also industrial companies, but are still a relatively new technology in primitive form. The general public still have a high expectation of what a dialogue system is capable of, and do not appreciate the realistic capabilities of current dialogue systems. Usability is therefore paramount, and due consideration must be given to meeting the requirements and needs of different users that will interact with such a system. There are notable differences between the usability of a spoken dialogue system when compared to a Graphical Browser, and these must be taken into consideration when designing a spoken dialogue system. Available mechanisms and measurements for evaluating spoken dialogue systems have also been included, and will become of use after the implementation.

Current technologies have also been reviewed, followed by a discussion of more advanced architectures found in research. Chapter 3 reviews more advanced aspects of dialogue systems that are relevant to this dissertation and identifies shortcomings in current research that need to be addressed.

Chapter 3: Advanced Dialogue Research

The following chapter extends the foundations of dialogue systems presented previously by introducing more advanced features currently investigated in research. Specifically, the area of dynamic dialogue will be explored, including systems created using both structured and unstructured data. The categorisation of dynamic dialogue systems is done so in this way as the data VoiceBrowse will be required to access can be distinguished by its structure. Brief discussions on adaptive dialogue and information retrieval follow, as techniques and insights in both these areas have been utilised during the research, although in a less significant way. Research gaps and ongoing issues will be presented throughout and summarised at the conclusion of the chapter, including shortcomings which will be used to identify the contribution and functionality of VoiceBrowse.

3.1 Spoken Dialogue Systems: Dynamic Dialogue Systems

The ‘dynamic’ creation of spoken dialogue systems refers to the automatic creation of dialogue as opposed to being hand crafted by the developer. Two distinctions can be made between dialogue systems: where the entire dialogue system has been created dynamically; or where the system is created to be dynamic in nature, such as prompts and grammars created ‘on-the-fly’ as the dialogue evolves. Each type presents its own benefits.

First, any system that can be produced automatically with minimal development effort is attractive and desirable in terms of effort and resources required. Commercially, the development of any new IT solution is usually driven by development costs; an organisation will usually want the best program developed to meet their own

requirements, and at the lowest cost available. To make spoken dialogue systems attractive therefore, the development costs and effort must be minimal, it must be effective for the customer, and integrate seamlessly into existing computer systems. It is for this reason that dynamically created dialogue systems would have real commercial value and prove very advantageous for companies. Companies will already have their own IT infrastructure, both internally, and externally facing the customer side, such as the e-commerce website allowing the user to books flights or hotels online. One can deduce therefore that the infrastructure will include systems such as networks, servers, and well formed databases. To reduce development costs, and avoid re-inventing the wheel, it would be of benefit if a spoken dialogue system could be developed using this existing infra-structure. This corresponds to the first type of dynamic dialogue system, where the system in its entirety is created automatically from existing structures.

Secondly, if dialogue can be produced dynamically during runtime, then this removes problems associated with static dialogues, as discussed in Section 2.5. The contents of prompts can be varied and specific for each user, real time data structures, such as live commercial databases, can be used to provide information to users, and the dialogue does not have to follow a set path, but can evolve uniquely for each user. This corresponds to the second type of dynamic dialogue, where the system has been created to evolve dynamically during runtime.

In static dialogue systems, the domain knowledge is incorporated into the dialogue specification. Separating the domain from the dialogue knowledge however allows the dialogue manager to select the appropriate knowledge required for the dialogue, and to dynamically create the dialogue, either as one whole complete dialogue, or as it progresses.

This knowledge can be represented as a structured entity, such as a database or a XML representation, or alternatively, it can be in an unstructured form, such as simple text or content held online. Much of the work in this area of dynamic dialogues has focused on creating a good representation of the domain knowledge that is both accessible and meaningful to the dialogue manager. If this is created specifically for a dialogue manager, it removes complexities associated with understanding and retrieving information from knowledge sources not created for use with a dialogue manager, as in the case of relying on unstructured online content. Related research relying upon well structured domain knowledge to realise dynamic dialogue, such as the GEMINI and AMITíES systems or other work based on XML structures or ontologies, is presented below in Section 3.1.1.

However, a well defined and specifically structured domain representation is not always possible, as in the case of online content that is stored across multiple web sites. Different web sites will have different structures to one another, and how a dialogue manager extracts information from one will be different to the method used for extracting information from another. In this situation, it is commonly the case that specific web sites are used so that the site structure can be made available to the dialogue manager during development. A discussion of such systems is presented below in Section 3.1.2

3.1.1 Dynamic Dialogue Systems: Utilising Structured Content

GEMINI (Generic Environment for Multilingual Interactive Natural Interface), an EC funded research project, is a dynamic dialogue system of the former type which creates an entire dialogue system from existing infrastructure. Its foremost goal is to produce multimodal and multilingual dialogue interfaces to databases with a minimum amount

of human effort (Hamerich et al. 2003). Given a database structure, access to it, and a description of the requests that are allowable from the user, GEMINI should be able to automatically generate the dialogue scripts to run the service (Hamerich et al. 2004b). A novel feature of GEMINI is the language that was developed to allow this semi-automatic creation of dialogue systems, GDialogXML (Schubert & Hamerich 2005). GDialogXML specifies a dialogue using an XML based language, which is both modality and language independent, but allows the representation of the entire dialogue, from dialogue flow, to data and back-end modelling, and even user modelling. This specification is then turned into VoiceXML and XHTML realising multimodal dialogues. However, modalities are not currently fused together, and have to be run separately, so systems produced by GEMINI are not multimodal in the strictest sense (D'Haro et al. 2006). The system can reuse generic dialogue components in future systems to speed up the development time even more (Hamerich et al. 2004a).

AMITíES (Hardy et al. 2006) is another system that produces just the dialogue based on data driven techniques and existing data structures. Similar to the Queen's Communicator, it is based upon the DARPA Communicator architecture (see Section 2.10) and the dialogue manager and language understanding is automatically created, whether for a financial system or a travel system. To provide structure to the system, a corpus of annotated data at both the functional layer and semantic layer is required[14]. However, a common problem of data driven techniques includes the necessity of a large annotated corpus for modelling the interaction and testing. DialogStudio is a framework proposal for building data driven dialogue systems, attempting to overcome

[14] Refer to (Hardy et al. 2003) for information regarding the annotating of functional and semantic layers using DAMSL and XDML

this issue. It has been evaluated with a certain degree of success in three different domains (electronic program guide, immigrant simulation domain and weather information domain), but its generic effectiveness remains to be tested as a framework for building dialogue systems in other domains (Jung et al. 2007).

These systems have been of the former type of dynamic dialogue system, where the entire dialogue system has been created from structured content. Other research attempts to realise dynamic dialogue of the form that evolves 'on-the-fly' as the dialogue progresses, using structures such as databases and ontologies (Beveridge & Milward 2003; Montoro et al. 2004). By utilising an ontology to represent devices in a smart domain, Montoro et al. have produced a system that can automatically create the dialogue to interact with such devices through voice. By automatically creating the dialogue based on the XML specification of the environment, new devices can be added and removed, and the dialogue updated automatically. Milward & Beveridge (2003) have used an ontology to a similar effect, this time in the domain of medical knowledge. Here, a medical ontology has been produced in the area of breast cancer, allowing the users to question the domain knowledge about this area. Using the ontology, the system can relate terms and concepts - for example if the user responds "There is some distortion", the system can relate the term 'distortion' to various other concepts in the ontology. Here the concept was human skin change, so the ontology can be searched to find a concept which is related to skin change and for which 'distortion' is a term.

By using a structured domain source, dialogue developers can utilise scripts to access the content using associated query languages in a similar manner to web technologies and scripting. For most task based dialogues following a finite state or frame based dialogue, the required inputs to a query on the domain knowledge are mapped onto

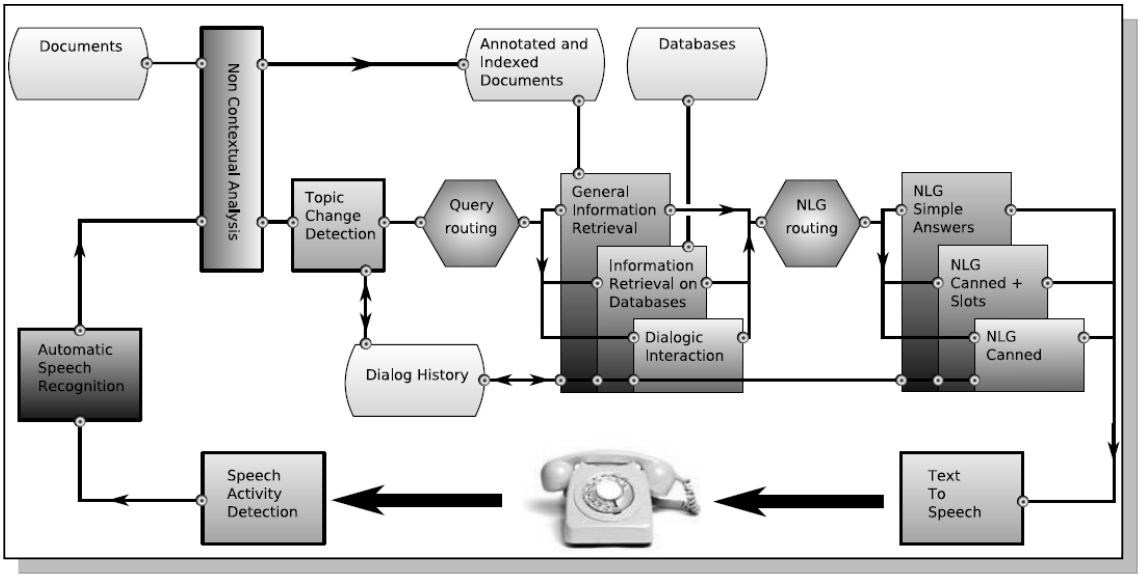


Figure 3.1: Ritel Architecture

distinctive stages of the dialogue. Dynamic dialogues therefore become more complicated if either of these two elements do not confirm to this approach, where either the domain knowledge is not of a well defined structure, and therefore cannot be queried easily, or the dialogue is of a type which does not follow a set path to complete a distinct task, such as a conversational, narrative form of dialogue. For the latter complication, Ritel (Figure 3.1) integrates spoken language dialogue technology with open-domain informational retrieval to allow a dialogue interface to a question answering system (Rosset et al. 2006). Based on information retrieval techniques, the dialogue interfaces with a specific collection of documents that are pre-processed and prepared in an appropriate manner for question answering. With the addition of more sophisticated language understanding techniques, dialogues that do not follow a set path or pattern become both manageable and feasible. However, dynamic dialogues systems relying on an unstructured knowledge base present more complicated problems.

3.1.2 Dynamic Dialogue Systems: Utilising Unstructured Online Content

One application area that promises great potential for a spoken dialogue system is interfacing with the World Wide Web, recognised by over 14 years of research in making the World Wide Web audible, Raman (1998) is a publication of research completed previously in 1994, not long after the birth of the World Wide Web itself. Aural Cascaded Style Sheets (Raman 1997b) aimed to work alongside Cascading Style Sheets (CSS), used by HTML developers, to output web page text in audio form. Although not dialogue in the true sense, it is the first documented case of providing an alternative interface to the graphical interface through speech (Raman 1997a).

Generally spoken dialogue systems are created for a specific task or domain. Even when designed to utilize online content to generate dialogue dynamically, only a small number of web sites are made available to the dialogue manager, along with specific knowledge of the site's structure. By contrast, graphical web browsers allow a user to accomplish a number of different tasks and to access different contents in a more open-ended manner. Furthermore, compared with GUIs, spoken dialogue systems are still a primitive interface in terms of usability as they do not allow users to interact with independent, unstructured domain knowledge and content in a generic and usable way.

More recent research attempting to realise a dialogue interface to the Internet has been consistent with the ethos of separating the domain knowledge from the dialogue knowledge, and treating the online content as the domain knowledge to the dialogue manager. Although the ever increasing popularity of the Internet has been partly due the syntactical standards made available for developers, there is no structural standard for the representation of web sites, and the technologies utilised contain no semantic information of the information represented. Additional issues arise when one considers

that there can be no anticipation of the requests a user may make, due to the vastness of the domain source, in contrast with a spoken dialogue system created for a sole task, in addition to the further requirement for informational retrieval techniques to extract the relevant content from the Internet.

The GENESIS system addresses the issues associated with preparing the content to be delivered through spoken dialogue (Polifroni et al. 2003). GENESIS is concerned primarily with the retrieval and preparation of task oriented content from the Internet for use in a spoken dialogue system. Although long term goals are for the system to be fully automated, the user's dialogue is simulated and the domain is limited to two types of requests, namely restaurants in Boston and hotels. The focus of the research is the preparation of the online content, such as how to cluster the domain data appropriately, and how to construct relevant summaries to represent the data to the user in dialogue (Polifroni & Walker 2006). For example, the restaurants could be classified as distances from landmarks in Boston or by cuisine. Further grouping data can take place once additional constraints have been provided. Gruenstein et al. present a similar system with focus on creating a structured database for dialogue, but utilising numerous unstructured online sources for task based dialogues (Gruenstein et al. 2006).

The Internet of course is a plethora of knowledge, and there is much more information available online concerned not only with specific task domains such as flight, hotels, or car bookings, but also with much more general, informative domains, such as news stories or sports results. Comparisons between the two domains can be made, however they need to be treated separately, as there are a different set of issues and challenges associated with each domain type. Similar to GENESIS, there is a small research push to deliver this content through spoken dialogue technology.

To automatically create dialogue, Gonzales-Ferreras & Cardenoso-Payo (2005) propose to first parse the data and apply a form of tree structuring. This concept is demonstrated in their spoken dialogue system for a newspaper front end. This approach, however, limits the dialogue to specific domains, as it is not possible to apply a tree structure to every form of unstructured data, and even if it were possible, it could not be done in real-time during a live dialogue with a user.

Pargellis et al. present a system that matches the user's interests, stated explicitly by selecting interest areas from a web page (Pargellis et al. 1999, 2004). As the research covers a wide range of issues and challenges from numerous research areas, the work and effort presented by Pargellis et al. is focused mainly on what constitutes a related story when compared to the user's profile. Other areas of the system are therefore more primitive, such as the content coming from one web site[15] and users explicitly stating their own profile.

Research presented until this point has limited the online domains available for dialogue to a single domain, or a small subset of domains, due to the non-standard, unstructured nature of online content. An alternative to this is the Semantic Web (Fensel 2003), a collection of online web pages that have been semantically annotated. The benefits of semantic information encoded into plain text documents that provide relevant meanings for machines are numerous for dialogue systems, and Reithinger et al. have demonstrated this with their own multimodal dialogue client for the Semantic Web, known as SmartWeb (Reithinger et al. 2007). The current evolution however from the World Wide Web to the Semantic Web has occurred on a relatively low scale, and the

[15] Content available for request is delivered from http://www.ESPN.com only.

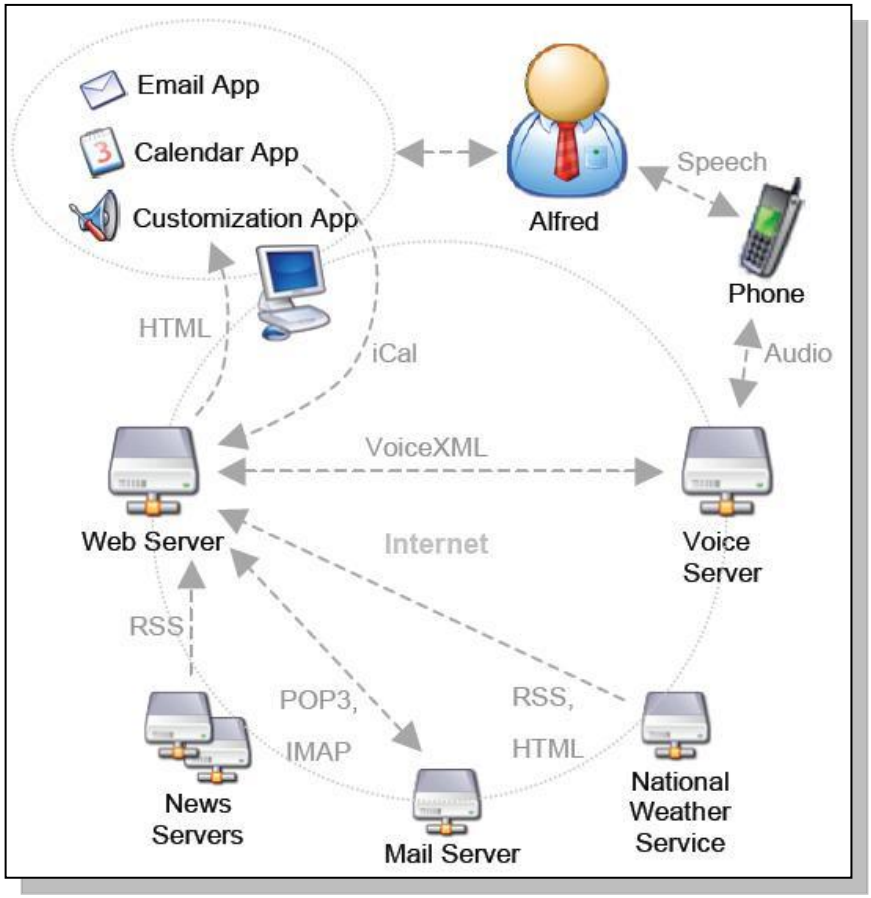


Figure 3.2: Customisable Phone Access to Personal Information (Perez-Quiñones and Rode 2004).

extra human effort required to produce the same information understandable by machines has limited SmartWeb and other Semantic Web based solutions to a small subset of domains and functionalities.

Perez-Quiñones and Rode provide a primitive solution involving the use of RSS to achieve multi-domain dialogues (Perez-Quiñones & Rode 2004). By using the XML based standard RSS, PHP scripts are utilised to extract descriptions of related stories from various feeds, illustrated in Figure 3.2. This dynamic, informative content is complemented by personal information held in a calendar file. However, the range of available RSS feeds are limited to the sole topic area of news headlines, there is no interactive dialogue between machine and human, and the functionality available to

users is somewhat limited, restricted to a low number of key tasks associated with managing the personal information.

The RSS content is simply used for outputs to the user with no allowable responses from the user. Furthermore, the presentation of the content has not been considered, being presented to the user in a simple unordered linear list of content items.

WebContext is a similar system, also developed at Virginia Tech (Capra 2003), with the focus here on using voice on a mobile device to re-find information that has been archived on a main computer. The rationale here is that it is likely a user will browse the Internet using their home or office computer, followed by the need to query this content once they have left the computer, to confirm a telephone number or obtain directions to a certain place, for example. This is achieved by use of a standard context representation for the online content, which can be archived and queried. This work is currently done by hand, so that it does not allow automatic dialogue with online content in the true sense. Nevertheless, the research has provided productive results with regard to user behaviour when searching and re-finding information through voice: importantly it was found that users mainly refer to web pages by title and descriptions, not by URLs; and that users do not fully express their query initially, but engage in a collaborative dialogue to find specific information, providing more details as the dialogue progresses (Capra & Perez-Quiñones 2005).

The dialogue implementation is of a scripted and inflexible nature using a Context Free Grammar (CFG) due to the 'Grammar Inclusion Problem', where a grammar becomes too large if all the words on every archived web page were to be included in the language model (Capra et al. 2001). This is similar to other dynamic systems, where

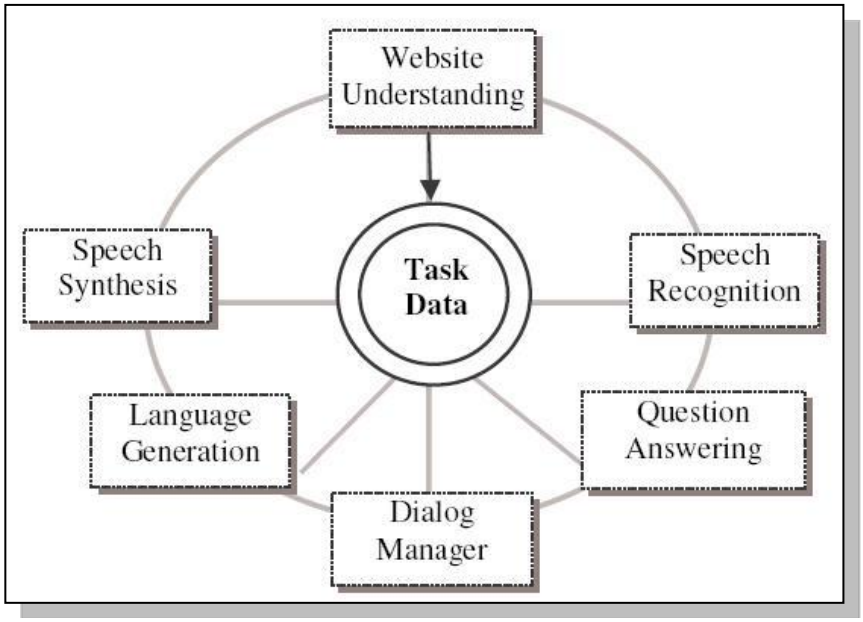


Figure 3.3: WebTalk Architecture

(Feng et al. 2003)

the speech recognition performance and word error rate can deteriorate due to the access and inclusion of a large amount of data from a database (Callejas & López-Cózar 2005).

The systems presented thus far have been dynamic systems with regard to the dialogue evolving as it progresses. WebTalk is an automatic commercial front end question answering solution for building the dialogue system itself from existing online technologies (Feng et al. 2005). The architecture, presented as Figure 3.3, includes an additional component to typical dialogue components known as the website understanding component. The goal is to mine a web site, and instantly create an interactive dialogue system that can answer questions and perform transactional requests (Feng et al. 2006). This is beneficial commercially for companies as they can replace human help desk operators with a dialogue system, removing much of the effort required for building dialogue systems by reusing existing technologies. However the system is specific to a web site, its associated structure technologies, and handles a

specific type of query. It is not faced therefore with the complexities associated with utilising unstructured online content for dialogue from various sources.

3.1.3 Dynamic Dialogue Systems: Research Issues

To create a dynamic dialogue system, it is an accepted prerequisite that the domain knowledge be separate from and accessible to the dialogue manager. For the domain knowledge to be usable, the dialogue manager must: have knowledge of it; have access to it; and methods for querying and extracting information. However, no piece of research is conclusive in this area, offering no concrete evidence to suggest the best way to accomplish these tasks.

GEMINI for example, despite having great potential for developing dialogue systems, still requires development effort, and is semi-automatic. True multimodality is not included, but simulated, with each modality having to be implemented separately. AMITIíES creates dialogue from structured data, however if no structured data is present then the dynamic dialogue cannot be created.

GENESIS accesses online domain knowledge for task oriented dialogues aimed at providing assistance to the user, similar to a telephone help desk agent, helping them to solve a particular problem or get specific advice. This can be particularly challenging due to the preparation needs of the online and unstructured content. Polifroni et al. (2003) primarily focus on domain data that can be structured in an efficient and effective manner, such as by street name or cuisine type, continuously narrowing down the subset of data that the user is interested in as the dialogue progresses. The same could not be said if the domain knowledge was more general – a dataset that could not have similar structural classification algorithms applied to it, such as news or other informative content types. A different method of classification is needed, such as

dining, lodging, car rental etc, so that the domains can be separated by the types of information they represent. Searches can be refined further by the user, just as in task oriented dialogues. "Give me news stories for Northern Ireland", for example. A major difference between informational and task based dialogues is that there is no set path for the dialogue to follow, but rather it is dictated by the user when engaged in dialogue.

With a task driven interaction, the user will usually have a specific question in mind when they are engaged in dialogue to derive an answer from the conversational partner. However, if there isn't a specific task or question to be tackled, then the dialogue itself will usually be directed as new information is presented to the user. This is in contrast to task based dialogues as the dialogue develops more opportunistically, whereas when there is a specific task to be addressed a pre-defined dialogue structure exists.

This is the first issue associated with browsing the Internet using a dialogue interface. With no set task to be completed, the dialogue itself is of an open ended form, reacting to information requested by the user. However, there is a second, more challenging issue. Traditional dynamic systems have been created with one, or a very small, number of well defined tasks and goals catered for. To achieve this, they are provided with a well structured and defined representation of domain knowledge, and associated methods and mechanisms for retrieving data from it. However, online sources do not follow the same knowledge representation. Web sites contain numerous different content types and domains, and none are created specifically for use through a dialogue interface. Previously, research has limited the dialogue to specific web sites, of which the structure and knowledge is made available to the dialogue manager.

To combat this, Perez-Quiñones & Rode have utilised XML technology by using RSS feeds to provide the dialogue manager with informational stories from different sources.

However, their system has only been demonstrated in the topic area of news, not in a multi-domain or multi-content setup, and task oriented dialogues, such as those catered for by Polifroni et al. have not been included. Furthermore, their system contains no dialogues based on the informational content delivered to the system, limiting the user to interaction with the content using a primitive dialogue interface. This effect became known as the grammar inclusion problem; where it became infeasible to include every word from the online content sources available to WebContext.

In short, the limitations of current dynamic dialogue systems can be summarised as follows:

- The domain knowledge is well structured and clearly defined.
- Dialogue management is created specifically for each domain, and is not easily transferable to another domain type or structure.
- Dynamic dialogues usually have set paths of interaction to accomplish a fixed task or goal.
- Accessing informative content generically is beyond the scope of current dynamic dialogue systems.
- Dynamic dialogues accessing online content are currently limited to a single or small number of websites.

Further challenges can be identified when one considers the need of the dialogue manager to react to the user's input, often relating to the previous result of the dialogue manager. A user may wish to hear more about the said news story, or maybe to hear about a different story unrelated to the current interaction. It is this engagement of dialogue between computers and humans in general conversation that presents

numerous challenges for researchers to consider. Dynamic dialogues do allow a certain degree of adaptation to particular users, as introduced in the following section.

3.2 Adaptive Spoken Dialogue Systems

People constantly change and adapt their dialogue to match the conversational partner - VoiceXML specifications of dialogue are static however, and do not allow dialogue systems to automatically adapt for different dialogue partners. By utilising the dynamic production of dialogue and prompts, this adaptation to different speakers can be realised, as the prompts and dialogue flow can be automatically created at run-time, tailored for that particular user. More or less help and guidance can be given or other considerations taken into account, whilst the system creates the prompts.

It is important at this point to briefly distinguish between adaptive and adaptable dialogues. An adaptable dialogue simply lets the user decide and specify certain parameters of the dialogue. Adaptive dialogue, however, is an active feature that changes the dialogue based on parameters detected from the user, such as high error rates. The emphasis here is on the dialogue system detecting the user's requirements and adapting itself, with this adaptation occurring unnoticed by the user and not interfering with the dialogue task.

Adaptive techniques can range from the very simple to the very complex. In task oriented dialogues, dialogue performance has been shown to be a useful indication if the user is experiencing problems in the dialogue. Techniques based on this approach will be introduced in Section 3.2.1. However, as developers face the formidable task of writing software for millions of users while making it work as if it were designed for each individual user only, user modelling and other techniques can enhance dialogue systems even more by adapting the actual content of the dialogue (Fischer 2001). User

modelling and other techniques for adapting the content of prompts will be discussed in Sections 3.2.2 and 3.2.3.

3.2.1 Adaptive Dialogue Systems: Adapting Dialogue

Errors that occur during dialogue can be used as an indication to the system that the dialogue management needs to be adapted, either to offer more assistance, or to change the way in which inputs and outputs are handled. The research challenge is for the system to discover the root cause of the errors, and then adapt its dialogue in such a way as to remove the problematic areas.

The main cause of understanding errors in dialogue systems is that the user has a greater expectation of what the system is capable of, and may not be aware of certain system limitations, and therefore might try to formulate a goal which the system cannot handle (Bohus & Rudnicky 2005c). This false belief of the user is further encouraged, especially with new users, when the dialogue gives the initiative to the user. An open ended question, such as "How may I help you?" may result in the user thinking they can ask about any topic or query, even those unrelated to the domain of the dialogue system.

To prevent this from occurring, the system can choose to switch to system or mixed initiative, to provide more help and guidance. Although this may be of help to the more inexperienced users who require the extra help, this will be of less assistance to the more experienced users who would be able to complete the dialogue more quickly if they had the initiative of the conversation. It would be helpful therefore if the system could detect if errors were ongoing throughout the dialogue, and could therefore adapt its dialogue strategy and initiative to match this perceived expertise level of the user.

Based on this hypothesis, researchers at BMW have developed an adaptive spoken dialogue system for the company's iDrive system (Haller 2003). The system first

classifies the users as either novice or expert; dependent upon different features of the dialogue, such as if the user is asking for help, the elapsed time since the last interaction, and confidence measures (Hassel & Hagen 2005). Once classified, the user will receive the relevant system prompts for their expertise level - novice users will hear the available commands that can be uttered, while experts will receive more condensed prompts. This approach has also been used in other adaptive dialogue research - for example a system for the health care domain (Giorgino et al. 2005).

It is common that certain interaction parameters are used to adapt the dialogue specifically for a user. MIMIC (Mixed Initiative Movie Information Consultant) is a more advanced dialogue system that also adapts to different levels of the user's ability (Chu-Carroll 2000). This system is interesting because it not only considers the initiative of the dialogue, but also the dialogue strategy which, as defined by the MIMIC system, is a set of dialogue acts that MIMIC can choose to use to provide further assistance to the user. For example, even though the system may choose to take the initiative from the user, it may just ask the question "What theatre?", as in normal system led initiatives. A user, however, may still experience difficulties answering this. The system then adopts a different strategy, providing more help to the user, "What theatre? Please choose between Wellmont or Clearview" (Chu-Carroll 2000). The initiative and strategy are implemented independently of one another, allowing for a finer degree of adaptation, unlike Hassel and Hagen's system which is crude in its adaptation for two very different groups of users.

A similar method is presented by Veldhuijzen van Zanten (1998, 1999) who has proposed a hierarchical slot structure as opposed to the typical flat slot structure in use by most dialogue managers. This is a novel feature for a dialogue manager that allows

the system to control the initiative on a much finer scale than before. It can ask very general questions for user led initiative, "When do you want to travel?". The user can answer this by providing the entire set of required values to the system. If, however, the system detects that the user is not so familiar with the system, slots from the next level down the hierarchy can be included, "On which date do you wish to leave?". If problems still occur, the initiative can be made finer and finer, right down to the lowest granularity, "What is the departure month?".

Complementary to adapting the dialogue initiative is the dialogue strategy. Chu et al. (2005) suggest that changing from one strategy to another continuously, and not just when errors occur, offers the best form of dialogue adaptation. If errors occur, or the user does not provide an informative answer as required, a more finite state based approach is taken to lead the user through the dialogue. If they do start to over specify answers or errors become less common, then the system switches to a more open frame based or plan based approach.

This approach is similar to what can be achieved with the JASPIS architecture[16]. Made up of different agents, with evaluators deciding which agent should be given the current task, these agents can be used to model different interaction strategies for the same task (Jokinen et al. 2002). This demonstrates how a truly adaptive and advanced dialogue system requires a more advanced architecture.

3.2.2 Adaptive Dialogue Systems: Introduction to User Modelling

User modelling has been the subject of much investigation since conversational systems started to appear in the 1980s. One reason for this emphasis on user modelling is the

[16] For a fuller appreciation of the JASPIS architecture, refer to 2.2.2.

fact that such models are necessary prerequisites in order for a system to be capable of exhibiting a wide range of cooperative dialogue behaviour (Wahlster & Kobsa 1989) and for selecting relevant content to be uttered back to the user if utilising dynamic dialogue (Carenini & Moore 2001). In human-human dialogue people acquire and use knowledge about their conversational partners, and for machines to interact in the same way they too need to acquire information about their conversational partner (Hjalmarsson 2005). A user model can contain a variety of facts about a user, such as the user's domain knowledge, the user's goal in asking a question, and various attributes about the user that might help a system, both its problem solving activity and its generation process (Paris 1993). It is therefore a knowledge source about the user which contains assumptions and beliefs that may be of relevance to the dialogue systems.

User modelling can range from very simple to the very complex. Simple models can be acquired by allowing the user to select their interests and create a profile from a list of options presented to them, which are usually related directly to subject and topic areas that are being modelled against the user. More complex systems however will not ask the user to explicitly create a profile for them, but will create models implicitly based on the user's actions, viewing and browsing habits. Brusilovsky & Tasso (2004) describe Information Filtering as a 'listening and learning' approach where the system first understands what the user wants, then evaluates whether a document is relevant or not according to their model, and finally updates the user model based upon feedback after the document has been delivered.

With the widespread use of the Internet, and its exponential growth since its birth, there is now a vast amount of information online. By utilising a user model, content can be

delivered to the user with a higher degree of accuracy, as only content that the user will be interested in will be delivered to them.

Information filtering can also be confused with other forms of information access based on user modelling, so it is important to distinguish between the types and remove ambiguity as to what constitutes information filtering. Proposed by Brusilovsky & Tasso (2004) are four different classifications of information access methods; information filtering, information retrieval, hypertext browsing, and information visualisation. Whereas information filtering is the delivery of content based upon a user's interests, Information Retrieval is the return of ranked links based upon relevance to the user model, to be discussed further in Section 3.3. Hypertext Browsing and Information Visualisation are outside the scope of VoiceBrowse, and so will be excluded from this review.

3.2.3 Adaptive Dialogue Systems: Adapting Content

User Modelling, specifically Information Filtering, has been proposed as one way to adapt the content of output prompts from a dialogue system to the user. Research from Kyoto University has demonstrated this by implementing a user model to classify a user according to three different measurements: skill level, knowledge level and degree of urgency (Komatani et al. 2003). The skill level concerns the user's expertise relating to using the system. The knowledge level concerns the user's expertise in the domain, while the degree of urgency is how quickly the user needs the information from the system. The amount of details, and content of the prompts, will be adapted towards the particular user who is currently interacting with the system, with respect to these three user attributes.

User modelling allows a higher level of dialogue adaptation than that based on dialogue indicators and performance. Although adaptation was performed before, users would still be getting the same prompts based on how they were classified by the system, novice or expert. As user modelling uses the user's qualities to adapt the dialogue, each dialogue will be truly unique for that user. Komantani et al. (2005) use the skill level and urgency of the user to adapt the dialogue initiative, whilst the knowledge level of the user is used to adapt the content of the output prompts, adding more domain specific information if needed for the user.

Adapting dialogue based on user qualities is obviously more beneficial therefore than basing the adaptation technique on dialogue performance. For a further application of integrating user modelling with dialogue systems, please refer back to Section 3.1.2 where work by Pargellis et al. was presented which utilised user modelling for the delivery of content from the Internet through spoken dialogue. Pargellis et al. was focused on adapting the content of the dialogue rather than the initiative and prompts like Komatani et al.

With regard to mobile based devices, user modelling is seen as very important due to the larger constraints of a smaller interface for displaying information. Presently this mobile work largely includes in-car navigation and conversational devices with the aim of limiting the cognitive overhead of interacting with such a system. In this in-car domain, developers face very specific challenges associated with user modelling, including identifying and retrieving the current user and associated model, creation of new models for new drivers, updating and enhancing the models through limited dialogue and identifying the most appropriate information to include and use in a model (Bernsen 2003).

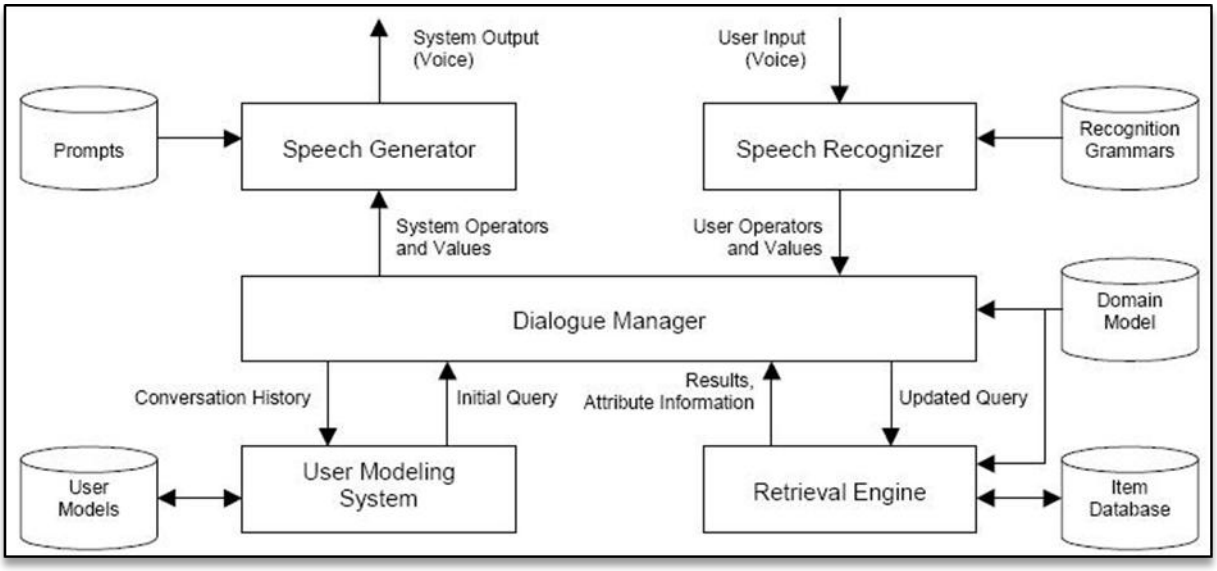


Figure 3.4: Adaptive Place Advisor Architecture

Early work in this domain includes The Adaptive Place Advisor, joint work between DaimlerChrysler Research and Stanford University, which is a system for advising drivers through conversation of restaurants in a certain area (Göker & Thompson 2000). The system's architecture, presented as Figure 3.4, shows how a user model can interact with a dialogue system. In addition to the normal components of spoken dialogue architectures, such as a speech recognisers and dialogue manager, it also contains retrievals engines specialised for the system to access the specifically structured items in the database, and a user modelling system which interacts directly with the dialogue manager. Conversation histories are passed to the user modelling system which updates certain attributes in the model, such as cuisine, price range and parking availability. Queries from the user are then passed to the user model, which refine the queries based on relevant attribute-values pairs found in the user model.

Early versions of the user modelling system did have certain limitations, such as the lack of ability to combine attribute-values pairs, and therefore often suffered from refining user's queries too much, limiting the results fetched by the retrievals engine. Later versions however, implemented included a more powerful user modelling system,

capable of relaxing certain constraints, based on preference orders of the attribute values pairs (Thompson et al 2004).

Fischer et al. present an alternative hybrid content based approach of gathering explicit user preferences and also updating the user model based on implicit information inferred from the user interaction with personalised web based information delivered to an in-car system. Evaluations show however that a large number of dialogues is required to achieve a high level of performance with regard to the user model (Fischer et al. 2007).

3.2.4 Adaptive Dialogue Systems: Research Issues

Dialogue management and content can be adapted based either on dialogue performance and indications of problems arising, or on the user's qualities and interests acquired by some form of user modelling or learning algorithm. This is comparable to the human method of adaptation during interaction, first by short term adaptation based on dialogue, and then on content and topics in the long term as the conversational partner learns the profile of the person.

This can also be classified, not in terms of short term and long term adaptation, but task oriented or content oriented adaptation. It is a trend that, when involved in task oriented dialogue, the dialogue strategy is adapted to match the user's perceived experience. In theory, this should help the user solve the task more efficiently by providing the relevant amount of guidance and flexibility. For example, if the user is 'expert' with respect to having prior use of the system, then the system should allow them to take the initiative, or not confirm each value as it is elicited. If the user is 'novice' however, having limited knowledge of the dialogue system, then the system should provide more

help and guidance which, although increasing the average length of the dialogue, will result in a successful task completion.

Compared to a system that delivers informative content however, it is the content itself that needs to be adapted. When considering a system that delivers content through dialogue however from the Internet, prompt adaptation is made somewhat more complex due to the wealth of information available online, the interest in which will differ from user to user. Pargellis et al. addressed this, although their research was more focused on information extraction from an online source, and so the area of user modelling was simulated to an extent by getting the user to state their own interests and subject area explicitly. This is a basic form of user modelling, and less advanced than implicit methods of gaining a user profile. This is one area in which the work delivered by Pargellis et al. could be improved, however it would be of more importance and value if the system could be implemented in a more general way, rather than just using the one web site for the source of the content.

Furthermore, information available online consists of both task and information based content. In application terms, this means the system could either be engaged in task oriented dialogue, or content based dialogue. Therefore, to be truly adaptable, dialogue features as well as content will be required to be adaptive. Some form of heuristics adapting dialogue style and user modelling will also be required.

In summary, lesson learnt from current adaptive dialogue research include:

- Using interaction parameters throughout the interaction provides a mechanism to adapt the dialogue initiative and strategy as needed, usually in task based dialogues.

- Outputted content from a dialogue system can be adapted with the use of a user model for information-based dialogues.
- Both elements and approaches are required to be implemented due to vast range of both task and informative based content that can be found online.

3.3 Information Retrieval

As discussed later in Chapter 4, VoiceBrowse will make use of techniques from the area of Information Retrieval (IR), and so a brief overview of this area is included.

IR systems usually employ one or more classification and retrieval algorithms to explore a large collection of documents and return the most appropriate based on the user's query. Often done by way of keywords, IR applications range from everyday public use to tools assisting and supporting specialists in their place of work, such as an engineer who might use IR to locate information in the manuals for a large project (Witten et al. 1999).

Providing a natural and intuitive method for accessing vast amounts of information, a dialogue system provides an enhanced alternative interface for use within an IR application. By using a natural and non-programmatically based language, retrieval queries can be constructed through collaboration between machine and user. The interactive nature of dialogue can suit IR applications where predefined queries need to be constructed, as often an incomplete query is presented to the system leading to some uncertainty about the information required. Notable work with respect to this includes Zhong and Gilbert (2005), who concluded that users are not likely to present more than two key terms when engaged in dialogue, and therefore further utterances are required to construct the query.

In IR, it is common that term weights are given to each word in the document collection, with higher weightings given to key terms that appear fewer times, based on the assumption that the fewer times a term appears in a document collection, the more specific it is and can therefore help identify individual documents. Web based IR, such as online querying and searching, is faced with unique challenges and issues due to the greater amount of data available online and the greater number of simultaneous searches. Indexing, clustering and ranking algorithms for online documents are common methods used to realise web based IR (Kobayashi & Takeda 2000).

Sieg et al. (2004, 2006) present work known as ARCH, which although not based on dialogue, utilizes interactive query formulation and domain specific hierarchies to produce a 'richer and therefore less ambiguous query'. Using hierarchies provides an effective way of enhancing similarity measurements, especially where they are already available. Ganesan et al. (2003) show also how this method can overcome the problem of sparse data in IR.

Other researchers have also proposed their own versions of COSIM or tw calculations to enhance the performance above the baseline. Choi et al. (2005) enhance the COSIM also with semantic information, focusing their attentions on the semantic web and the metadata that it contains about documents. Their Semantic Web based Information Query System (SW-IQS) uses an ontology server to enhance the efficiency and accuracy of IR for unstructured and semi-structured data.

Alternatively other researchers have relied on WordNet; a well used and tried hierarchy of words, to provide for synonyms in queries - as demonstrated by Richardson & Smeaton (1995) and Feng et al. (2004). Bollegala et al. (2007) propose an alternative method of measuring similarity between words - rather than relying on precompiled

taxonomies, such as WordNet, they offer an approach of relying on page counts and snippets returned by the search engine Google[17]. This overcomes one major disadvantage of relying on WordNet; that is similarity between proper nouns may not be recognized e.g., 'Apple' is the name of a large computer company, but this sense is not included in the WordNet database (Bollegala et al. 2007).

3.4 Summary

Dynamic and adaptive dialogue research aims to overcome the limitations of current spoken dialogue systems, introduced in the previous Chapter. The limitations include the limited flexibility of the dialogue, static prompts and dialogue management, and lack of individualisation with regard to content.

An approach to overcome this is to separate the domain knowledge from the dialogue knowledge, a consideration being to ensure that the domain knowledge is well structured and accessible to the dialogue manager. A system designed to deliver personalised content from the Internet presents a challenge with regard to this requirement, as the content will vary in form, structure and type. Additional issues arise from a broad range of research areas, such as user modelling to tailor the information delivered for each user, and information retrieval to identify and fetch the content

To conclude, the Internet has become a hub of information for people, and for some it has also became a central part of their daily lives. There is great potential for a system that would automatically navigate to a specific web 'page', and, instead of displaying its contents visually to the user, perhaps engage both parties in a form of conversation related to the content. Usability consequences arise however, as spoken dialogue

[17] http:www/google.co.uk

systems do have limited scope for presenting vast amounts of information to the user. Additionally, whereas the spoken dialogue systems presented so far in the research are created for a specific purpose or task, a spoken dialogue system to access information will be required to manage and present information generically but meaningfully to the user.

This research will focus on developing an architecture to realise a spoken dialogue system that can utilise information from the Internet to encourage dialogue – in other words provide a dialogue interface for browsing the Internet.

Chapter 4: VoiceBrowse Introduction and Architecture

Following the review of the literature related to dynamic and adaptive dialogue systems, both needs and gaps were identified with respect to browsing online content through dialogue, and it is these gaps that VoiceBrowse aims to investigate and explore. Current dialogue managers developed to parse and understand one website are not highly portable to different websites - previous approaches therefore have been to limit the scope of the system to a smaller subset of websites. VoiceBrowse will attempt to make the first step at achieving truly generic dialogue across multiple domains and content types, a dialogue system capable of supporting interaction between human and machine based on any website or content.

This chapter first addresses the current research gaps by introducing a set of requirements that will provide the basis for the functionality and research contributions of VoiceBrowse, and also a foundation on which to evaluate the system once it has been implemented. Next, the VoiceBrowse architecture will be presented followed by a discussion of its main components and the proposed solution of using RSS and API feeds to cater for the online information retrieval required. The remainder of the chapter will discuss the focus of the research, and indicate those components of the overall architecture that will be considered beyond the scope of the dissertation.

4.1 VoiceBrowse: Requirements Derived From Research Gaps

The proposed research aims to support a level of dialogue and interaction more extensive than that which more traditional spoken dialogue systems architecture currently allow. An architecture that will support the areas of dialogue management, content delivery, user modelling and multimodality will be required to realise the

following set of requirements, identified in the review of the literature. Not all of the requirements will be implemented in VoiceBrowse, and Section 4.9 will discuss this further.

Requirements – system functionality

1.1 The system should cater for voice input and output.

1.2 The system should allow the user to interact with existing Internet based sites.

1.3 The system should cater for common daily web based tasks e.g., reading news, flight bookings etc. using the Internet to accomplish the goals. This should include both task-based and information-based dialogues.

Research requirements – technical contributions

2.1 The dialogue manager shall interact with various domain and content types from various sources.

2.2 The online websites shall not be pre-processed or structured in any specific manner.

2.3 The system shall interact with multimodal outputs where appropriate.

2.4 The system shall be available for interaction on various devices with different capabilities.

2.5 The system shall personalise the requested content for different users.

2.6 The system shall be easy extendible with new types and sources of content.

Research requirements – usability contributions

3.1 The system shall be usable in an efficient and effective manner.

3.2 The functionality of the system should be quick to learn and easy to use.

3.3 The system shall provide help to the user when required.

3.4 The system's output should be meaningful to the user, and generated in an appropriate manner.

4.2 VoiceBrowse: Applications and Users

The potential for such a system extends beyond the research issues and questions to be addressed, as it is possible that there would be benefit in deploying VoiceBrowse in the real world. A current limitation noted for the graphical web browser is its reliance on the use of the mouse and keyboard, both of which require a home computer setup, and the ability to use such devices. To extend the Internet beyond the graphical browser without requiring either a computer or the knowledge and ability to provide hands on interaction would improve the accessibility of online content greatly.

Without requiring a home computer to access online content, or other similar hardware such as a PDA or smart phone, users would be able to access the Internet from a simple device such as a telephone. For commercial use, companies, and even cities, could implement dedicated VoiceBrowsing booths that would allow the simple finding and retrieving of online content related to their company or tourist information.

Moreover, and perhaps with greater promise, is the accessibility of online content to those who either do not have the knowledge of how to use current graphical browsers, or are incapacitated from doing so due to mobility reasons. Removing the need for hands-on interaction will open up the Internet to a much wider audience than is currently possible. It is these scenarios and targeted users that VoiceBrowse will seek to aid and support.

4.3 VoiceBrowse: Use Cases and Example Dialogues

To help drive development and assist with realising the above requirements, a set of use cases were produced to illustrate the functionality of VoiceBrowse. Figures 4.1 and 4.2 show the high level functionality of the system, allowing the user to accomplish either

Figure 4.1: VoiceBrowse Use Case – System Functionality

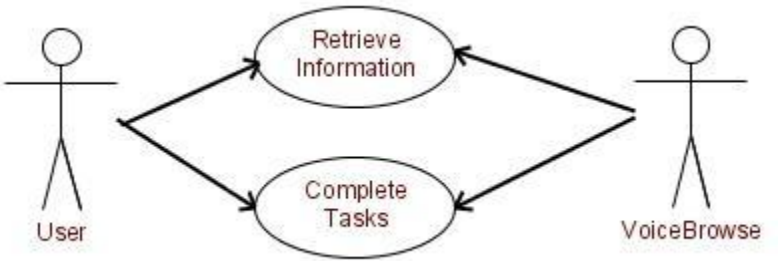


Figure 4.2: VoiceBrowse Use Case - Available Dialogues Types

task-based dialogues or information-based dialogues driven by the utilisation of online content. It is the responsibility of VoiceBrowse to interpret the user's requests, extract the information from the various online sources, and then prepare the content in an appropriate ,manner for spoken dialogue. The above Use Cases show that it is the collaboration between user and VoiceBrowse that allows the relevant dialogues to be accomplished.

Figure 4.3 shows the use case for the roles available to the user and system during an information-based dialogue. This Use Case illustrates that the user can listen to short synopses of stories retrieved from web pages, or the user can request the main body of the story to be retrieved from its online source. Additionally, the user should be able to navigate around the story descriptions available, and also the main body of the story. It is VoiceBrowse's responsibility then to manage the current environment with regard to

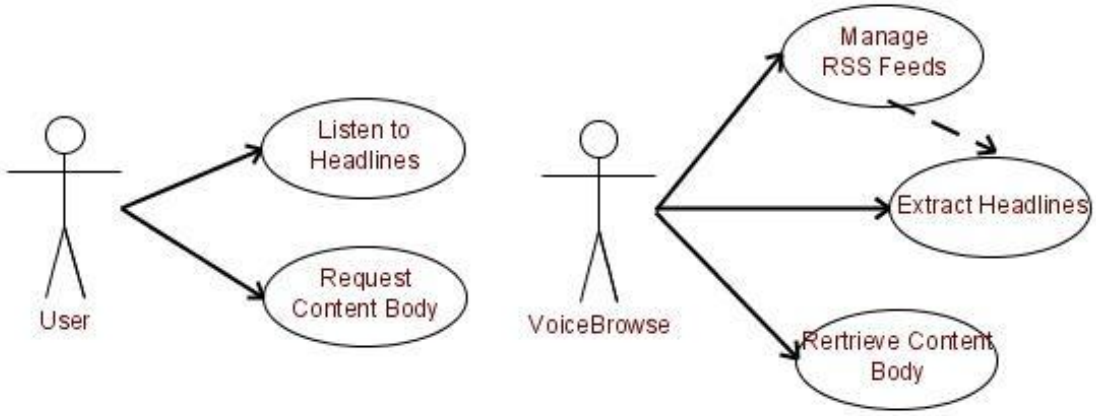


Figure 4.3: VoiceBrowse Use Case - Information-Based Dialogues

content, in particular the RSS feeds available. As discussed further in Section 4.5, by using XML technology a novel approach of utilising RSS feeds can provide a mechanism for delivering online content through dialogue. At this point however, the Use Case simply provides a pictorial illustration of its responsibility during the dialogue.

Similarly in Figure 4.4, a Use Case is presented to summarise the functionality and responsibility of both system and user during a task-based dialogue. This Use Case simply shows that, during such a dialogue, the functionality to provide the values required to complete the task is the responsibility of the user, after which they can

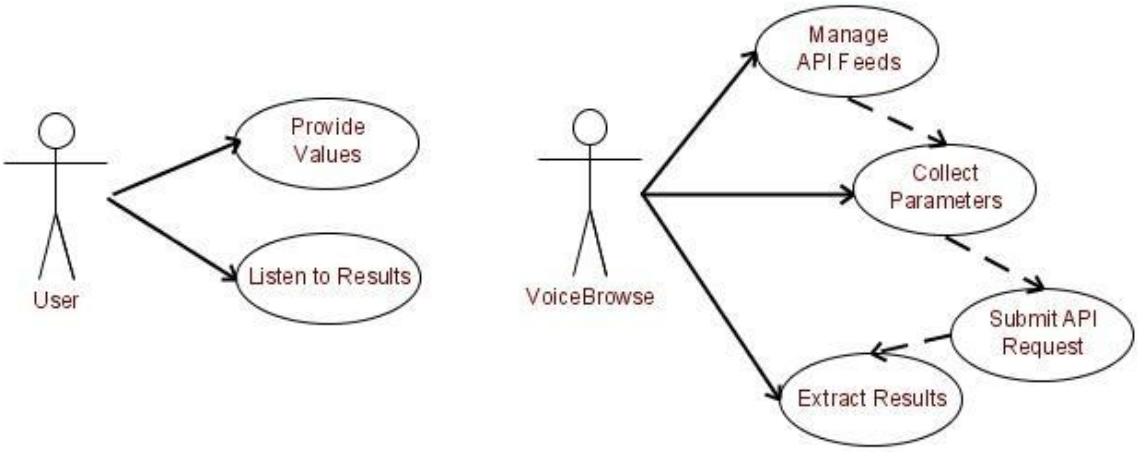


Figure 4.4: VoiceBrowse Use Case – Task-Based Dialogues

User (U) 1: Good morning, what is the news today?

System (S) 1: Good morning. The news today is "Crash landing for cargo plane at Birmingham airport", "Banking giant to create 145 jobs", "Hospital ward could close in Co. Donegal". The sports news is "Inter Milan begins negotiations with Lampard".

U2: Tell me more about Lampard.

S2: "The Serie A club Inter Milan is believed to have sent a fax to Chelsea to begin the process of reuniting the 30-year-old with former Blues coach Jose Mourinho."

U3: Actually, I have just remembered I have to book a flight to London for next Thursday; can you get me a list of available flights please?

S3: There are flights to London Heathrow and Gatwick from Belfast City Airport. The flights times are 09.00, 10.00, 10.15, 11.00.

U4: And what about return times that evening?

Figure 4.5: VoiceBrowse Dialogue

navigate through the results returned by the vendor or content provider. It is therefore VoiceBrowse's responsibility to: produce the appropriate dialogue to collect the required values from the user; submit the appropriate request for content to the correct vendor; and then extract and prepare the results into a meaningful form for the user.

To further demonstrate the proposed functionality of VoiceBrowse, consider the hypothetical dialogue shown in Figure 4.5. This dialogue illustrates all of the current research gaps identified in the literature that have formed the requirements of VoiceBrowse, and it exemplifies the Use Cases previously presented. Specifically VoiceBrowse should allow the user to engage in dialogue based on the delivery of content in a multimodal form that has been automatically generated from web site content, and personalised for that particular user. This is shown in System Utterance 1, reacting to the question from the user regarding the news headlines. In response to this, the system presents some general news headlines to the user, and also one football news headline. The content delivered to the user has also been personalised, due to the large

amount of information available online. The user model here has tailored the system's response to this particular user, delivering the news headlines that it now infers the user is interested in. Each person will have their own interests and relevant topics, and so the dialogue system will need to know which content to deliver and which content not to deliver when the user engages in dialogue. Different users will have different responses from VoiceBrowse as they interact with the system over time.

The user can then respond to the content delivered by the system, either by further questioning the system regarding the content or shifting to a different topic entirely, both these reactions illustrated in User Utterance 2. Here, the user has became interested in the headline regarding Chelsea midfielder Frank Lampard, and asks for more details regarding this story. However, during the output of the story's main body, the user remembers a flight that is required to be taken, and initiates a new dialogue with VoiceBrowse to complete this task.

Additionally, this then demonstrates the key element of VoiceBrowse that it is to be generic, not be fixed to one domain, content, or content source, but rather able to deal with a wide range of content types from a wide range of online sources. This is shown throughout the dialogue, as user and system interact with one another in both information-based and task-based dialogues.

Current dialogue systems are not able to handle this form of opportunistic generic dialogue, driven by unstructured content types from numerous sources. These gaps have been summarised above, and presented as the requirements of VoiceBrowse, and an architecture has been proposed to realise the system.

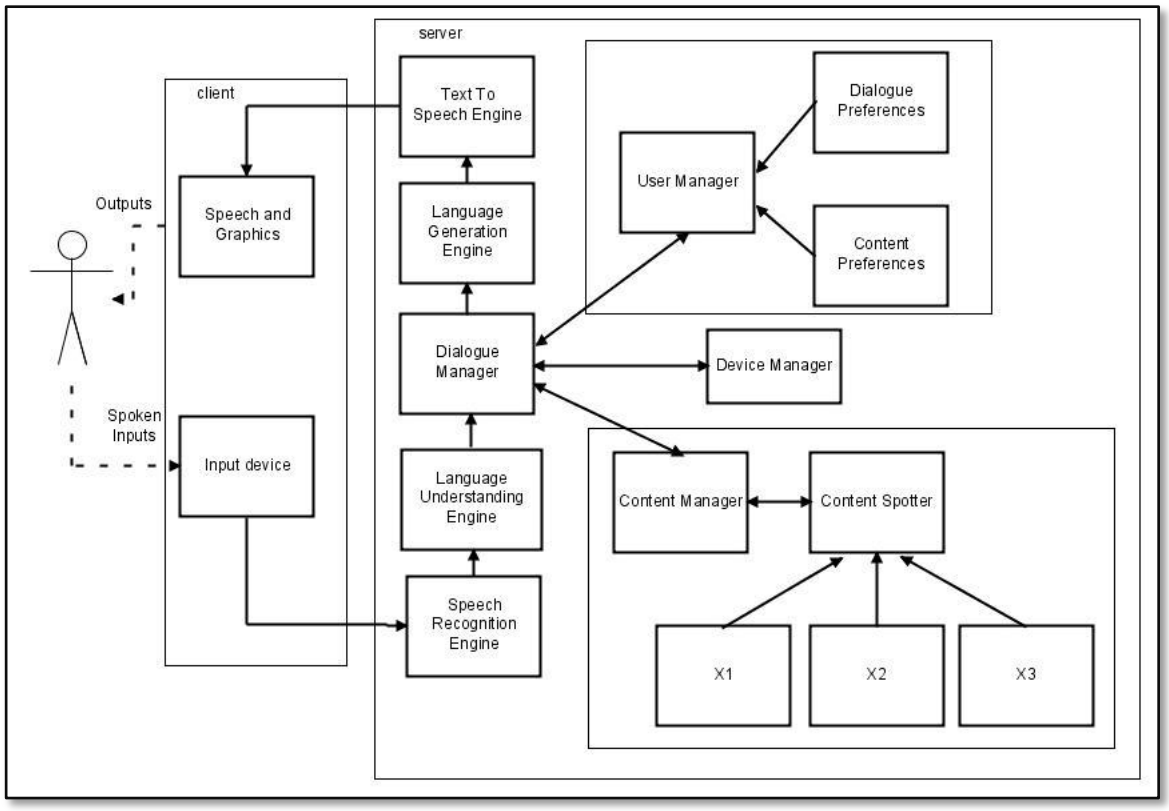


Figure 4.6: VoiceBrowse Architecture

4.4 VoiceBrowse: Architecture

The above requirements summarised the research gaps identified during the literature review, providing a means for various novel contributions to dialogue research. As stated in Chapter 2 an important aspect of any advanced dialogue system is the architecture which will support the advanced features. For VoiceBrowse, in addition to the Dialogue Manager required for such systems, it was decided to also include a Content Manager and a User Manager to provide the various content and user related functionalities specified above. A conceptual architecture was produced, illustrated in Figure 4.6, incorporating many features that have proved beneficial in similar dialogue systems reviewed in Chapter 2. The architectural components are represented as square boxes and the interactions between components as directed arrows.

The architecture itself was developed using a client-server paradigm, where the client represents the device that the user interfaces with during the interaction, and the server represents the logic and system side that contains the VoiceBrowse system. Designing the architecture as client-server facilitates the devices and the system to work independently of each other, allowing devices to be added and removed without affecting system functionality.

On the client side, the device will contain some form of input device that will accommodate spoken input, such as a microphone, and will cater for multimodal output through graphics and sound by use of graphical displays and speakers. It is proposed that a form of Device Manager will reside on the server that will manage the available devices and their capability within the current VoiceBrowse environment. Combined with the independence of the device and server, this will allow the server to output the current content to the most appropriate device in the environment.

The server side contains the main components required for a dialogue system, namely a speech recognition engine, a language understanding engine, a language generation engine and a text to speech engine. In addition, VoiceBrowse will contain three modules that will be unique to the system and provide the foundation for the functionality identified in the requirements: the Dialogue Manager, the Content Manager and the User Manager.

The interaction will start with the input component on the device capturing the user's utterance. This will primarily be a speech recogniser, but additional input components can be added to the device for multimodal input. This will then be passed to the Dialogue Manager, which will extract the user's intention from the utterance. Requests for content will be passed firstly to the User Manager to further refine the query based

on information contained in the user model, before passing the request to the Content Manager. The Content Manager is charged with managing and retrieving information from the Internet, and returning its results back to the Dialogue Manager via the language generation engine, which tailors the output into a meaningful form for dialogue.

To act as a bridge between the Dialogue Manager and the unstructured online content, a novel solution utilising RSS feeds, a well defined XML specification, provides a standard method for querying and accessing available online content. Described further below, RSS and API feeds are represented in Figure 4.6 as X_1, X_2, ..., X_n, and it is the role of the Content Spotter to select the most appropriate feed that has the necessary content to handle the current query. This, and the role of the other managers with regard to the use of RSS and API feeds, will be discussed further below.

4.5 VoiceBrowse: Utilising RSS and API feeds

The major challenge associated with Internet browsing through dialogue is catering for the vast array of content available online. The various web site structures and the numerous types of contents available create insurmountable problems for current dialogue managers that are created for a specific task and content. To address this issue, suitable processes are required to parse and extract information into a standard suitable for interaction with the Dialogue Manager. As this is anticipated to consume resources and time to achieve, another solution was conceived, one that reuses existing web technologies already in place to achieve the same results.

A novel approach to retrieving the content from online sources is to utilise APIs and RSS feeds. By doing so, one can retrieve information from online sources for use in a dialogue system without having to consider the challenges and issues associated with

```
<rss version="2.0">
<channel>
<title>BBC Sport Football UK Edition</title>
    <item>
        <title>We're still in title race - Grant</title>
        <description>Chelsea boss Avram Grant says his team can still win the Premier League title despite losing
            ground on leaders Man Utd.
        </description>
        <link>www.bbc.co.uk/.....</link>
    </item>
    <item>
        <title>Parry keen for talks with Benitez</title>
        <description>Liverpool chief executive Rick Parry is set to hold discussions with boss Rafa Benitez to clear up
            any problems.
        </description>
        <link>www.bbc.co.uk/.....</link>
    </item>
</channel>
</rss>
```

Figure 4.7: An Example RSS Feed

the information not being stored in a standard way. Already it is common to extract the content from a RSS feed and place it into a (x)HTML page for reading. To understand RSS further, consider the simplified RSS example overleaf in Figure 4.7. Although there are additional elements and attributes that make up a legal RSS specification in addition to those shown in Figure 4.7, what is important to note is that an RSS document contains one or many <item> elements,

each of which contains one <title> and one <description> element. The entire RSS document is specific to a particular content type, in this example Football from the BBC, and the current stories that make up the content are represented by the numerous <item> elements.

As RSS feeds are available for many different content types from different sources, the <description> elements provide a method for real-time information retrieval and extraction in a dialogue system. However, as the <description> element often only contains the introduction to a particular story, the challenge of interfacing one dialogue manager with various web sites of different structures still arises, due to the need to access the web page containing the story, indicated by the <link> element, if the user

requests more information regarding that particular story. The solution to overcome this will be discussed in Chapter 5.

In addition to RSS feeds, APIs are also provided by a range of different service and content providers. APIs provide an alternative to the front end interface for interacting with the content provider's core services. For example, an API from eBay can allow for the searching and bidding of items from an external web site, or a BBC API can allow for the searching and retrieval of information regarding the scheduling of programmes. Whereas RSS feeds provide information and content specific to a certain topic, APIs support the completion of certain tasks relating to the vendor's back end systems.

Like RSS, APIs simply provide the specification required for retrieving data from a content provider, and it is up to the developer to decide how to embed the API in an application, and the means by which they collect the required parameters from the end user and then package these into the API query. Returned results from the content provider in response to the API submission are in the form of an XML document, which can be transformed into a variety of specifications using XSLT or an appropriate scripting language. However, unlike RSS, the XML specifications of APIs are vendor specific, and can therefore not be accessed in a standard way. This will be an issue for consideration, discussed further in Chapter 5.

It should be noted that the current description of APIs relates to available APIs provided by online content providers for web development use. Whilst this is the primary scope of the research, it should be mentioned that APIs by definition are programmable interfaces, and not limited to online content providers. Many applications providers, such as Microsoft, make APIs freely available for developers to interact with applications such as operating systems and office applications. There would be

consideration for future work in integrating APIs for controlling devices and computer systems with a dialogue manager to extend such a system beyond the use of providing online content to end users.

It was decided that the APIs and RSS feeds be treated as plug-ins to the system; that is to say each plug-in will represent a content source, and whatever content is relevant to it. For example, if a user requests a news story or headlines, the BBC RSS[18] plug-in would be chosen here. Alternatively, if a user wishes to order some books from Amazon, the Amazon ordering API[19] would be used. Each plug-in will be encapsulated, so that they can be added and removed from the system without affecting the other plug-ins.

By using RSS feeds, it is a belief that a dialogue manager can be developed to generically extract domain knowledge from numerous sources of unstructured online content. Furthermore, as RSS feeds provide information-based content from various providers, and by contrast APIs provide a means of completing various tasks from various providers, this gives a distinction between two different types of dialogue, each of which will be handled differently - information-based dialogues, such as requests for news and weather, will be provided by RSS feeds, and task-based dialogues, such as booking cinema tickets and flight tickets, will be handled by the APIs.

This approach of utilising API and RSS feeds as a structured bridge to unstructured web sites to support dialogue management has not been studied previously. Making use of such programming mechanisms will remove the majority of issues arising with

---

[18] feed://www.bbc.co.uk/go/homepage/int/ne/nrss/log/i/-/news/rss/newsonline_uk_edition/front_page/rss.xml

[19] http://www.amazon.com

retrieving and extracting online content, offering a clearly defined, structured and accessible way of adding content into a spoken dialogue system. An introduction to the three managers now will be presented, introducing the workflow through the Architecture and the rationale for its structure. How the various feeds will function with the Content Manager will be described in detail in Chapter 5.

4.6 VoiceBrowse: User Manager

An important aspect identified in the literature review with regard to browsing a vast amount of content such as the Internet is that of user modelling. It is for this reason that a User Manager has been included in the VoiceBrowse architecture. After a request for information has been received by the system, it will be passed on to the User Manager which will enhance the request with information from the user model concerning the user's interests and dialogue preferences, such as number of result items, preferred output device and verification strategies. Once the request has been handled by the Content Manager, due to the enhancement by the user model, the results returned will be relevant for that particular user. For example, if the user makes a request regarding news headlines, the user model will refine this query to only include business and football headlines as the User Manager has inferred these interests from the dialogue history.

This is important as only a limited amount of information can be output to the user through voice at any time, so it is imperative that that information is relevant and meaningful to the user. User modelling provides a mechanism to acquire and utilise a collection of user's preferences and interests, allowing informational queries to be further refined and optimised producing a more relevant result set for the user. Due to the limited amount of content that can be conveyed in voice prompts when compared to

graphical displays, user modelling algorithms need to be more accurate and have better performance. The methods for presenting related information through dialogue also need to be addressed.

Consider recommender systems which, when used with a graphical interface, present numerous pieces of items related to the user's browsing habits on screen. If the user is interested in them, then the algorithm is a success. However, if the user is not too interested in the recommended items, then they can simply choose to ignore them. This displaying of unrelated items is not a huge upset to the user, as they can simply just look over them.

When compared to the use of a recommender system in a spoken dialogue system, it would be a lot less satisfying and bothersome for the user to have to 'listen' explicitly to a list of recommended items that they are not interested in. Recommender systems therefore do present some different challenges for researchers of user modelling to consider when used with spoken dialogue interfaces. One advantage, however, would be that the user explicitly has to tell the system that they are not interested in the recommended items. This would provide good feedback for the recommender algorithm, and allow it to refine its model of the user's interests. This is not always possible with a Graphical User Interface, as the user simply looks over any unrelated recommended items.

4.7 VoiceBrowse: Dialogue Manager

The dialogue manager will receive the input string spoken by the user, and decide what action to perform next in the interaction. A key component alongside the Dialogue Manager will be the Language Understanding Engine, which will extract the relevant meaning from an input utterance. Due to VoiceBrowse catering for dialogue with

multiple content types and tasks, the role of the Language Understanding Engine will include functions such as discovering the intent and relevant content communicated by the user, so that the Dialogue Manager can act appropriately. If it is a task based interaction, then more information may be needed from the user. If it is a request for content, or all required information has been elicited from the user and understood by the dialogue manager, then the request will be passed to the User Manager to retrieve any recorded preferences that may be utilised. Finally, the request will be passed to the Content Manager to retrieve the information from the Internet.

On the other side of the interaction is the output from VoiceBrowse, and the Dialogue Manager also has the responsibility of preparing the retrieved content to be delivered to the user in a meaningful and sensible way. In a traditional dialogue system developed for one specific task, or those developed for a static domain, outputs are easily anticipated and catered for through the use of templates at each dialogue state. However, with VoiceBrowse being both multi-domain and dynamic, the output will change at each state, for each user, and additionally be content specific; news stories will be required to be output differently from flight searches, for example. The Language Generation Engine will decipher what type of content is to be presented to the user, how it is to be presented, and interact with the Device Manager to decide if the current interacting device is to be used or infer that another device should be used. The Device Manager is charged with looking after the different devices currently contained in the system environment and available to it, so that the Dialogue Manager can know what output formats can be supported by the device that the user is currently interacting with. This will also allow the Dialogue Manager to suggest that the user should switch to a different device that is more appropriate in the environment than the current one.

To facilitate this communication between Dialogue Manager, Language Generation Engine and Content Manager, a standard XML specification will be developed. This standard will contain the content returned by the Content Manager, allowing the various managers and modules of VoiceBrowse to operate independently of one another. A more detailed discussion of the Dialogue Manager is provided in Chapter 6.

4.8 VoiceBrowse: Content Manager

The requests from users will eventually be passed to the Content Manager, which manages the online content available to VoiceBrowse. Similar to the Domain Spotter of the Queen's Communicator (O'Neill et al. 2005), and the Evaluators in the JASPIS Architecture (Turunen 2004), the Content Manager will review the collection of feeds and choose the most appropriate one to be returned to the user based on the input query. If a feed is available to appropriately handle the interaction, the Content Manager will extract the relevant content from this feed. It will then be passed back to the dialogue manager for presentation to the user. Once the dialogue has been prepared in a meaningful way by the content generator, it will be delivered back to the user. As previously mentioned, it may be the case that additional content is still required to be extracted from the source web page, in which case the issues surrounding information retrieval from numerous unstructured sources still arise. This will be addressed in Chapter 5 along with a detailed discussion of the Content Manager.

4.9 VoiceBrowse: Research focus

The proposed architecture extends beyond the remit of dialogue research by incorporating additional issues associated with research in other areas, such as multimodal dialogue, user modelling, and online information retrieval from

unstructured sources. Each of these areas presents their own unique challenges. Pargellis et al. (2004) additionally commented that these areas are made even more challenging in the diverse environment of the Internet, where content types are not standardised.

It is proposed therefore to narrow the architecture development to those areas specific to the research shortcomings identified with respect to dialogue, and therefore only the Dialogue and Content Managers will be implemented, with the modular nature of the architecture allowing the future development of the User and Device Managers.

4.10 Summary

This chapter has presented the requirements of VoiceBrowse that have evolved from the gaps and shortcomings identified in the literature review:

- Dependence on a specifically created and structured knowledge source for dialogue.
- Dialogue restricted to a particular domain or content type.
- Lack of dialogue usability knowledge with regard to browsing the Internet through voice.

In summary, VoiceBrowse will realise truly generic dialogue, utilising content not of a specific type or from a specific source but extracted from unstructured online sources to support both task-based and information-based dialogues between human and machine. The broad spectrum of possible content types will be problematic. Structured web services such as APIs and RSS feeds will be utilised to retrieve the content from the Internet, implemented as ‘plug-ins’ to VoiceBrowse. As new API and RSS plug-ins are made available for evolving content types and sources, they

can simply be integrated into the system without affecting functionality or the operation of the other plug-ins. However, generic methods are required to engage the user in conversation based on online information accessed from more than one source – this introduces associated issues and problems here with natural language understanding that will also be addressed.

Chapter 5: VoiceBrowse Content Manager

This chapter will introduce the Content Management component of VoiceBrowse. Its function and roles within the overall architecture will be explored, followed by a discussion on challenges to be overcome. The design and workflow of the Content Manager will be considered, followed by a preliminary experiment to evaluate the potential performance of the Content Manager.

5.1 Content Manager: Introduction

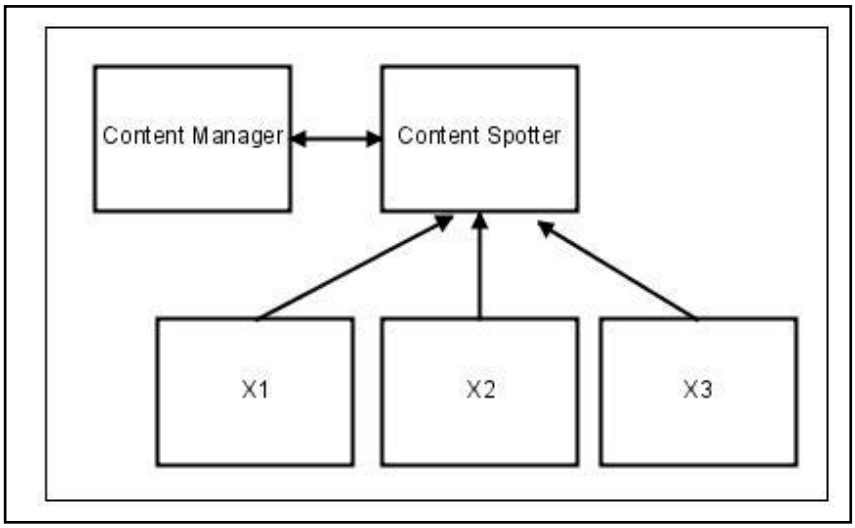


Figure 5.1: Content Manager

The Content Manager's primary role within the VoiceBrowse architecture is to make the online content from the Internet available to the Dialogue Manager (see Chapter 6). To provide for this functionality is a two-step process: firstly, to decide where the content should come from to satisfy a user's query; and secondly, to extract the relevant information from the online source.

The decision of choosing the most appropriate feed is the job of the Content Spotter, shown in Figure 5.1, with RSS and API feeds represented by X1, X2, and X3. This

polling or weighting of sub-components is similar to the domain spotter in the Queen's Communicator, and also the role of evaluators and agents within the JASPIS architectures (see Section 2.10), and is discussed further in Section 5.3.

Secondly, once relevant information from a particular feed has been identified, it is then the job of the Content Manager to extract the full body of content from the associated Web Site. To accomplish this generically would mean to overcome one of the limitations of current dynamic dialogue system identified in the research – that of the reliance on specifically crafted and well structured knowledge sources for dialogue management. It is proposed that by relying on structured RSS and API feeds as a 'bridge' to online web pages, identifying relevant online information becomes standard due to the set specification of RSS. Furthermore, as RSS feeds additionally contain URL information for each story represented, the location of the information to be extracted is also represented in a standard way. Therefore the second task of extracting unstructured information has been reduced to a series of standardised, multiple steps: given a URL, download the HTML located there; standardie the HTML to W3C conformity; and finally extract and output the main body of content to the user.

Many tools are freely available for 'cleaning' or 'standardising' HTML from web pages, such as HTMLtidy[20]. This vital step is required to overcome the possibility of attempting to parse illegal HTML documents, therefore encountering errors. Such tools also ensure that all bodies of text are encapsulated in relevant paragraph, or <p> tags, making them extractable by the XPATH query language. Once all HTML has been

[20] http://tidy.sourceforge.net/

made standard, the Dialogue Manager can treat all content of any source, type or structure as the same XML formatted document.

5.2 Content Manager: Issues and Concerns

Online information from multiple domains can create a number of potential issues and concerns when coupled with a spoken dialogue system, such as: how the API and RSS feeds can be 'plugged' into the architecture independently of the Dialogue Manager; and how can the dialogue manager evaluate and choose the most appropriate feed for a given request?

Regarding the former, RSS feeds can be made known to the Content Manager by including the URL of the RSS feed into the system. API feeds, however, are more challenging and require more effort due to their non-standard format. An API from Amazon will be entirely different from one from Expedia, for example - not only in terms of functionality offered and required parameters for operation - but also in how the API is represented in terms of mark-up and specification.

Consequently, due to the non-standard representation of APIs, the Content Manager cannot treat different APIs in the same manner, in contrast to the one method that can be used on any RSS specification. The interactions between the Content Manager and the different APIs, and also the RSS feeds, should be transparent to the Dialogue Manager – as illustrated in the architecture. The Dialogue Manager will interact with the Content Manager in a standard manner, irrespective of RSS or API feed in use. Extensibility of the domains should also be catered for with the encapsulation of the feeds and their communication with the Content Spotter conducted in a standard way. This will ensure that the feeds and Content Spotter can both be implemented independently allowing new feeds to be added to the system without affecting its operation.

Secondly, the polling or weighting algorithm required for the Content Spotter is made somewhat more complex for two different reasons: due to the ambiguity involved with natural language; and the fact that the utterance will contain more than one word for the classification process, all of which should be used to select the most appropriate feed.

Consider for example the utterance "What was that news about Heathrow today?". If each word is to be weighed against the available feeds, then 'news' feeds would return a high probability to handle the query, whereas 'travel' feeds may return a probability also, but lower, due to the presence of the word 'Heathrow'.

Further challenges arise when one considers that different content types also present different requirements from the user. A query concerning the news, for example, may not require any input parameters at all from the user, but a flight search query requires minimally at least a departure and arrival airport, and departing date. Although certain content types may not require parameters to be obtained from the user, certain content types may be more general than others. This leads to the issue of how much questioning a dialogue manager should perform before deciding that a request has been made in a general content area that is specific enough to satisfy the user. For example, the user may ask "What is the news?" - but should the system respond with news information, or further question the user to restrict this generic topic of news further? If so, the user could respond with a refined query to hear the 'Sports News' for example. Again VoiceBrowse could prove the sports news from an RSS feed, but maybe the system should enhance the query further, and ask the user for a particular type of sport, such as football. It is a fine balance between generic and specific queries when there are no boundaries to the information that can be accessed.

```
<rss version="2.0">
<channel>
<title>BBC Sport Football UK Edition</title>
    <item>
        <title>We're still in title race - Grant</title>
        <description>Chelsea boss Avram Grant says his team can still win the Premier League title despite losing
            ground on leaders Man Utd.
        </description>
        <link>www.bbc.co.uk/.....</link>
    </item>
    <item>
        <title>Parry keen for talks with Benitez</title>
        <description>Liverpool chief executive Rick Parry is set to hold discussions with boss Rafa Benitez to clear up
            any problems.
        </description>
        <link>www.bbc.co.uk/.....</link>
    </item>
</channel>
</rss>
```

Figure 5.2: An Example RSS Feed

5.3 Content Manager: Design and Process

When designing the Content Manager it is important to appreciate the content details available to it from RSS feeds. Consider once again the sample RSS feed, presented again as Figure 5.2. Information that is readily accessible from an RSS feed includes a short synopsis of a story, the title of a story, and the link to the full story itself. To facilitate polling or a similarity function, it would be first necessary to create a list of documents from all the RSS and API feeds available. This can be done programmatically by extracting the <description> and <title> elements from the RSS feed and inserting each into one XML file, creating an XML file such as 'documents.xml' that contains all the relevant information from all the available feeds – this is illustrated in Figure 5.3.

The result of the process is a document space that can be used as the basis for a similarity function. It was decided that the mechanism for performing this similarity function is the widely accepted Cosine Similarity (COSIM) function, which is seen in

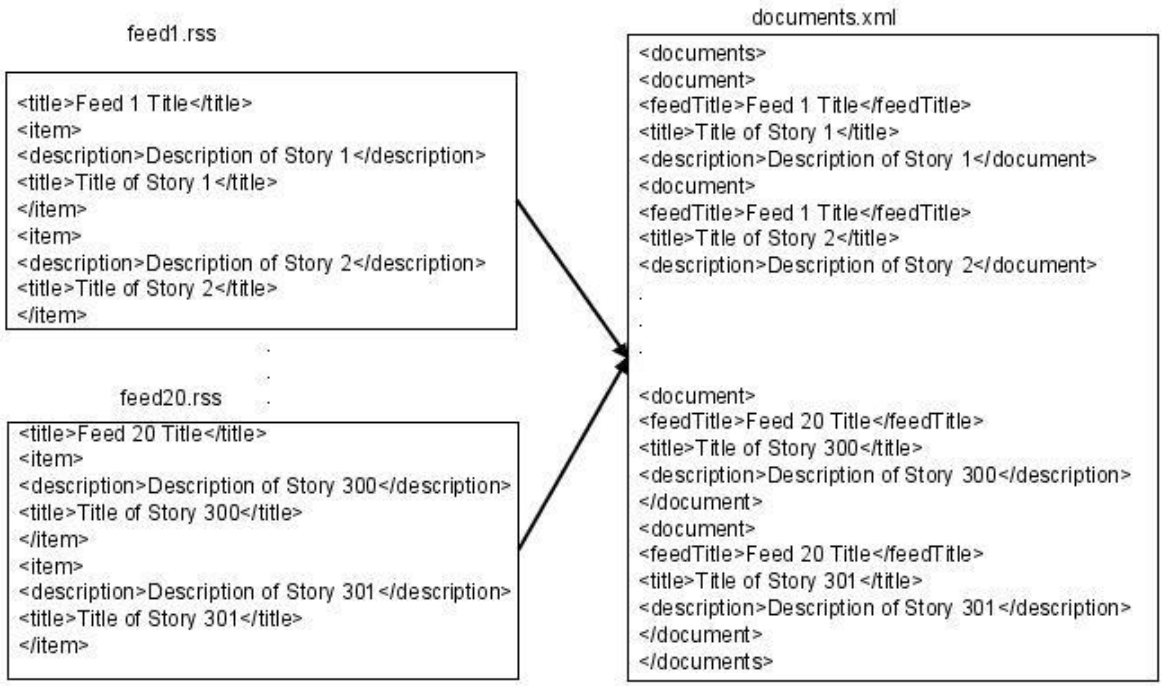


Figure 5.3: Document Space Creation From RSS Feeds

other dialogue systems such as call routing and spoken document retrieval systems (Ng et al. 2006; Chu-Carroll & Carpenter 1999). COSIM is defined below by (1) and (2);

$$cos\vartheta = \frac{v1 \cdot v2}{\|v1\|\|v2\|} \qquad (1)$$

$$V_d = \left[W_{1,d}, W_{2,d}, \ldots . W_{N,d}\right]^T \qquad (2)$$

where *W* is the term weight (tw) for each term within a document *d* and *V* is the vector magnitude of document *d*. By measuring the cosine of the angle between vectors from a standard origin, such as (0,0), a numerical value between 0 and 1 can be obtained as to how related or similar two documents are. As the angle decreases between vectors, the closer the cosine value is to 1, as COS 0 = 1. Traditionally in Information Retrieval, the product of the term frequency and the Inverse Document Frequency (IDF) is utilised to obtain the tw (3);

$$W_{t,d} = tf_t \cdot \log \frac{|D|}{|\{t \in d\}|} \qquad (3)$$

where tf_t is the term frequency of a term ***t*** in a document ***d***, and ***D*** is the total number of documents in the document space. $\log \frac{|D|}{|\{t \in d\}|}$ defines the IDF. The process of this similarity function is summarised in Figure 5.4:

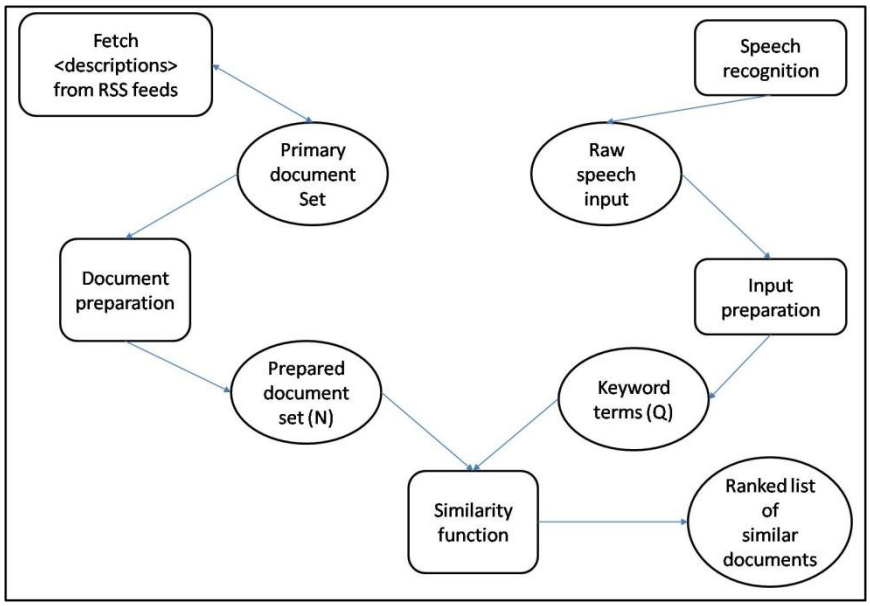


Figure 5.4: Content Spotter Process

In order for the Cosine Similarity function to be effective, the query (Q) that is being compared to the document space (N) must be of the same format. Pre-processing is required on both the user input side and content management side to ensure this, such as converting all characters to lower space.

5.4 Content Manager: Content Spotter Evaluation and Enhancements

To analyse the initial performance of the Content Spotter and the COSIM function before implementation, a series of preliminary experiments were conducted to investigate its effectiveness. The experimental setup is described in Table 5.1.

EXPERIMENTAL SETUP	
Total number of RSS feeds	22
Unique feed providers	7
Unique domains represented	14
Total number of documents fetched	406

Table 5.1: Preliminary Experiment Setup

Throughout the experiments, 22 feeds in total from 7 different content providers, such as BBC, Yahoo and NASA were available to the system for extracting information. This equated to 406 different <item> elements, or documents, across 14 different domains, where a domain is classified as a distinct topic, such as political or world news.

Using a simple web form setup, a phrase was entered into a text box that would then be submitted to the COSIM function. The returned ranked list of document were then displayed in descending order on a results web page, with the highest ranked document and its weight being displayed at the top. After each test, it was recorded if the document suggested by the COSIM function was indeed the most relevant in the document space. Table 5.2 presents the results of the preliminary experiments.

Out of 26 queries submitted, 87% received relevant documents returned. As it is assumed that the user will already have some knowledge of a content story or news event before

EXPERIMENTAL RESULTS		
Total number of queries	26	
	Overall	Excluding 'Out of Domain'
Relevant results returned	77%	87%
Irrelevant results returned	12%	13%
'Out of domain' returned	11%	N/A

Table 5.2: Preliminary Experiment Results

interacting with the system, out of domain queries can therefore be omitted from the results.

Close analysis of the 13% irrelevant results returned suggested three main shortcomings of the cosine similarity function:

- Higher tws were given to more unique terms within the document space due to smaller values of $|\{t \in d\}|$.
- The document weightings were contaminated by the inclusion of non key word terms such as stop words.
- Due to the matching of literal words, synonyms were not included during the similarity calculations.

To overcome these drawbacks, different tw calculations for performing the COSIM function were investigated to see if they offer any benefit over the benchmark tw calculation for a spoken dialogue system.

To explore the effect of giving the most common terms a higher tw, the Document Frequency (DF) was used, which gives the probability of a document containing term t. Garcia (2006) used a similar calculation, as defined by (4):

$$W_{t,d} = tf_t \cdot \frac{|\{t \in d\}|}{|D|} \qquad (4)$$

Although stop words were removed from the test queries by the Content Spotter during the experiment, this was a rule based filter that identifies specified stop words and removes them from the query. Not based on any syntactical parsing or grammatical understanding, previous results highlight a cause for concern where stop words were not caught by the filter, corrupting the similarity calculations. To overcome this, it was proposed to use only the Named Entities (NEs) to create the document space. The NEs were extracted using a freely available Named Entity Extractor[21], and the calculation is defined in (5):

$$W_{t,d} = tf_n \cdot \frac{|\{n \in d\}|}{|D|} \quad where\ n \in Q \qquad (5)$$

where *n* is a named entity from the query *Q*.

Lastly, to include associating similar words to those used in the user's query, the COSIM function was enhanced with WordNet[22], which provides both synonyms and mathematical differences between words. The associated tw calculation is defined in (6):

$$W_{t,d} = \left[tf_t x \frac{\text{Syns}_T - \text{Syns}_t}{Syns_T}\right] \cdot \log \frac{|D|}{|\{t \in d\}|} \quad where\ T \in Q \qquad (6)$$

[21] http://kmi.open.ac.uk/people/jianhan/ESpotter/

[22] http://wordnet.princeton.edu/

where *Syns* is the mathematical representation of a word given by WordNet, *T* is a key term from Query *Q*, and *t* is a term that is a synonym of *T*.

To investigate the feasibility and performance of each of the tw calculations, experiments were carried out where a query was input to each tw equation and then the returned document marked as relevant or irrelevant. In total there were 3 sets of COSIM experiments:

- Experiment 1 is the normal tw benchmark (3).
- Experiment 2 is the tw based on NE (5).
- Experiment 3 is the tw using WordNet to include synonyms of *Q* (6).

Furthermore, each of the 3 experiments were carried out twice, once with the normal IDF (3) calculation, and once using the DF (4), resulting in 6 experiments in total for each query. The same set of queries was used on the same document space throughout the experiment, and the results are in Table 5.3.

	tw exp. 1		**tw exp. 2**		**tw exp. 3**	
	IDF	**DF**	**IDF**	**DF**	**IDF**	**DF**
Calculation Number	**1**	**2**	**3**	**4**	**5**	**6**
% Relevant returns	96%	78%	59%	59%	81%	40%
% Irrelevant Returns	4%	22%	41%	41%	19%	60%
Average relevant similarity	0.843	0.923	0.983	0.983	0.548	0.943
Average irrelevant similarity	0.749	0.890	0.09	0.09	0.593	0.801

Table 5.3: Enhanced COSIM Experiment Results

The benchmark returned 96% relevancy using the traditional tw calculations. While it was thought that the DF, as opposed to the IDF, would provide a more reliable method for web IR for constructing a document space, it can be seen that this actually dropped the percentage of relevant documents to 78% in experiment 1. This was also repeated in experiment 3, effectively signalling that this method of creating a document space was not appropriate.

Experiment 2 shows that the percentage of relevant returns are the same using both the DF and IDF calculations for tw calculations based on NEs. The average similarity ratings for both relevant and irrelevant documents are also the same. Due to the low number of key terms in use during a query when using only NEs, this led to calculations based on a small document space and therefore no variance at all was observed when using DF or IDF with NEs. Poor performance of the NE recognizer in use might also be part of the reason why the percentage of relevant documents returned was so low.
To analyse the performance of tw experiment 2 against the benchmark, the results were prepared into a scatter graph, presented as Figure 5.5.

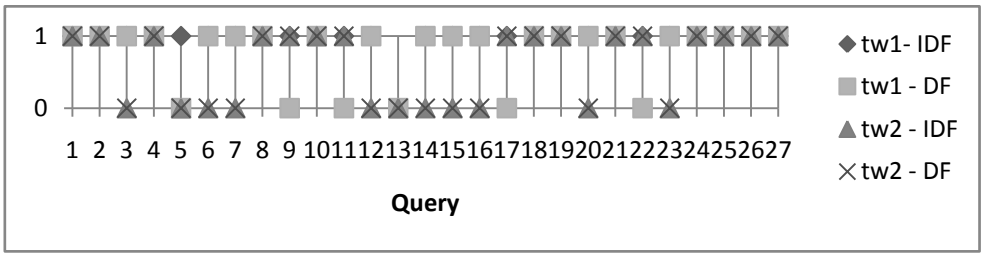


Figure 5.5: Comparison of Calculations tw1 and tw2

In the scatter graph above, the y axis represents a relevant return or irrelevant return, denoted by 1 and 0 respectively. The x axis represents each of the 27 queries used throughout the experiments. Due to the high performance of 96% of the benchmark tw1 using IDF (calculation 1), it can be seen that most diamonds appear at value 1 on the y

axis, apart from Q = 13. As the performance of tw2 using both DF and IDF calculations also return a negative response here, and the relevant responses of tw2 appear where there is already a relevant response of tw1, it can be suggested that using tw2 for term weights calculation does not provide any meaningful gain or benefit over the benchmark.

Experiment 3 shows that, although the percent of relevant documents returned is lower than the benchmark at 81%, analysis of the scatter graph (Figure 5.6) comparing tw experiments 1 and 3 provides justification for the future exploration of this tw method.

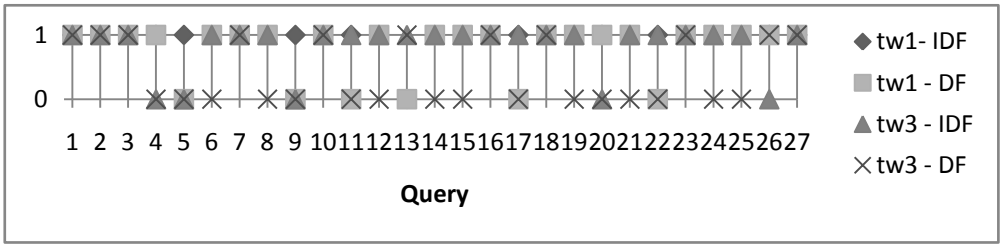


Figure 5.6: Comparison of Calculations tw1 and tw3

Previously highlighted was *Q* = 13, where one of the key terms was the word ‘injured’. The benchmark matched this query to a document containing the word ‘injured’ which was inappropriate as it also contained this key term. However, the tw3 calculation correctly matched ‘injured’ to its synonym ‘wounded’, and returned a more appropriate document.

The scatter graph also shows that the outcomes for tw3-IDF returned similar documents as the benchmark when using the IDF. The high number of ‘X’ symbols on y = 0 shows that the usage of DF to construct the document space with synonyms actually diminishes performance. Due to experiments 1 and 3 having similar performance when using IDF, further investigations should be carried out to advance this method of constructing a document space, as there is some benefit to be gained here as

demonstrated by matching not only the literal word from *Q*. Due to the highest number of relevant document returned by the benchmark calculation (1), it was then decided to proceed with the design of the Content Spotter utilising this calculation.

5.5 Summary

This chapter has introduced the Content Manager, exploring its roles and functionalities within the VoiceBrowse architecture. Expected challenges and issues were considered, and methods to overcome these were studied. It has been shown that, by using RSS and API feeds initially as structures of content querying and extraction, online content can then be downloaded and parsed by the Dialogue Manager in a standard manner. This will resolve shortcoming in current dialogue research, specifically the reliance on purposely crafted domain representations required for dialogue management.

The Content Manager therefore caters for the majority of technical requirements introduced in Chapter 4, and the practical hypothesis to be tested is that, by use of the Content Manager and its operation, one dialogue manager can generically access multi-domain types and structures from online sources. The Dialogue Manager of VoiceBrowse, to be discussed in the following chapter, discusses the usability issues related to the requirements introduced in Chapter 4.

Finally, preliminary testing of the Content Spotter offered initial insight into the expected performance of the similarity function, including the study of enhanced algorithms to improve on the benchmark. However, after careful analysis of the preliminary results, it was decided to proceed with the benchmark, although it was felt that resources such as WordNet do offer some benefit for future dialogue systems.

Chapter 6: VoiceBrowse Dialogue Manager

This chapter will introduce the dialogue management component of VoiceBrowse, including a discussion of the anticipated causes for concern with regard to generic dialogue management. The focus of the research involving the Dialogue Manager is on the usability requirements specified in Chapter 4.

6.1 Dialogue Manager: Introduction

It is important to note that the system will be engaging the user in dialogue to deliver the information requested by them from the Internet - VoiceBrowse is not a narration system that will take web pages or provided content and read it back to the user as in Andersen & Hjulmand (2005). A conversation will be taking place between the user and system to allow the meaningful delivery of content through dialogue. The user will request items of the system, but likewise, the system will request more information of the user to continuously refine the criteria of the content request.

The Dialogue Manager and the components it interacts with within the VoiceBrowse architecture can be seen in Figure 6.1. Of special consideration is the relationship between the Dialogue Manager and the Content Manager, as it is the relationship between the two that enables the generic nature of VoiceBrowse.

It is the role of the Dialogue Manager to accept the inputs from the user, and if the input is a query for information, pass the terms to the Content Manager to perform the similarity function. The Content Manager will then pass its suggested content back to the Dialogue Manager for output, or if the suggested content requires parameter values from the user to

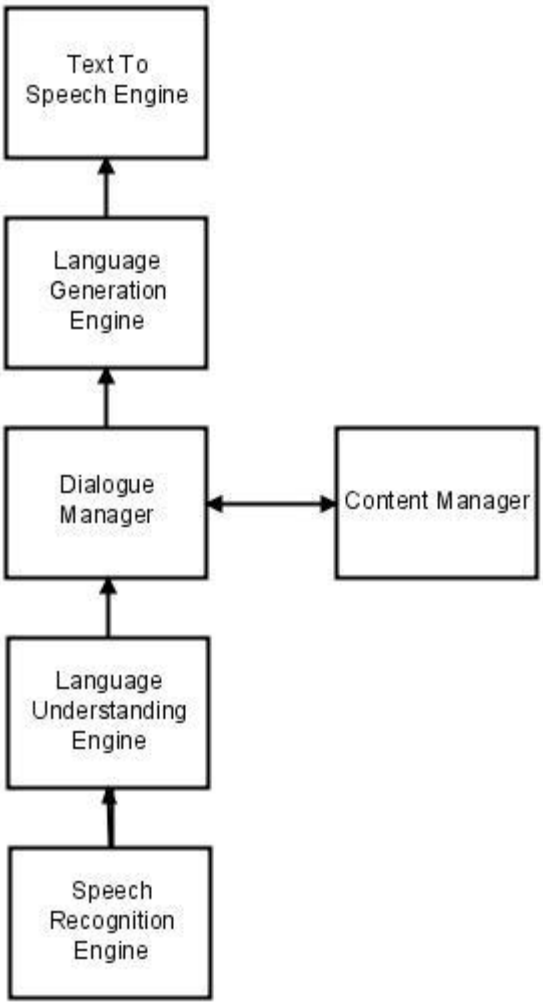


Figure 6.1: Dialogue Manager

proceed, such as flight details, then pass this list of parameters back to the Dialogue Manager for collection from the user. The handling of user input is discussed further in Section 6.4

The Dialogue Manager also controls the output of information to the user, specifically the amount of information output. As discussed in Section 2.8, people can cognitively handle a limited amount of information communicated through voice, and so a primary role of the Dialogue Manager with respect to system outputs is managing the amount of information presented to the user. Consequently mechanisms must also be available to the user to navigate through the resulting content in a standard manner, such as

functions to move back and forward in the document space. The handling of the system outputs to the user is discussed further in Section 6.5.

6.2 Dialogue Manager: Issues and Concerns

Currently, dialogue managers in dynamic dialogue systems are created specifically for one domain knowledge and structure. A dialogue manager created to access various domains from unknown structures and sources has not yet been realised, and current methods of dialogue management are not therefore appropriate. To accommodate interactions in various domains generically, three issues arise:

- The handling of inputs from the user regarding any content type that is available online.
- The interpretation and understanding of the request in a generic manner.
- The output of a meaningful response based on the type of content to the user.

In typical dialogue systems, the input of speech at any stage is usually predictable due to the domain being fixed - utterances can often be handled by the use of keywords, or open ended questions dealt with by examination of human-human conversation or past dialogue with the system. In addition, language understanding is usually an unambiguous task due to the interpretation being made in a specific area. For VoiceBrowse therefore, due to multiple domains being handled, uttered words from the user cannot be anticipated, and the input also has the potential to be ambiguous from the viewpoint of the Dialogue Manager.

System outputs in typical dialogue systems are normally hand crafted and structured specifically for the user guidance and direction at any given dialogue state. Care is

given to the outputs so that the user can know exactly how to respond, and there is no ambiguity as to how the user can understand the request.

Once again, due to VoiceBrowse functioning with multiple domains, it is difficult to structure output prompts both generically and specifically at the same time. An output prompt to collect the departure city for a flight booking for example might read "Please tell me the city from which you wish to fly", whilst for a hotel booking it might read "Please tell me the city in which you wish to stay". These two outputs appear to have some overlap in wording, hinting that generic prompts might be viable, but if the next task to be handled by VoiceBrowse is to purchase an item of Amazon for example, then the output would have to read "What is the item you wish to purchase?" Considering therefore the range of tasks available to be completed online, the wording of system prompts for most stages of the dialogue will be problematic.

Enabling the Dialogue Manager to be generic also has an effect on the initiative that can be used. When the system is created specifically for a single domain, system initiative can be used as the task to be completed can be anticipated, and the required stages of dialogue defined to extract the necessary information from the user. Similar for user initiative, language models and grammar can be created to accommodate a range of natural language inputs based upon a fixed domain. The same dialogue states and language understanding models cannot be easily transferred to alternative domains without intervention from a dialogue designer.

Consider also the reasons discussed previously to use either system or user initiative. System initiative is less flexible, but as the inputs required from the user are relatively small at each stage, and usually from a restricted set of keywords, the possibility of error is low. Dialogues tend to take longer however, so user initiative can be used to

overcome this due to the possibility of natural language inputs. However, with the increased flexibility comes an increased chance of error during the dialogue, as language understanding is now a more complicated process.

Additionally, it would be expected that handling generic inputs and outputs could be catered for with less effort by system initiative than by user initiative – a simplistic version could create a hierarchical structure of content categories, for example, which the Dialogue Manager would lead the user to using step by step questions. Recognising and interpreting open ended queries based on an infinite knowledge set would obviously be more difficult.

A contrast is evident therefore between dialogue usability and management with respect to which initiative is used – system initiative may result in longer, more repetitive dialogues, but may be better suited to offer generic dialogues for online content. User initiative though may offer shorter and more human like conversations, but this leads to an increased complexity with respect to dialogue management and speech input and interpretation. The dialogue initiatives to be studied in VoiceBrowse will be considered in the following section.

6.3 Dialogue Manager: Proposed Dialogue Strategies

Previous research has investigated system-led and user-led dialogue approaches to dialogue (Chu-Carroll 2000). To date, this has been within the domain of task based dialogues, such as booking flights, hotels or tickets using a spoken dialogue system. As yet, no studies have compared the two approaches based on a more information-based type of dialogue, where there is not a task, or series of sub-tasks to be completed, or even a finite set of paths to follow through the system.

In this domain of information-based dialogues, the definition of dialogue initiative changes somewhat, as the parties involved in dialogue are now no longer collaborating with one another to solve a task. Instead, the dialogue evolves more opportunistically, based on the previous utterances and what the user has now learnt from the dialogue. For example, they may be more interested in one news story than another, and therefore would request more information on it. Alternatively, nothing in the current interaction may be of interest, and a complete change of topic may occur, such as to sports stories. As the dialogue evolves according to the user's interests, the user will always have the initiative in the dialogue, requesting information of the system, and then making further requests based on the response.

The traditional classification therefore of dialogue into system or user led initiative is not appropriate for information-based dialogues. Using the foundations of both approaches however, one could classify such dialogues as flexible or inflexible, or an open versus closed approach to dialogue.

An 'open' approach to dialogue can be defined as one where the user is not restricted in how they may say things. Questions and requests can be asked of the dialogue system using natural language, the content of which is required to be in the domains currently monitored by the system. A 'closed' approach is one where the utterances allowed to be spoken by the user are from a finite set of utterances understood by the system, similar to a command and control type of system. The open approach is therefore more flexible and natural than the closed approach.

To study the effect of dialogue initiative on usability with regard to browsing the Internet through voice, it has been proposed to use two different dialogue strategies, namely 'closed' and 'open'. To be a fair comparison, the system should be identical in

every way, and the modular architecture will support this as only the objects implementing the dialogue manager will be changed during the comparison.

To facilitate the development of a less flexible initiative, grammars need to be constructed that will allow the user to speak key phrases and commands that the system can recognise and interpret. This is discussed in the following section, followed by the handling of system outputs in Section 6.5.

6.4 Dialogue Manager: Handling User Inputs

The dialogue manager receives the input string recognised by the ASR and its function is then to decide what action to perform next in the interaction. If the speaker's intent is interpreted as referring to a task-based interaction, then more information may be needed from the user. If it is a request for content, or all required information has been elicited from the user and understood by the Dialogue Manager, then the request will be passed to the Content Manager to retrieve the information from the Internet.

If the closed strategy is in use, then the input should be a topic or provider with which a feed can be uniquely identified. It is proposed to use the <title> element of the RSS feeds, held as <feedTitle> in the document space, to create a grammar that will allow the input of either a provider name or news category. This also provides a means of matching the input directly onto the document space - illustrated in Figure 6.2. This will allow the user to utter such phrases as "BBC News", or "Football Headlines".

The Content Manager will then return only those documents in the document space whose feed title matches the query. If the query is too broad at this stage, the Content Manager will indicate this to the Dialogue Manager which will then output the possible

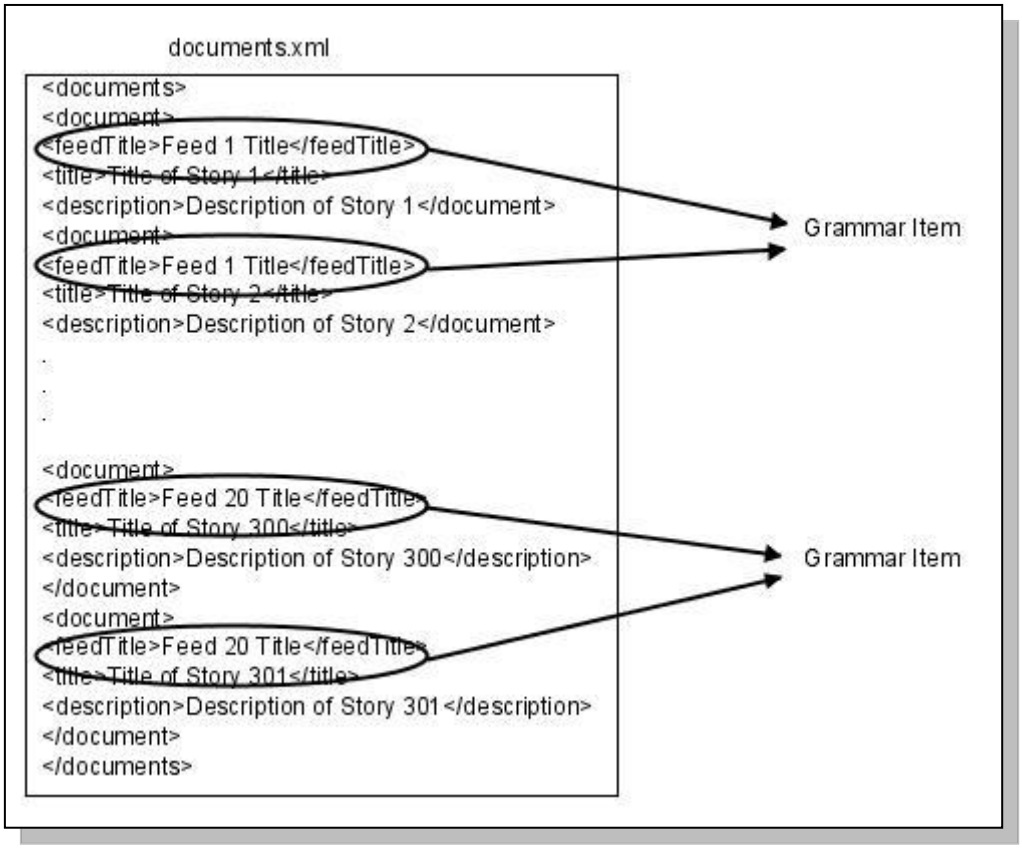


Figure 6.2: Grammar Creation In The Dialogue Manager

feeds to the user, and ask for a more specific term - for example “You said news. Do you mean sports or entertainment news?”.

If the open dialogue strategy is in use, the Content Manager creates a document space of stories from the different <description> elements from the RSS feeds in the VoiceBrowse environment. This document space will be used instead of the <feed> elements to create a grammar that will allow questions based on its contents. The open ended question recognised from the user is then input to the Content Spotter, which uses a Cosine Similarity function to return a ranked list of weighted documents based on their similarity to the current query. The query is first prepared by removing specified stop words and converting to lower case, to create a query form similar to that of the documents. The topmost related document is then output to the user, followed by the next if requested, and so on.

Independent of dialogue strategy, while the short <description> elements are being output, the user can ask for more information for any story by stating the story's number in the list. The next step is then to fetch the main story or content body from the provider's website.

For task based dialogues, the Content Manager identities the API to handle the current task. With each API that is included in the environment, specified parameters are used to create a dialogue that will allow the user to elicit values for these required parameters. Once each required parameter has a value, the attribute-value pairs can be packaged into an HTTP request and sent to the relevant provider.

6.5 Dialogue Manager: Handling System Outputs

In a traditional dialogue system developed for one specific task, or those developed for a static domain, outputs are easily anticipated and catered for through the use of templates at each dialogue state. However, as VoiceBrowse is both multi-domain and dynamic, the output will change at each state and for each user.

Furthermore the output will also be dependent upon content - content requested by the user is required to be delivered in a meaningful way that can be easily understood by the user. This preparation of content is complicated due to the wide range of different types of content that the user could request - content can come in many different shapes and forms that must be handled by the system. Each content type will present its own requirements to the system**,** when being delivered to the user. For example, a news article will be too long to be delivered as a whole article, and so will need to be summarised before being delivered. Likewise, if a user searched for flights, VoiceBrowse would have to output a possible long list of times and dates to the user – a

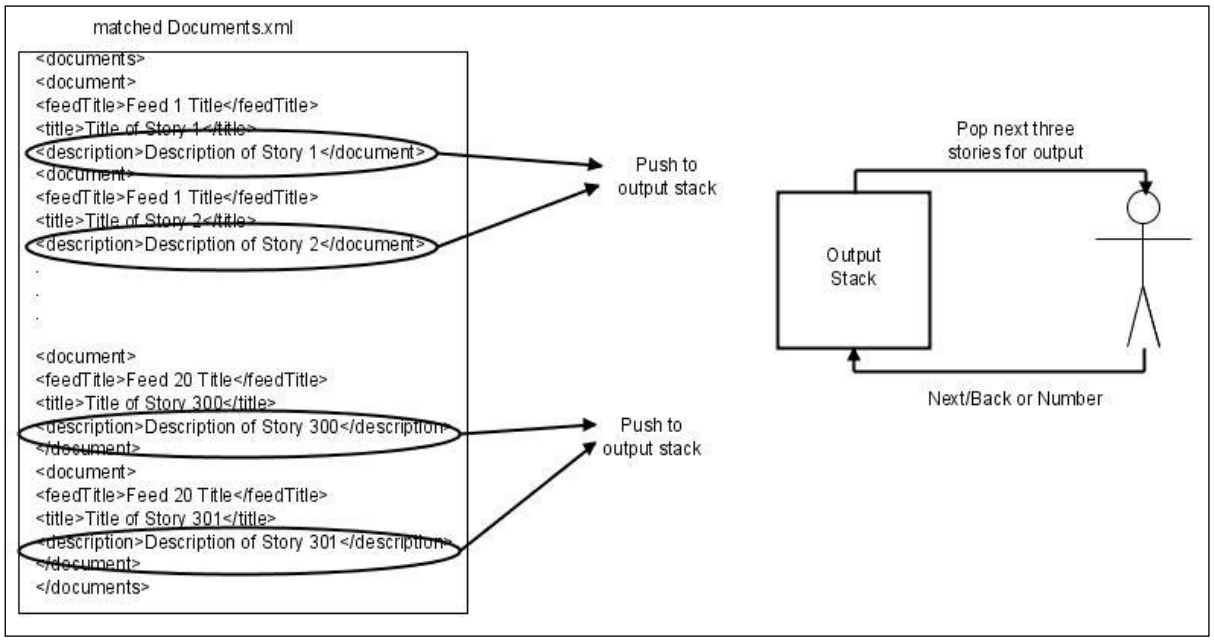


Figure 6.3: Outputting Content To The User

content item that may be best summarised by grouping flights together based on some criterion – for example "There are 5 flights in the morning".

To understand how it is possible to handle the system outputs, it is first necessary to understand which content is available to the Dialogue Manager and how it is represented. The output of the Dialogue Manager is provided by the output of the Content Manager - an XML file containing either a list of <documents> from the Content Spotter or a list of results from the API response. If the open-ended approach is in use, the XML file is a ranked list of related documents. The content of the <description> element of the topmost related document is output to the user by inserting the text into a VoiceXML <prompt>, followed by the next story if requested, and so on. With the system-directed strategy, it is simply a linear list of documents output in similar fashion, illustrated in Figure 6.3.

The output from the Content Manager in either case is then a structured XML document of results – each result containing the <description> element, and also the <source> of

the full story. The Dialogue Manager will output each result in a manageable amount, such as three at a time to the user. If the user requests more information about a particular story, the Dialogue Manager can then send the <source> of that story back to the Content Manager, which proceeds to access the main body of content from the story's web page (see Chapter 5). This web page is then output to the user, also in a manageable amount such as three <p> tags at a time.

With the API response, VoiceBrowse extracts the results from the XML response and presents these to the user five at a time. Future work in this respect will allow the user a finer grain of control over the presentation of results, such as sorting and filtering, discussed in Chapter 9.

6.6 Summary

Generic dialogue management in the capacity of online browsing poses many challenges and concerns that must be overcome: User inputs cannot generally be anticipated; language interpretation has the potential to be ambiguous; and system outputs must be worded both non-specifically yet still be effective in eliciting the expected utterance from the user.

The cooperation of the Dialogue Manager with the Content Manager will be pivotal in overcoming these issues and realising the generic nature of VoiceBrowse. The entire workflow of VoiceBrowse, which is based on RSS and API feeds, can now be amalgamated as is shown in Figure 6.4.

The workflow shows that the process starts with the RSS and API feeds, which make up the core of the system, providing a standard and structured means to cater for dialogue aspects, including grammar creation and language interpretation. By using appropriate

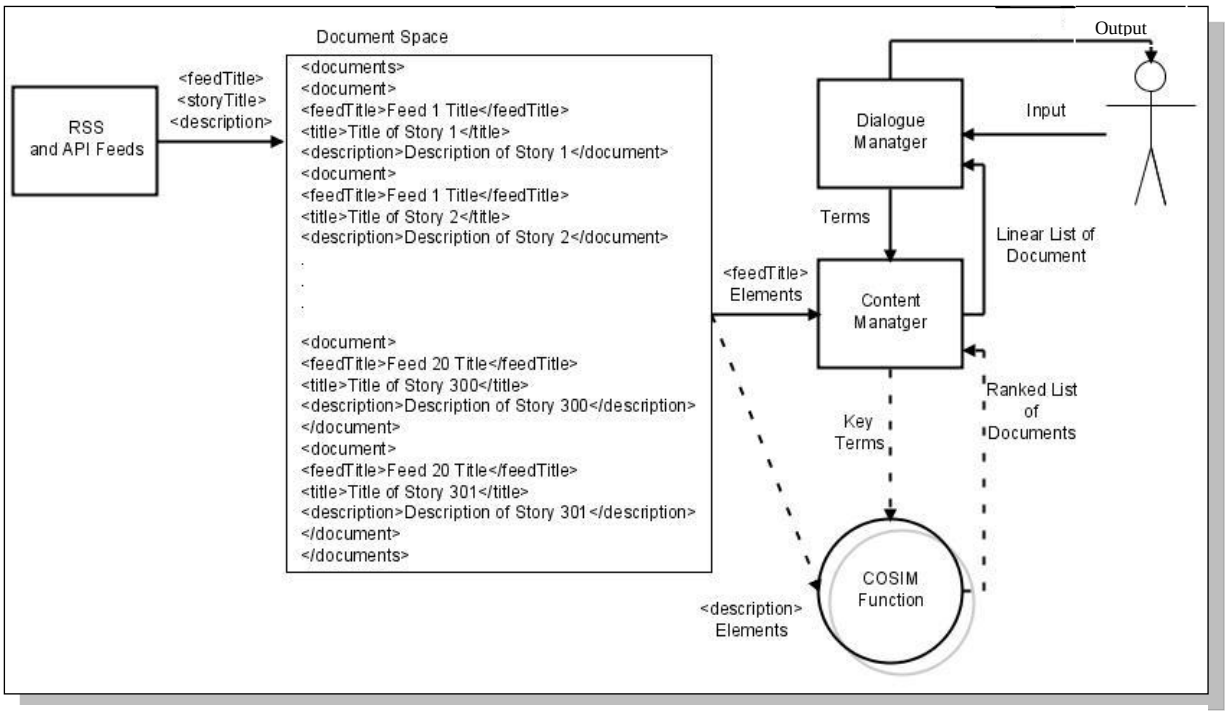


Figure 6.4: VoiceBrowse Workflow

procedural code, a document space of documents and associated titles can be created. An input from the user is received by the Dialogue Manager, which has used this document space to create the grammar required to recognise and understand the user's utterance. The execution from this point then depends on the dialogue strategy in use.

If the closed dialogue version is being used, then the Dialogue Manager will match the user's utterance to the grammar created from the Feed Titles. This will allow the user to choose content, either from a specific provider, such as 'BBC' or 'CNN', or by category, such as 'Sports' or 'World' News. This is possible as Feed Titles provide both the provider and category of news that the feed is related to – an example would be 'BBC Football News'.

The Content Manager can then use the document space to match the documents with the same <feedTitle> element as the user's utterance. A linear list of results is then output back to the user in a manageable amount, for example three <description> elements at a time.

However, if the open version of VoiceBrowse is in use, then the input from the user is an open ended question, with the words from the <description> elements in the document space being used as an aid to improve recognition of the ASR, as it is highly probably that these words will form the basis of queries uttered by the user. Once recognised, the string is passed to the Content Manager, which performs the COSINE Similarity function on the document space, using the user's utterance as its input. Output to the user then proceeds as in the closed approach, although the list of documents to be output is now in a ranked order.

This concludes the design phase of the research: the literature has been explored and shortcomings and limitations of current research noted; the inadequacies identified relating to dynamic dialogue systems were noted, and requirements created to fulfil these failings with regard to the browsing of online content through dialogue; and an architecture has been developed that will realise the listed requirements, and its components have been surveyed in depth. The next chapter introduces the implementation phase in which the conceptual designs are transformed into a realisation of the VoiceBrowse system.

Chapter 7: VoiceBrowse Implementation

Once the VoiceBrowse architecture had been conceptually developed, and research questions to be addressed concisely noted, current dialogue technologies and implementations were explored to gain knowledge of the available tools to realise VoiceBrowse. During this stage, various languages, reusable components and open source architectures were studied, and each of their advantages and disadvantages noted with regarding to implementing a system like VoiceBrowse – the most relevant of these are introduced in Section 7.1. A discussion of the chosen technology of VoiceXML follows including the work taken to convert VoiceBrowse from a theoretical architecture to relevant VoiceXML Call Flow Diagrams (Section 7.2) and associated prompt designs (Section 7.3). In the remainder of the chapter, implementing VoiceBrowse will be discussed in Sections 7.4 - 7.8, and the challenges and issues encountered are discussed in Section 7.9. Example dialogues with VoiceBrowse after the implementation are included as Section 7.10.

7.1 Current Dialogue Technologies

In order to realise the VoiceBrowse architecture, a number of available tools and options for each of the main components of a spoken dialogue system were investigated, from speech recognition engines through to Text-To-Speech technologies. In addition, supporting technologies, such as programming and scripting languages, and database management systems and associated query languages were also explored.

Automatic Speech Recognition (ASR) is available in many open source as well as proprietary implementations. A widely used open source recogniser is the Hidden Markov Model Toolkit (HTK)[23]– a C implemented speech recogniser. A Java alternative is Sphinx 4[24], the latest version of the widely used CMU Sphinx Recogniser[25], which is a commonly used ASR in research projects. Developed and maintained by Carnegie-Mellon University, it offers a high degree of portability and flexibility due to its implementation in the Java programming language. It is a speaker independent ASR that can be updated with language models produced by its associated program SimpleLM[26]. A recent evolution of Sphinx has seen the development of PocketSphinx[27], a more lightweight ASR to permit effective speech recognition on mobile devices.

However, it is common that open source programs are not compiled or readily available in an executable format, and so open source ASRs, including Sphinx, often have complicated and complex protocols that are required for installation and operation to mimic the development environment. To overcome these problems, proprietary solutions have been developed that are available to be used 'out of the box' – minimum setup and installation is required, and they can be used in similar fashion to other programming languages. A popular proprietary option is the Microsoft Speech

---

[23] http://htk.eng.cam.ac.uk/

[24] http://cmusphinx.sourceforge.net/sphinx4/

[25] http://cmusphinx.sourceforge.net/html/cmusphinx.php

[26] http://cmusphinx.sourceforge.net/html/download.php#SimpleLM

[27] http://www.speech.cs.cmu.edu/pocketsphinx/

Recogniser[28], included as a component of the Windows XP and Vista Operating Systems. Primarily used as the ASR for speech based interfaces to the operating systems, it is also possible to use the speech recogniser programmatically with use of the SAPI, or Speech Application Programming Interface[29]. This has the benefit of integrating with speech applications seamlessly, with no additional prerequisites or components required. Other proprietary options are available, such as IBM's ViaVoice[30], or Nuance's Dragon Naturally Speaking[31]. However, due to their commercial development, proprietary ASRs tend to offer limited scope as to the extent to which they can be tailored, and most are available only as 'out of the box' recognisers.

The same classification can also be used with regard to output technologies – Text-To-Speech (TTS) engines are available as both open source and proprietary packages. Most notable with regard to open source TTS engines is Festival TTS[32], which has appeared as part of many dialogue systems in research (Bohus & Rudnicky 2005a; Hanna et al. 2005). Originally written in C++, APIs are now available for Festival to enable its implementation on various other platforms. A native Java alternative also worth a mention is FreeTTS[33], another example of another popular TTS engine.

Like their freely available ASR counterparts, the benefits that a developer gains in flexibility and customisation with using an open source TTS have to be balanced against

---

[28] http://www.microsoft.com/speech/

[29] http://www.microsoft.com/downloads/details.aspx?FamilyID=5e86ec97-40a7-453f-b0ee-6583171b4530&DisplayLang=en

[30] http://www-01.ibm.com/software/pervasive/embedded_viavoice/

[31] http://www.nuance.co.uk/naturallyspeaking/

[32] http://www.cstr.ed.ac.uk/projects/festival/

[33] http://freetts.sourceforge.net/docs/index.php

the benefits of the simplicity and ease of an implementation that a proprietary product can offer. Cepstral[34] is an example of a commercial TTS engine that can be installed and utilised on a particular system within minutes by running the appropriate executable file. Others include Loquendo TTS[35] and AT&T Natural Voices[36].

Many dialogue architectures are developed in a modular paradigm, allowing the substitution of different implementations for each component - the aforementioned system by Bohus & Rudnicky (2005) is a typical example of such a system.

An alternative to incorporating together many different components of a dialogue system to form a complete architecture is to utilise the VoiceXML standard, for which there are many platforms available for implementation. Some VoiceXML platforms consist not only of the required VoiceXML Interpreter and associated technologies, but also of ASR and TTS engines to offer a holistic solution for a spoken dialogue system[37+38]. Such platforms can provide developers with a solution that offers reduced system deployment time due to the inclusion of all the required components.

Web based VoiceXML platforms are also available[39+40], where developers can uploaded VoiceXML to a remote server – interactions can either then be text based, or speech enabled by using Voice Over IP telephony software, such as Skype[41]. This approach offers the additional benefit of not requiring any specialist hardware or software to run

---

[34] http://cepstral.com/

[35] http://www.loquendo.com/en/technology/TTS.htm

[36] http://www.naturalvoices.att.com/

[37] http://www.voxeo.com/products/voicexml-ivr-platform.jsp

[38] http://www-01.ibm.com/software/pervasive/voice_toolkit/

[39] http://bevocal.com

[40] https://studio.tellme.com/

[41] http://www.skype.com

the VoiceXML platform, although such web based platforms usually offer a limited form of functionality with regard to what can be achieved through VoiceXML – lack of scripting language and database support for example. Such scripting and database technologies are often used to complement VoiceXML solutions to realise a 'dynamic' system. Similar to web programming techniques, this allows the amalgamation of a spoken dialogue system with a Database Management System, providing real time information to end users of the system.

7.2 VoiceBrowse Implementation: VoiceXML and Call Flow Diagrams

Following an exploration of a range of current dialogue technologies and associated technologies, it was felt that a VoiceXML implementation of VoiceBrowse was preferable for several reasons:

- It offers compliance with W3C standards, promoting a current standard specification for dialogue systems and maintenance sustainability, and allowing future development and collaboration with external parties.
- The availability of large amounts of literature helps to resolve issues and problems encountered with the VoiceXML specification.
- A VoiceXML system can be implemented on a range of devices in a variety of formats, for example on a standalone system on a personal computer, a telephone based system accessed remotely, or the possibility of installation on small form computers, such as mobile devices, including PDAs.
- By choosing an appropriate VoiceXML platform, required prerequisite components for a dialogue system, such as a speech recognition and TTS engine etc., will be integrated into the platform.

- VoiceXML platforms are themselves based on web technology paradigms, a solution that will support the technologies involved in the delivery of online content through dialogue.

As stated previously, there are various VoiceXML platforms to choose from, including commercial solutions, freely available solutions and web based solutions – each of which can offer benefits over other solutions. The choice of platform is therefore often dependent on the required requirements of the system, which for VoiceBrowse can be summarised as:

- Ability to call the system over the phone.
- Ability to create dynamic scripts using an appropriate scripting language.
- For economic reasons, the platform should not be expensive to obtain or use.
- Ability to record calls to the system for later analysis.
- Ability to install the platform on the Windows XP Operating System.

Considering the above requirements, it was decided to proceed with Voxeo Prophecy which satisfies all the above points. Other solutions were discounted, as they could either not easily be set up for external telephone calls to the system without additional hardware and software (IBM Voice Tool Kit), or that they were web based, and therefore unable to run executables and scripting languages due to security issues - for example, BeVocal.

The Voxeo Platform itself can be freely downloaded from the Voxeo website, and installation is a simple matter of running the executable file. Once installed, calls to the

platform can either be through a locally installed SIP phone, or externally through Voice Over IP (VOIP), for example with the use of Skype[42]. The platform also comes with its own web server which can execute PHP scripts, but due to the writer having more experience with .Net technologies for web programming; it was felt that the use of ASP.NET[43] would better facilitate the development.

A requirement for the execution of ASP.NET scripts was a compatible web server, where the ASP scripts would reside and be accessible to the Prophecy platform during runtime. Furthermore, as ASP.NET itself is based on .NET technology, a native Windows technology, an obvious choice of web server was the bundled Windows Internet Information Service (IIS), provided with most Windows packages. To complete the development environment, an ASP.NET editor was preferred to a simple text editor, due to the associated syntactical debugging and file management facilities included with most professional scripting editors. With the choice of ASP.NET and IIS already made, a Microsoft Integrated Development Environment (IDE) was thought best and Microsoft Visual Web Developer 2005 Edition[44] was chosen for its suitability and ease of use. In addition, the package also came with Microsoft SQL Server 2005 Express Edition[45] to cater for any data storage needs that may arise.

The VoiceBrowse environment is shown graphically in Figure 7.1. During an interaction, the workflow is as follows: the user initiates a call to VoiceBrowse, either through a telephone or smart mobile device, or embedded microphone in a pervasive

---

[42] http://www.skype.com

[43] http://www.asp.net/

[44] http://www.microsoft.com/express/2005/download/default.aspx

[45] http://www.microsoft.com/sql/editions/express/default.mspx

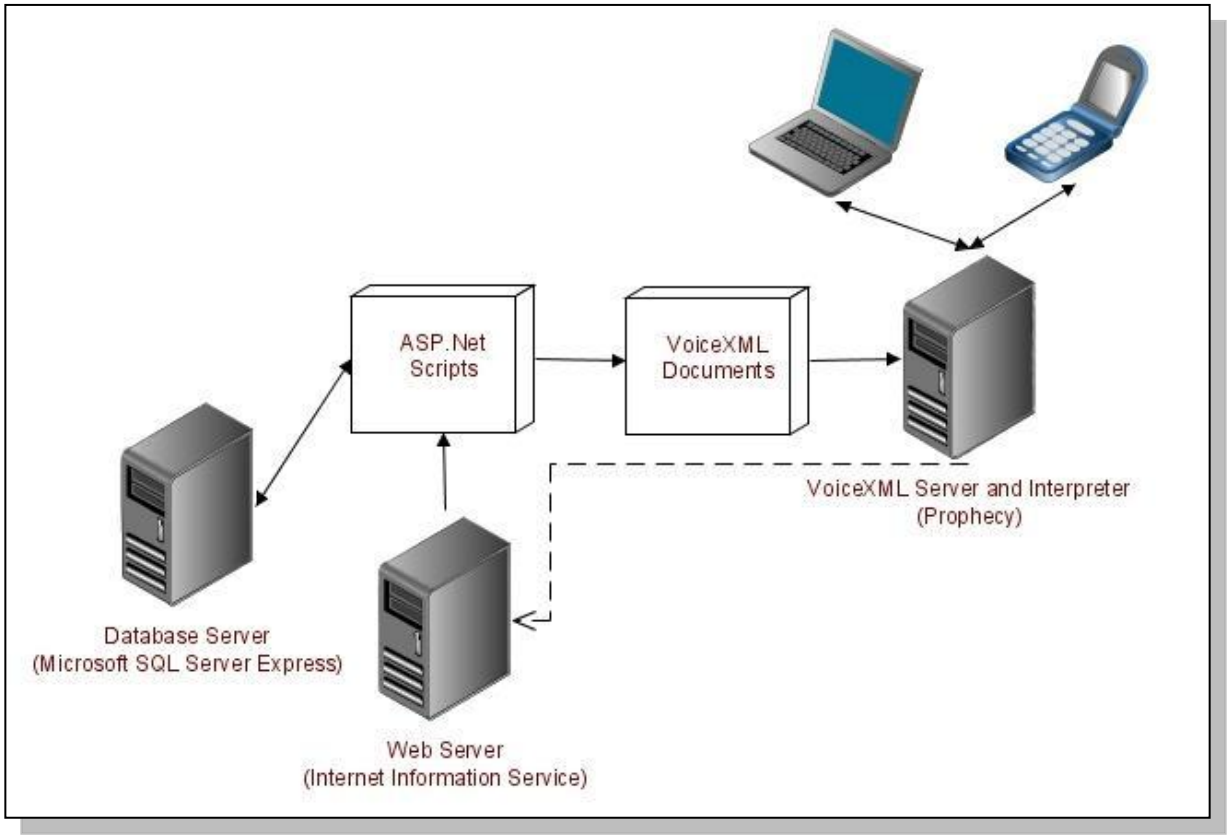


Figure 7.1: VoiceBrowse Environment

environment. This call and all calls to the VoiceXML server are passed to the Web Server (IIS), which controls the flow of the dialogue, and generates the VoiceXML dynamically at runtime through ASP.NET scripts. The ASP.NET scripts also have access to a database server (SQL Server) for any data storage and retrieval needs. Once generated, the VoiceXML is interpreted by the Prophecy server, and output information is relayed back to the user, and the dialogue enters the next stage of the interaction.

With each element of the development environment installed and verified as working, the first step of the process was to create the specified files from the call flow diagrams. In addition, to contain generic functions and logic that would be available to the VoiceXML files throughout the interaction, dialogueManager.aspx and contentManager.aspx were also created. This allowed the reuse of code across various files, maintaining a coherent and consistent implementation.

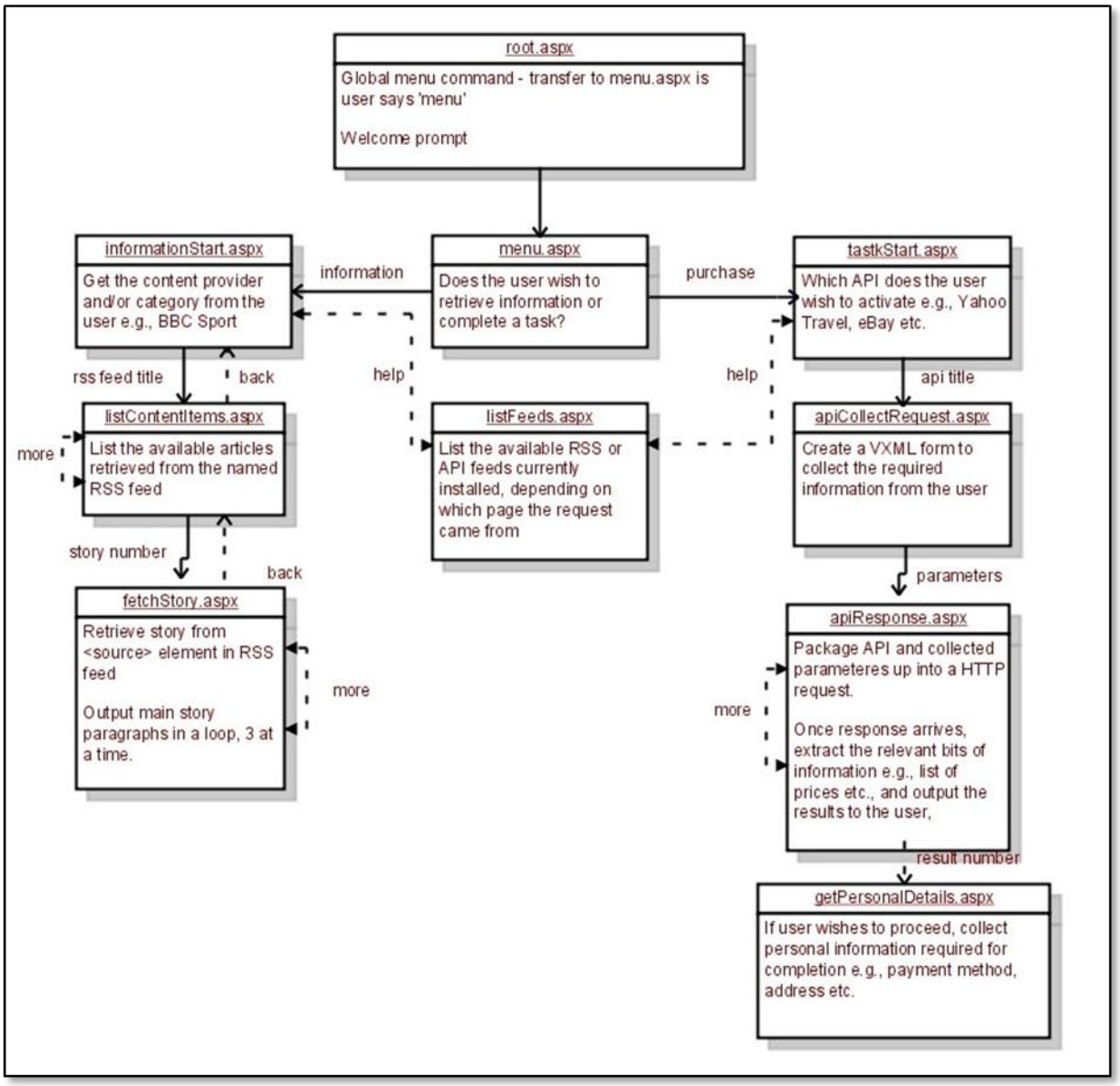


Figure 7.2: VoiceBrowse Call Flow Diagram

Having decided on the implementation platform for VoiceBrowse, the next stage of development was to transform the conceptual architecture of VoiceBrowse into a relevant form of system design diagram, such as an object class hierarchy. VoiceXML has an associated design diagram known as Call Flow Diagrams, showing the interaction paths and transitions through a system. Figure 7.2 shows the initial Call Flow Diagram produced for VoiceBrowse, transformed from the architecture presented in Figure 4.6.

A common initial file in any VoiceXML or dialogue system is the welcome prompt, designed to convey the system's functionality to the user, and communicate the available help functions available throughout the interaction. In a VoiceXML system this is often known as the 'root' document, and so becomes root.aspx in the VoiceBrowse environment.

Given the two different types of dialogue that VoiceBrowse aims to accommodate, narrative and task based, it was initially decided to handle the design of the flow for each dialogue type separately. Menu.aspx is the first time that a user is expected to speak to VoiceBrowse, and it appears sequentially after the welcome prompt. Here the user is expected to say if he/she wishes to complete a task or engage in a narrative dialogue. It is good dialogue usability practice to provide help at each stage in the dialogue, and so an advanced form of help was made available at this initial stage. By requesting help, listFeeds.aspx was available to list the content currently available in VoiceBrowse, for both narrative and task based dialogues.

Depending on the user's utterance, the dialogue then transitions to one of two different paths: one to handle narrative dialogues (informationStart.aspx); or one to handle task based dialogues (taskStart.aspx). informationStart.aspx prompts the user to either state the content provider or category they wish to access, or it allows the user to ask an open ended question, depending on which implementation of VoiceBrowse is active. The next step in the dialogue flow is to output the results from the search to the user, and if the user requests more information regarding a particular item of content, VoiceBrowse should access this from the main web page and output this to the user. listContentItems.aspx and fetchStory.aspx provide this functionality respectively.

Alternatively, if the user had requested to complete a task, then taskStart.aspx retrieves from the user the particular API that they wish to engage in dialogue with. Based on this response, apiCollectRequest,aspx creates a VoiceXML form to collect the required parameters of the API, before it is submitted to the relevant server. Once the API response has been received back from the vendor, apiResponse.aspx outputs the results to the user. If the user wishes to proceed with the task, getPersonalDetails.aspx collects the required details to continue with the task.

With the initial call flow diagram produced, the next step of implementation was to create the initial prompts of the system. Prompt design is an important step of any dialogue development, and can assist the developer in creating the final call flow of the system, as often the prompt design will highlight changes in the call flow that are necessary to accommodate a higher degree of usability. The prompt design for VoiceBrowse is discussed in detail in the following section, but this did lead to a revised version of the initial call flow diagram, which is presented as Figure 7.3.

The major amendment to the initial call flow diagram is the absence of menu.aspx, removing the differentiation of a narrative or a task based dialogue from the user's viewpoint. This is in response to the prompt design for menu.aspx, and the problematic wording of the menu prompt requesting the user to make a distinction between the two different types of dialogue. The wording was problematic for two reasons: that appropriate, concise words suitable for a dialogue prompt could not be determined to convey the different meaning of the two different types of dialogue – for example asking the user to decide between task and information would not be appropriate, as seeking information regarding flight times could be thought of as an information-based

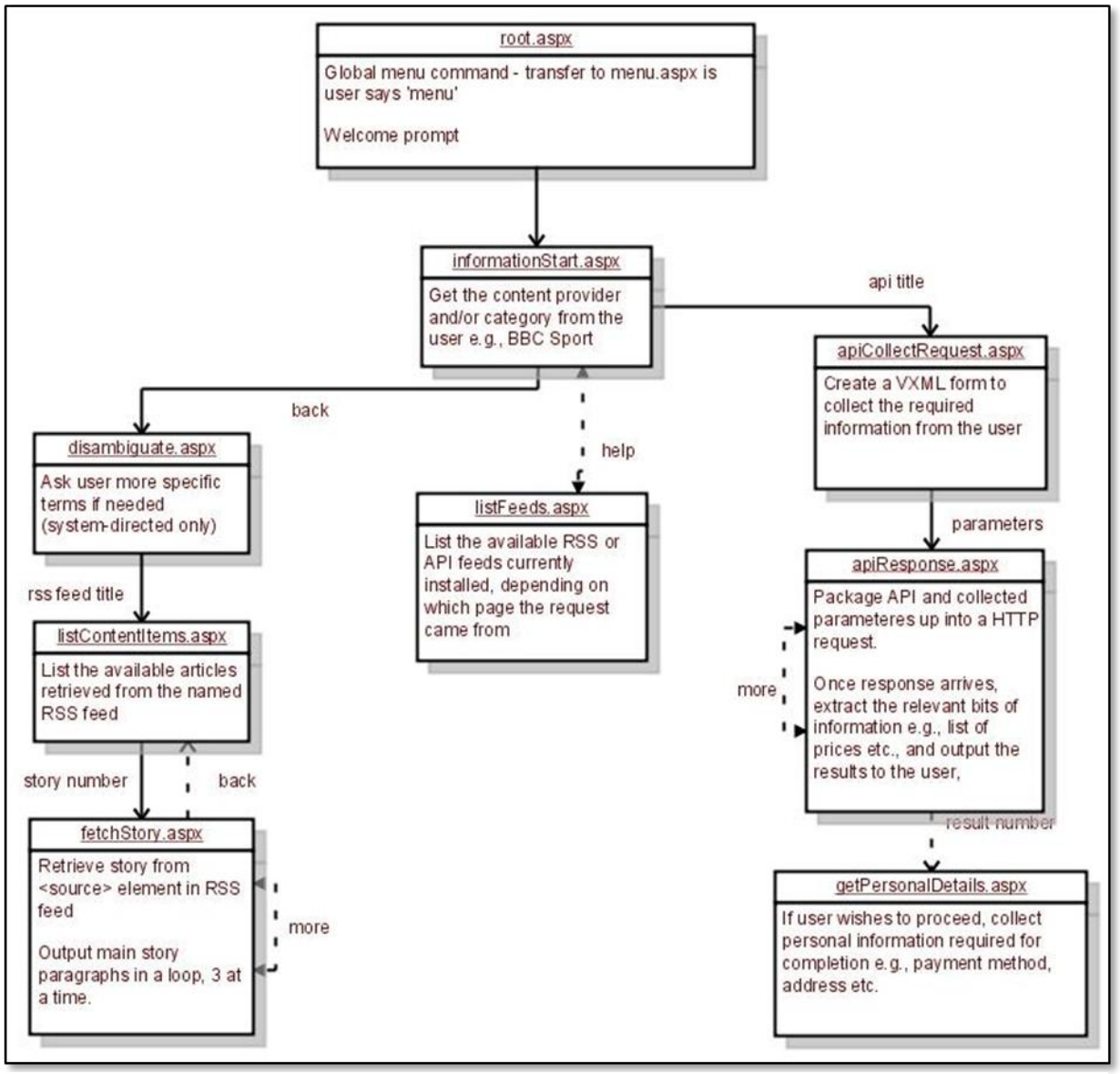


Figure 7.3: VoiceBrowse Revised Call Flow Diagram

dialogue, but in reality it is a task based dialogue; and that end users would not understand the division between the two types of dialogue and the necessity for it.

To overcome this issue, the call flow diagram was revised as above with the removal of menu.aspx, and a programmatical solution was sought to classify the user's request as either information or task based, removing the onus from the user to distinguish between the two different types of dialogue handled by VoiceBrowse. The prompt for informationStart.aspx was generic, allowing the user to request either narrative or task based information at this point, such as the news or to purchase a cinema ticket, without

having to explicitly state beforehand if the dialogue is to be information or task based. Depending on the classification of the utterance by the VoiceBrowse program, the dialogue then makes the transition to the two separate dialogues paths as before.

However, in addition, the call flow diagram was further revised by the inclusion of a new file, disambiguate.aspx. In response to the prompt design, it was felt that there would be the possibility of a user under-specifying a request for information at the initial informationStart.aspx state. The user might say 'news', for example, which could be considered a generic term, and consequently the wrong sub-type of news would be returned to the user by VoiceBrowse. Disambiguate.aspx therefore verifies that a user's utterance will return a unique content set from the RSS feeds, and if not, prompts the user for more specific information, e.g., to refine news further as 'sports' or entertainment' news. With the call flow diagram now fixed, and the associated prompt designs resolved also (see Section 7.3), VoiceBrowse was then implemented using the chosen technologies and platform, discussed from Section 7.4 onwards.

7.3 VoiceBrowse Implementation: Prompt Design

The prompt designs are key to the usability of a dialogue system, as the prompts are the 'front end' of the system that the user will interact with. Similar to judging a graphical interface on the aesthetical appeal, it is often the content of the system prompts that will provide the basis for the user's judgement of the system. Furthermore, prompts can assist the language understanding of a system by constraining the user input to the allowable utterances of the language understanding engine at the particular dialogue state.

For example, consider the first state of VoiceBrowse where user input is allowed. Presented above in the call flow design, it is the function of the informationStart.aspx

state to convey to the user the function of VoiceBrowse, the available options for retrieving information, and the allowable inputs at this starting stage. If this is to be implemented in the closed version of VoiceBrowse, then the allowable inputs are a small set of key words, dynamically created from the titles of the current feeds in the environment. Taking into account that the allowable tasks, content sources and types are virtually unlimited; creating a generic prompt to communicate this to the user can be problematic. If VoiceBrowse were created solely for a cinema booking domain, the first prompt could arguably be "What cinema would you like information for?" However, considering that VoiceBrowse hypothetically in the next interaction could be requested to retrieve flight information, this prompt specific to cinema bookings is now not appropriate.

To make the prompt generic, it therefore must not only be task or content specific, but still express to the user that the system is expecting the name of a cinema, flight company or another appropriate vendor or content provider, such as CNN or BBC for example. The words 'vendor' or 'feed' may not be known by the end users who will use the system, so more appropriate words are needed. Similar considerations are given to the prompts throughout the dialogue states, and a detailed description of the prompt design and their rationale are included as Table 7.1 overleaf.

#	Prompt	VoiceXML Document	Expected Response and Comments
1	Welcome to VoiceBrowse! To return to this point. Say main menu at any time.	root.aspx	No response. A welcome prompt and introduces the 'main menu' keyword.
2	Say help at anytime for assistance. Please say the provider and also category if you wish, or if you want to know what is available, say list!	informationStart.aspx	This is the starting point of an interaction in VoiceBrowse. The user should utter the name of a provider, for example BBC or eBay, or the category of information they desire, such as 'Football News' or 'World News'. Alternatively, the user can say help for more information of what is required at this stage, or additionally they can hear what information is available simply by saying list.
3	Please say the name of a provider, such as BBC or Yahoo. You can also say a specific category, such as News or Sport. If you do not know what providers and categories are available, say list.	informationStart.aspx	This prompt will execute if the user says the keyword 'help' during the previous prompt (#2)

4	Possible choices are: Feed 1, BBC Sport News. Feed 2, BBC Business News. Feed 3, BBC Headline News. Please say repeat, or more. If you have finished with the help, say done.	listFeeds.aspx	If the user says the keyword 'list' in prompt 2, they are taken to this prompt. The purpose of this prompt is to make known to the user what 'feeds' are currently available to the system three at a time. Expected response is 'repeat' to hear the three items again, or 'more' to hear the next 3 items. They can also say 'done' to exit this prompt and return to prompt 2.
5	You have requested VALUE. I have found X different feeds. Options include: X1, X2, ..., XN. Please say the specific topic or provider.	disambiguate.aspx	If the value the user elicited at prompt 2 has identified >1 feed, this prompt makes the user aware of this. The prompt then gives the opportunity to provide some more specific criteria. For example, if the user said 'BBC News' there might be 6 different feeds matching BBC News. The prompt will utter this to the user, followed by the title of the different feeds, such as 'BBC Sport News', 'BBC Entertainment News', 'BBC World News' etc.

6	I haven't found any stories matching your query, let's start again.	disambiguate.aspx	This prompt will fire if no feeds are found matching the user's utterance from prompt 2. Execution will then return to informationStart.aspx.
7	OK, I have found FEED TITLE. There are X stories, and I will now read them out 3 at a time. Say the story number to access the full story, or say repeat, next or back. Story 1.....Story 2.....Story 3.....	listContentItems.aspx	Once a specific feed has been identified, stories from that feed are listed 3 at a time. Once again, the user can say back, next or repeat in a similar fashion to prompt 4. This will either: go back to the previous stage of the dialogue; list the next 3 stories from the current feed; or repeat the current 3 stories uttered to the user. Alternatively, if the user desires more information regarding a particular story, they can utter the associated story number.
8	OK, you want story X. One moment.	listContentItems.aspx	A short prompt to indicate to the user that the system has captured and understood the story number elicited by the user in prompt 7.

9	PARAGRAPH 1. PARAGRAPH 2. PARAGRAPH 3. Please say repeat, next, back, or main menu.	fetchMainContent.aspx	The purpose of this prompt is to output the requested story fetched from the online source three paragraphs at a time. The prompt begins and ends by reminding the user of the keywords that they can say to move on with the dialogue. Similar to previous prompts (#7) the user can say repeat next or back. Main menu is also active still, and provides the opportunity for the user to start a new dialogue with VoiceBrowse by returning them to the main menu (prompt #2)
11	Please tell me the X	apiCollectRequest.aspx	Once a specific API has been identified, the system asks the user for each of the required parameters for its operation. Worded to ensure the generic capture of parameters.
12	Did you say X?	apiCollectRequest.aspx	After the system has collected the parameter, its value will be confirmed. The expected response here is ‘yes’ or ‘no’.

14	Ok. One second.	apiCollectRequest.aspx	Confirmatory message output to the user to acknowledge that all required parameter values have now been collected and confirmed by the system.
15	There are X results. Result 1, X1. Result 2, X2. Result 3, X3. You can say repeat, next, or back at anytime. ,	apiResponse.aspx	The purpose of this prompt is to utter the API results to the user, three results at a time. The prompt begins by reminding the user of the keywords that they can say to move through the result set. Similar to previous prompts (#7), the user can say back, next or repeat in a similar fashion to prompt 4. This will either: go back to the previous stage of the dialogue; list the next 3 results; or repeat the current 3 results uttered to the user. Alternatively, if the user desires more information regarding a particular story, they can utter the associated story number.

Table 7.1: VoiceBrowse Prompt Design

7.4 VoiceBrowse Implementation: Content Manager

Implementation was done in four phases, described in Sections 7.4 to 7.8 - functions for the Content Manager and Dialogue Manager were developed first of all, followed by the VoiceXML Scripts to handle the request and outputting of information based dialogues, and finally the handling of task based dialogues.

The Content Manager makes available to the system throughout the interaction functions related to the management of online information, such as making the content from the RSS and API feeds available, retrieving the main body of content from a source web page, and creating the document space to be used during the Cosine Similarity function.

Paramount to the operation of the Content Manager and the functionality of VoiceBrowse is the mechanism for the inclusion of RSS and API feeds within the VoiceBrowse environment. As RSS feeds are available by accessing a source URL, the address of the URL is therefore required to be made available to VoiceBrowse. The RSS feed at this address can then be accessed by parsing the XML at the URL's location, which is standard. A standard XML specification representing an RSS and its URL was devised for VoiceBrowse, so that the Content Manager can access different RSS feeds in the same way. The XML specification to represent a RSS feed in VoiceBrowse is given in Code Excerpt 1:

The XML shown above is a hierarchical feed representation, specifying first of all that the XML contains a 'feed' plug in for use in VoiceBrowse, and then that the feed is of type RSS. API specifications are not of a standard representation (see Chapter 2) and so

```
<feed>
  <rss>
  <source>
       <url>http://newsrss.bbc.co.uk/rss/newsonline_uk_edition/business
            /rss.xml</url>
  </source>
  </rss>
</feed>
```

Code Excerpt 1: API VoiceBrowse Specification

to make the operation of an API generic with the Content Spotter, it is necessary to first understand that API usage consists of two aspects: an XML specification, that is required to be submitted to a specified server; and required parameters, that are to be inserted into the XML specification. With this is mind, the following XML specification was devised to represent an API feed plug in for VoiceBrowse, shown in Code Excerpt 2.

Code Excerpt 2 shows the API specification for an eBay API and it makes known to the Content Manager that one parameter for this API operation is required to be completed by the user, namely the 'item' parameter. The XPATH contained as the node's value is the path where the Content Manager is required to insert the elicited information into the API schema.

The <schema> and <url> elements respectively inform the Content Manager where the API schema can be found on the local hard drive, and then where the API must be sent to once the required parameters have been gathered from the user.

In addition, it is also a common feature of an API request that so called 'header' parameters are attached to the API specification. These are denoted in the above <parameters> by use of a '£' symbol in front of the name attribute. During parsing, the

```
<feed>
  <api title="ebay search">
    <request>
      <schema>feeds\APIs\ebaySearchRequest.xml</schema>
      <url>https://api.ebay.com/ws/api.dll</url>
      <parameter name="item">GetSearchResultsRequest/Query</parameter>
      <parameter name="£X-EBAY-API-COMPATIBILITY-LEVEL">433</parameter>
      <parameter name="£X-EBAY-API-DEV-
NAME">R2DJD8XR5XUB61JL13O817F74V3EG3</parameter>
      <parameter name="£X-EBAY-API-APP-
NAME">CRAIGWOOTTW4O3AG1AE6KUVF43MR38</parameter>
      <parameter name="£X-EBAY-API-CERT-NAME">U11QJF8O63D$293555B4F-
JQJPNPZ9</parameter>
      <parameter name="£X-EBAY-API-CALL-
NAME">GetSearchResults</parameter>
      <parameter name="£X-EBAY-API-SITEID">3</parameter>
      <parameter name="$xmlns">ebay
urn:ebay:apis:eBLBaseComponents</parameter>
    </request>
    <response>
      <parameter name="title">Ebay search</parameter>
      <parameter name="list">//ebay:Title</parameter>
    </response>
  </api>
</feed>
```

Code Excerpt 2: VoiceBrowse API Specification

Content Manager will ignore these parameters when collecting information from the user, but instead use the value of the nodes as header information. For example, the first head parameter is ‘X-EBAY-API-COMPATIBILITY-LEVEL’, and will take the value of 433.

Complementing the <request> information of the API specification is the <response> information, used once the Content Manager has made the API request and receives the results back from the server in the form of an XML file. The <response> parameters advise the Content Manager of the XPATH to be used to retrieve the relevant parts of the results file, which are in turn output to the user via a VoiceXML form.

To promote the extensibility of VoiceBrowse, each feed was specified in a separate file, allowing feeds to be added and removed from the system without affecting other parts of the code or the system's functionality. It was required then to integrate the information from the various feed files into one XML file for easy access and parsing by the Content Manager. To achieve this, the Content Manager loads the feed XML files one at a time, all held in the 'feeds' directory, and then inserts the information into a document 'feeds.xml'. Any time the feeds directory is changed, feeds.xml is cleared, and the function executed again to populate the file with the latest information. Code Excerpt 3 shows the ASP.NET code to realise this.

```
1. xmlTempDoc.WriteStartElement("feeds")
2. Dim currentDir As ObjectModel.ReadOnlyCollection(Of String) =
   My.Computer.FileSystem.GetFiles(System.Configuration.Configuration
   Manager.AppSettings("root") + "feeds")
3. For Each oneFile As String In currentDir
    4. xmlTempDoc.WriteStartElement("feed")
    5. xpathDoc = New XPathDocument(oneFile)
    6. xmlTempDoc.WriteStartElement("filename")
    7. xmlTempDoc.WriteString(oneFile)
    8. xmlTempDoc.WriteEndElement()
    9. If xmlNav.Evaluate("count(//rss)") > 0 Then
      10.xpathDoc = New XpathDocument(xmlNav.SelectSingleNode("/
               feed/rss/source/url").Value.ToString)
      11.xmlTempDoc.WriteStartElement("type")
      12.xmlTempDoc.WriteString("rss")
      13.xmlTempDoc.WriteEndElement() '</type>
      14.xmlTempDoc.WriteStartElement("title")
      15.xmlTempDoc.WriteString(xmlNav.Evaluate("string(normalize-
         space(string(translate(/rss/channel/title,'£$*%^&*()!@:;<>=-
         #~`¬¦€|\/','')))))").ToString.ToLower)
      16.xmlTempDoc.WriteEndElement() '</title>
   17.   Else
      18.xmlTempDoc.WriteStartElement("type")
      19.xmlTempDoc.WriteString("api")
      20.xmlTempDoc.WriteEndElement() '</type>
      21.xmlTempDoc.WriteStartElement("title")
      22.xmlTempDoc.WriteString(xmlNav.SelectSingleNode("feed/api").G
         etAttribute("title", "").ToLower)
      23.xmlTempDoc.WriteEndElement() '</title>
   24. End If
   25. xmlTempDoc.WriteEndElement() '</feed>
   26. Next
   27. xmlTempDoc.WriteEndElement() '</documents>
   28. xmlTempDoc.WriteEndDocument()
```

Code Excerpt 3: Function to Produce List of Current Feeds

Line 2 in the above code creates a collection of files in the feeds directory, each of which will be accessed during the loop specified between lines 3 and 26. The body of the loop writes the path of the filename to the feeds.xml document (lines 6-8), and then enters a conditional branch based on whether the current file represents a RSS feed or an API feed (line 9). If the current file is an API, it simple uses the file information to write the 'title' attribute of the feed to feeds.xml. However if it is an RSS feed, the title must be retrieved from the source URL. To achieve this, the source of the RSS feed, held in current file of the loop, is loaded as an XML file into the system, and XPATH is used to access the 'rss/channel/title' path (line 15), which is then placed into feeds.xml. The XPATH function 'translate ()' is used here to remove any illegal characters with respect to XML grammars and prompts.

To produce the document space, documents.xml, the Content Spotter is required to iterate through each of the RSS documents located at the source URLs held in feed.xml, and then paste in the title, the description, and the source elements for each <item> element. This is done in similar method to extracting the <title> from the RSS feed above, however Code Excerpt 4 shows an additional piece of processing required when inserting the <description> elements.

As it is the content of the <description> elements in the RSS channel that will eventually become the content of the VoiceXML prompt, text and coding not suitable for dialogue is required to be removed. It is common that <description> elements contain some HTML or other mark-up to aid the graphical rendering of the RSS feed, all of which would be un-needed auxiliary information when used in dialogue. The process to remove the mark-up is based on the usage of '<' and '>' to denote the start and end of marked up text. Firstly, the content of the description element is extracted

```
 1. xmlTempDoc.WriteStartElement("description")
 2. s = node.Evaluate("string(normalize
    space(string(translate(.,'£$*%^&*()!@:;=
    _#~`¬¦€|\/',''))))").ToString.ToLower
 3. If s.Contains("<") Or s.Contains("&lt;") Or s.Contains("&gt;") Then
     4. html = False
     5. For Each word As String In s.Split(" ")
           6. If word.Contains("<") Or s.Contains("&lt;") Then
                7. html = True
           8. End If
           9. If html = False And Not word.Contains("src=") Or Not
              word.Contains("width=") Or Not word.Contains("height=") Then
                10.    xmlTempDoc.WriteString(word..ToString + " ")
           11.   End If
          12. If word.Contains(">") Or s.Contains("&gt;") Then
                   13. html = False
           14.   End If
    15.   Next
16. Else : xmlTempDoc.WriteString(s..ToString)
17. End If
18. xmlTempDoc.WriteEndElement() '</description>
```

Code Excerpt 4: Function to Produce Document Space

and stored in a local variable, shown in line 2. Next, the Content Manager checks the description for the presence of any marked up text. If not, the description can be written straight to the document space, otherwise the description is then split into an array of words (line 5). A loop then iterates through each word, and if it contains no mark up, it is output to the document space. If the presence of a starting character '<' is detected, then no words are output until the presence of the ending character '>' is detected. The result of the above function is a document list, with each document containing a title and short description of a story, the title of the RSS that the story came from, and also the source of the web page that contains the full story.

To fetch the main story from a website is a two part process: the first part downloads all the HTML from the source URL, given by the <source> element in the document space; and then the source HTML is 'tidied' to conform to W3C standards, with all text

```
 1. Dim filename As String = Date.Now.Ticks.ToString + ".xhtml"
 2. aRequestedHTML = objWebClient.DownloadData(url)
 3. strRequestedHTML = objUTF8.GetString(aRequestedHTML)
 4.         FileIO.FileSystem.WriteAllText("C:\extractedHtml\webPage.ht
            ml",strRequestedHTML, False)
 5. myProcess.StartInfo.FileName = "c:\extractedHTML\tidy.exe"
 6. myProcess.StartInfo.Arguments = "-config
    c:\extractedHTML\htmlTidyConfig.txt -o c:\extractedHTML\" +
    filename + " c:\extractedHTML\webPage.html"
 7. myProcess.StartInfo.WindowStyle =
    System.Diagnostics.ProcessWindowStyle.Hidden
 8. myProcess.Start
 9.     While IO.File.OpenRead("C:\extractedHTML\" + filename).CanRead
        = False
10. End While
11. Return filename
```

Code Excerpt 5: Function to Fetch Body of Content From Source URL

bodies enclosed in HTML <p> tags. Code Excerpt 5 shows this process implemented in VoiceBrowse. Lines 2 – 4 show the ASP.NET for downloading HTML as a String Object from the source URL. This is then placed into a temporary HTML file, which will be used as an input parameter during the execution of the HTMLTidy program. As discussed in Section 5.1, HTMLTidy is a freely available program that transposes illegal HTML code to W3C compliant HTML. This will enable standard parsing of the downloaded content, as used in Code Excerpt 22.

Lines 5 and 6 show the ASP.NET required to run the HTMLTidy executable, and line 8 shows VoiceBrowse starting the process in the background. The While loop that follows then holds the thread of execution until the background process has completed, and then finally returns the name of the created file back to the body of code that had called this function (line11). The newly created file contains the tidied HTML code from the downloaded web page, and can then be used by VoiceBrowse later in the interaction to extract the main body of content for output to the user.

```
1. if (isset($_FILES['query']))
2. { if
   (move_uploaded_file($_FILES['query']['tmp_name'],"App_Code/temp/to
   Recognise.wav")) {
3. $code = 1;
4. } else {
5. $code = 0;}
6. } else {
7. $code = 0;
8. }
9. ?>
```

Code Excerpt 6: PHP To Save <record> Audio Data

7.5 VoiceBrowse Implementation: Dialogue Manager

The main task of the dialogue manager is the recognition of a spoken utterance from the user during interaction with the open version of VoiceBrowse. The decision making process of how to achieve this in the VoiceXML setup of VoiceBrowse is discussed in Section 7.9, however it was decided that the Microsoft Speech Recogniser would be used to translate the user's speech to text. This was a three step process:

1. Use a VoiceXML <record> element to allow the user free speech with the system.
2. Submit the <record> audio to a PHP script which will save the audio to a .wav file - shown in Code Excerpt 6.
3. Finally, use VisualBasic.NET to translate the .wav file to text using the Microsoft Speech Recogniser.

The PHP script firstly checks that the data has been received ok (line 1), and then saves the audio data to a local file, Recognise.wav (line2). A local variable is set to either 1 or 0 depending upon the success of the save operation. With the user's speech now saved to a local .wav file, control can be handled back to the next VoiceXML state, which will

```
1. Function recognise(ByVal num As Integer) As Integer
2. myProcess.StartInfo.FileName =
   "c:\extractedHTML\VoiceBrowseConsoleTaskDictation.exe"
3. myProcess.StartInfo.WindowStyle =
   System.Diagnostics.ProcessWindowStyle.Hidden
4. Try
     5. myProcess.Start()
     6. myProcess.WaitForExit()
     7. Return 1
8. Catch ex As Exception
     9. Return 0
10. End Try
11. Return 1
```

Code Excerpt 7: Function to Start Speech Recognition Process

use a Dialogue Manager function to recognise the audio data from the .wav file into plan text. This recognition() function is shown as Code Excerpt 7.

The function 'recognise()', shown here, works in a similar fashion to fetch the main content from a story's website, by starting a system process to run an executable file (line 2 and 5). It is the executable file that contains the needed VisualBasic.Net code to recognise the audio .wav file data into text, shown in Code Excerpt 8.

```
1. For Each document As System.Xml.XPath.XPathNavigator In xmlIT
     2. For Each word As String In document.Value.ToString.Split(" ")
          3. If word.Contains("A") Or word.Contains("B") Or
             word.Contains("C") Or word.Contains("D") ... Or
             word.Contains("Z") Then
              4. documentGrammar.Append(word.Trim.ToString, 0, 1)
          5. End If
     6. Next
7. Next
8. Dim grammar As New Speech.Recognition.Grammar(documentGrammar)
9. grammar.Enabled = True
10. reco.LoadGrammar(grammar)
11. reco.LoadGrammar(New Speech.Recognition.DictationGrammar())
12. reco.SetInputToWaveFile("App_Code\temp\toRecognise.wav")
13. Dim result As Recognition.RecognitionResult = reco.Recognize
14. IO.File.WriteAllText("App_Code\temp\recognised.txt", result)
```

Code Excerpt 8: Visual Basic.Net Speech Recognition Function

To perform the recognition function on the .Wav file, it is first necessary to activate the grammars for the recognition engine. As discussed in the next section, two different grammars are required to be activated in order to perform the recognition, a dictation grammar, available with the recogniser, and a document grammar containing the out of vocabulary words, not in the dictation engine. The two loops shown here (lines 1 – 7) iterate through each document in the document space, and then through each word in the document, and if an out of vocabulary word is detected, this is added to the document grammar (line 4). As discussed in the following section, there is currently no mechanism in the current implementation of the System.Speech namespace to detect those words not in the dictation grammar, and so this is currently done on the assumption that the dictation grammar is a comprehensive grammar of the English language, and it is the proper nouns in the document space, denoted by use of a capitalised first letter, that need to be added to the document grammar.

After the completion of the loops, the document grammar, along with the dictation grammar, are activated and loaded into the recognition engine (line 10 and 11), the .wav file containing the saved audio data from the user is set as the recogniser's input, and the recognition process begins (lines 12 and 13). The final line of code, line 14, saves the recognised text to a local text file, recognised.txt. Control of the interaction will then return to the VoiceXML state that called the recognition function, which will confirm the recognised text with the user, discussed later in the Chapter.

Another important function of the Dialogue Manager is to provide dialogue history controls to VoiceBrowse, specifically to record the path of the dialogue through the interaction, and to retrieve the previous states of the interaction from memory. From

```
1. dataHistory.InsertCommand = "insert into tblHistory(previousURL,
   currentURL, time) Values(@previousURL, @currentURL, @time)"
2. dataHistory.InsertParameters.Add("currentURL", currentUrl)
3. dataHistory.InsertParameters.Add("previousURL", previousURL)
4. dataHistory.InsertParameters.Add("time", TypeCode.DateTime,
   Date.Now)
5. dataHistory.Insert()
```

Code Excerpt 9: Function to Add new Dialogue History Record to Database

the Call Flow Diagrams above, it is this 'back' function that provides essential navigation through the document space, the operation of which can be compared to the back button or functionality of a graphical web browser. With each ASP.NET script being accessed from the Web Server by means of a HTTP request, the address of the current and previous scripts can be stored in memory. The back mechanism can then be implemented by retrieving the address of the previous script accessed by the Web Browser. To assist with this functionality, the Dialogue Manager makes use of the database server available in the environment, Microsoft SQL Server Express 2005, and Code Excerpt 9 shows the code required to store the current and previous URLs of the scripts into the database.This function will be executed each time a script is accessed, and line 1 also stores the time that the current script has been accessed. For operation of the back function therefore, the Dialogue Manager should retrieve the value held in the 'previousURL' field of the last record to be added to the database, got by decrease sorting on the 'time' field. Code Excerpt 10 shows the specification of the back functionality in ASP.NET:

```
1. dataHistory.SelectCommand = "SELECT previousURL FROM tblHistory
   WHERE currentURL = '" + url + "' ORDER BY time DESC"
2. dvHistory = dataHistory.Select(selArg)
3. Return dvHistory.Item(0).Item(0).ToString
```

Code Excerpt 10: Function to Get Previous Dialogue State From History

7.6 VoiceBrowse Implementation: Information Based Dialogues

The pseudo code shown in Table 7.2 overleaf defines the various scripts that were required to be implemented from the Call Flow and Prompt Design Diagrams with respect to information based dialogues.

Code Excerpt 11 shows the main ASP.NET statement that was used during the implementation, the output statement. Normally used for outputting text to a graphical system, the 'Write' method of the 'Response' object was used in this case for outputting VoiceXML statements to the Prophecy Server.

```
1.    Response.Write("VoiceXML text to go here")
```

Code Excerpt 11: ASP.Net Response Statement

The next increment of development was the implementation of the call flow diagram itself – starting with the root document of the system. Root.aspx, as defined in the pseudo code above, contains the global commands and specification for handling the 'main menu' and 'back' functionality of VoiceBrowse. In addition, the error handling routines for <nomatch> and <noinput> events are defined in root.aspx to globally handle any cases of the user utterances not conforming to the active grammar or not being heard by the system respectively. More specific error routines are defined in further documents when needed. Standard VoiceXML <link>, <nomatch> and <noinput> elements are used here to handle the main menu and error specifications, and the role of the back command is catered for by using an additional <link> command that directs to an ASP page containing the function from Code Excerpt 10.

Filename	Pseudo Code
root.aspx	Create the VoiceXML required to return the user to informationStart.aspx if 'main menu' or 'start over' is spoken by the user. Create the VoiceXML required to return the user to the previous state in the dialogue if 'back' is spoken by the user. This must be in the root document as it is to be available at all stages throughout the dialogue. Define error handlers for noinput and nomatch events. Start the recording event to record the dialogue to .wav file Play welcome prompt, and pass to informationStart.aspx.
informationStart.aspx	Read in the titles of each feed available to VoiceBrowse. Use these titles to define an XML grammar of available items Create a VoiceXML form to collect the user's choice of feed using the above XML based grammar. Create a sub-dialogue to handle a 'help' request from the user. At the end of this sub-dialogue, control will pass back to the above form to collect the user's choice of feed. At the end of this interaction, pass control on to disambiguate.aspx, along with the user's utterance. Alternatively, if the open version of VoiceBrowse is in use, then accept the user's input via a VoiceXML <record> element to allow for the free form input of a user's query. Then pass the audio data to saveRecord.php to create a local .wav file.

disambiguate.aspx	Receive the input from the previous file, and fetch the feeds with matching words in the title. If numbers of matching feeds = 1, then the user has requested a unique feed, so proceed to listContentItems.aspx, passing the title of the unique feed to this file. If the unique feed is an API, pass to apiCollectRequest.aspx. If number of matching feeds = 0, then no feeds have been found with the user's request, and so pass control back to informationStart.aspx. If the number of matching feeds > 1, then it is not possible to determine the requested feed from the current utterance. Create a XML grammar from the matching titles, and generate a VoiceXML form to iterate through the possible feeds. Once identified, pass the feed title and proceed to listContentItems.aspx.
listFeeds.aspx	Available during the informationStart.aspx phase when the closed version is in use, output the feed titles currently in the VoiceBrowse environment 5 at a time to the user with an associated title number. Use a loop counter to go from its value + 3. The loop counter will be incremented if the user wishes to navigate forward in the list. A XML grammar file will allow the user to repeat the three titles, go back in the list of titles, move on to the next three results or say done to go back to informationStart.aspx.

Table 7.2: VoiceBrowse Pseudo Code For Information Based Dialogues

One noteworthy element used in the root.aspx document is the Voxeo proprietary extension to the VoiceXML language <voxeo:recordcall>, which, when used, creates a .wav file of the interaction with the Voxeo Prophecy Server. The element is used in root.aspx as shown in Code Excerpt 12, and the specification of the element's attributes is included as Table 7.3

Attribute Name	Description
value	'Value' specifies which percentage of calls to the Voxeo Prophecy Server is recorded as .wav files. If the value of this attribute is 100, then all calls will be recorded.
info	The value of 'Info' attribute is amended to the filename of the created .wav file for identification purposes.

Table 7.3: Voxeo RecordCall Attribute Specification

```
Response.Write("<voxeo:recordcall value='100' info='voiceBrowse'/>")
```

Code Excerpt 12: Voxeo Proprietary RecordCall Element

The implementation of the record call element in VoiceBrowse therefore records all calls placed to the Voxeo Prophecy server, which can then be used during the evaluation and analysis phase of the project.

The root document of a VoiceXML system is primarily used to create and initialise global variables, and create the global menu controls and commands, and hence the control is passed straight to informationStart.aspx, the purpose of which is to retrieve the user's intentions with regard to their desired information source and type. As

```
1  While xmlFeedIt.MoveNext
     2  For Each word As String In xmlFeedIt.Current.Value.Split(" ")
          3  If Not grammar.Contains(word) Then
          4  grammar = grammar + "<item repeat='0-1'>" + word + "
             </item>"
     5  End If
6  Next
7  End While
8  Response.Write("<form id='main'>")
9  Response.Write("<field name='category'>")
10 Response.Write("<grammar version='1.0' root='choice'"))
11 Response.Write("<rule id='choice'>")
12 Response.Write(grammar)
13 Response.Write("</rule>")
14 Response.Write("</grammar>")
15 Response.Write("<prompt>Say help for assistance. Please say the
   provider and also the category if you wish,  or if you want to
   know what is available, say list!</prompt>")
16 Response.Write("<filled>")
17 Response.Write("<if cond=""category=='list'"">")
18 Response.Write("<var name='position'  expr='1'/>")
19 Response.Write("<submit next='listFeeds.aspx'
   namelist='position'/>")
20 Response.Write("<elseif cond=""category=='help'""/>")
21 Response.Write("<goto next='#help'/>")
22 Response.Write("<else/>")
23 Response.Write("<submit next='disambiguate.aspx'
   namelist='category'/>")
24 Response.Write("</if>")
25 Response.Write("</filled>")
26 Response.Write("</form>")
```

Code Excerpt 13: informationStart.aspx In Closed Version

discussed previously, this is implemented as two different dialogue strategies, a ‘closed’ approach and an ‘open’ approach, shown respectively in Code Excerpts 13 and 17.

The closed approach, in terms of VoiceXML specification, requires more effort at this early stage as a relevant grammar is required to be constructed to allow the user’s input to be recognised. At this stage of the dialogue, the grammar is to contain the titles of the RSS and API feeds currently in the VoiceBrowse environment, extracted from the feeds.xml file.

Lines 1 – 6 in Code Excerpt 13 show this process – while there are <feed> elements in feeds.xml (line 1) split the feeds title into separate words (line 2), and place each word

into a <item> element if the grammar does not already contain the current word (lines 3 and 4).

The body of informationStart.aspx is then a VoiceXML <form> construct, in one field of which the constructed grammar is active. A conditional statement is specified in the <filled> specification of the field to catch utterances containing the word 'list' which are passed to the listFeeds.aspx document. Not discussed here, but included on the attached Code CD, list.aspx makes up part of the help functionality of VoiceBrowse which iterates through the currently installed RSS and API Feeds in the VoiceBrowse environment – essentially the content from the While Loop discussed above is placed into a VoiceXML prompt.

The rationale for splitting the titles into separate words, each of which become an allowable grammar item is twofold: to not make the feed selection restrictive but to match as many feeds as possible to the current input; and because users will not commonly know the exact arrangement of words in an RSS <title> element. For example, consider the RSS feed from the BBC for the Northern Ireland News Headlines. If a user wished to request this feed from VoiceBrowse, and grammar creation was not done by word division, the user would be required to say the exact phrase "BBC News Northern Ireland Edition", which is the title of the RSS feed, which in turn becomes the grammar of informationStart.aspx. However, by breaking the <title> elements into separate words, each of which become part of the active grammar, the user could simply state "News", or "Northern Ireland".

The uttered terms are then passed to and used by disambiguate.aspx to select any matching feeds, such as "BBC News Northern Ireland Edition" or "RTE Northern Ireland News" if the words "News" was uttered (Code Excerpt 14). The output of this

```
1. category = Request.QueryString.Item("category").ToString
2. If Not Request.QueryString.Item("oldCategory") Is Nothing Then
    3. category = category + " " +
       Request.QueryString.Item("oldCategory").Trim.ToString
4. End If
5. xmlFeedIt = xmlNav.Select("feeds/feed/title")
6. While xmlFeedIt.MoveNext
   7.  contains = True
   8.  For Each word As String In category.Split(" ")
       9.  If Not xmlFeedIt.Current.Value.Contains(word) Then
      10.  contains = False
      11.  Exit For
      12. End If
   13. Next
   14. If contains Then
      15. count = count + 1
      16. If count = 1 Then
              17.  type =
                   xmlFeedIt.Current.SelectSingleNode("type").Value
      18. End If
      19.    For Each word As String In
             xmlFeedIt.Current.Value.ToString.Split(" ")
          20.  If Not uniqueTitle.Contains(word) Then
             21.  uniqueTitle = uniqueTitle + word + " "
          22.  End If
      23. Next
   24. End If
25. End While
26. uniqueTitle = uniqueTitle.Trim
```

Code Excerpt 14: Function to Match User Utterance To Available Feeds

function is to then present the user with the available options of matching feeds, of which the user can select one to continue. Not only does this allow greater flexibility when inputting a category of content, it further refines a user's query so that content returned is not too generic i.e., 'News' could refer to 'News' or 'Sports News' and so disambiguate.aspx will prevent this.

Not shown in Code Excerpt 14 is the grammar construction, which is once again constructed from the feed titles in feeds.xml to allow further input by the user at this stage. Execution begins by retrieving the value of the user's utterance from informationStart.aspx by accessing the querystring variable 'category' (line 1), which is the value that must be compared to the feed titles. Lines 2 – 4 perform a vital function

in this disambiguation stage of adding the new phrase, uttered by the user, to the old phrase previously given. This is done as although the current state is disambiguate.aspx, and the user responds with an applicable option put forward by VoiceBrowse, control is once again passed back to disambiguate.aspx. This will allow VoiceBrowse to confirm that the query is specific enough to continue, and if not, then a new set of options is to be presented to the user. Consider the response to an initial query 'News' for example, to which VoiceBrowse asks the user to clarify between 'World' and 'Sports' news. If the user's response is "Sports", the query at this stage could still refer to any number of content types, such as 'Football News' or 'Rugby News'. By passing control back to disambiguate.aspx, VoiceBrowse can continue to refine the query with input from the user, until it is specific enough to proceed to the next state.

Due to current limitations of the XPATH specification (see Section 7.5), matching feed titles from the document space must be identified using a loop to iterate through each one and compare each word in the title to each word in the query separately. Line 5 shows the .NET to retrieve the list of feed titles from the documents.xml file, and so it is the While loop in lines 6 to 25 that performs the main functionality of disambiguate.aspx.

The condition of the While loop (line 6) ensures that the iteration continues while there are feed titles to compare. The first stage of the While loop is to iterate through each word in the query (line 8), and compare each word in the query to the content of the feed title (line 9). Once the For loop has completed all iterations, a Boolean variable 'contains' true or false to indicate if the current feed title contains any words from the query.

Based on this condition (line 14), if the current feed does contain a word from the query, a count variable is incremented by 1 (line 15) that records the number of feed titles that have matching words. If the current feed does not have any matching words, then execution continues with the While loop, which progresses on to the next feed title.

If the current feed is the first feed to be found containing matching words from the query (line 16), then a local variable 'type' is created to record if the current feed is of type RSS or API. This value is used later in disambiguate.aspx if only one matching feed has been found, and dialogue can then proceed to the next state.

The last part of the While loop adds the content from current feed title to a variable 'uniqueTitle', which was used to create the input grammar and output prompt of the interaction. To ensure that there is no duplication of words in the variable uniqueTitle, the feed title is split up into separate words (line 19), and if the variable does not already contain that particular word (line 20) then the word is added to the variable. The result is therefore a variable which contains unique words from the matching feed titles, for example 'World' and 'Football' in response to the input 'News'.

Once the matching feeds have been identified, they must be output to the user, and a grammar constructed of the matching feeds to allow the user to utter the required feed name. The execution of these steps is shown in Code Excerpt 15, and begins with the grammar construction in lines 1 – 3. The feed titles held in the variable uniqueTitle are split into separate words, and a For loop used to insert each word into an <item> element. A <prompt> is then constructed for output (line 11), with each word of the uniqueTitle variable used, separated by a space, as its content (lines 12 – 14). The effect is an output statement that presents the matched feed titles to the user, and asks for a specific one to continue. An example output might be "You have requested News.

```
1. For Each word As String In uniqueTitle.Split(" ")
2. grammar = grammar + "<item repeat='0-1'>" + word + " </item>"
3. Next
4. Response.Write("<var name='oldCategory'  expr=""'" + category.Trim
   + "'""/>")
5. Response.Write("<field name='category'>")
6. Response.Write("<grammar version='1.0' root='choice'")
7. Response.Write("<rule id='choice'>")
8. Response.Write(grammar)
9. Response.Write("</rule>")
10.      Response.Write("</grammar>")
11.      Response.Write("<prompt>You have requested  " + category +
   ". I have found " + count.ToString + " different feeds.  Options
   include: ")
12.      For Each word As String In uniqueTitle.Split(" ")
13.      Response.Write(word + ",  ")
14.      Next
15.      Response.Write(".  Which would you like?</prompt>")
```

Code Excerpt 15: Outputting Matched Feeds to User and Waiting For Input

I have found 2 different feeds. Options include BBC Sport News, BBC World News, Sky News. Which would you like?"

Once input is detected from the user, control is passed back to disambiguate.aspx to ensure that the user's query now identifies a unique feed with which to continue the dialogue. If so, recall that the result of feed title matching algorithm will produce a variable 'count' with a value of 1, indicating that only one feed has been identified that matches the user's input. If so execution takes one of two possible branches based on the type of the unique feed type being either of type RSS or API. A 'count' value of 0 indicates that VoiceBrowse has failed to identify any feeds matching the query.

Code Excerpt 16 shows the execution of this branching in disambiguate.aspx. Lines 1 to 3 contain the condition to catch any Scenario where no matching feed title has been found, and so a relevant error message is output to the user, and then control is passed back to the informationStart.aspx state.

```
1.     If count = 0 Then
2.  Response.Write("<block>I haven't found any options matching your
    query”)
3.  Response.Write("<submit next='informationStart.aspx' /></block>")
4.     Else
5.  If type = "rss" Then
6.  Response.Write("<var name='position'  expr='1'/>")
7.  Response.Write("<var name='category'  expr=""'" +
    uniqueTitle.Trim + "'""/>")
8.  Response.Write("<block>OK, I have found " + uniqueTitle.Trim
    + ".  There are")
9.  Response.Write("<submit next='listContentItems.aspx'
    namelist='category position'/></block>")
10. Else
11. Response.Write("<var name='position'  expr='1'/>")
12. Response.Write("<var name='count'  expr='1'/>")
13. Response.Write("<var name='provider'  expr=""'" +
    uniqueTitle.Trim + "'""/>")
14. Response.Write("<block>OK, I have found " + uniqueTitle.Trim
    + ".")
15. Response.Write("<submit
    next='http://localhost/vbClosed/apiCollectRequestVXML.aspx'
    namelist='provider position count'/></block>")
16. End If
17.    End If
```

Code Excerpt 16: Transitions to Informative or Task Based Dialogues

Line 5 checks if the matching feed is of type RSS – if so, the required query string variables for listContentItems.aspx are created, a confirmatory message is output to the user containing the feed found by VoiceBrowse, and control is passed on to listContentItems.aspx. If the proposed feed was of type API, the query string parameters are created that are required by APICollectRequest.aspx, and control is passed to this state (see Section 7.8).

As discussed previously, the open approach of the system, shown in Code Excerpt 17, utilises a VoiceXML <record> element (Lines 1- 8) to allow free form speech input by the user, which is then passed to a PHP script which saves the audio data as a .wav file (refer to Code Excerpt 6). Line 10 shows this HTTP submission of the audio data to the PHP script, which in turn, after the creation of the .wav file, passes control back to recogniseDictation.aspx (Code Excerpt 18).

```
1  Response.Write("<record name='query' beep='true' maxtime='10s'
   finalsilence='3000ms'>")
2  Response.Write("<prompt>")
3  Response.Write("What is your query?")
4  Response.Write("</prompt>")
5  Response.Write("<noinput>")
6  Response.Write("Sorry, I did not hear anything. <reprompt/>")
7  Response.Write("</noinput>")
8  Response.Write("</record>")
9  Response.Write("<filled>")
10 Response.Write("<submit method='post' enctype='multipart/form-
   data' namelist='query'
   next='http://localhost:9990/saveRecord.php'/>")
11 Response.Write("</filled>")
```

Code Excerpt 17: informationStart.aspx In Open Version

recogniseDictation.aspx makes use of the recognise() function in the Content Manager (Code Excerpt 8) to perform the recognition process on the saved .wav file (line 1). If the recognition has been successful (line 4), control is passed to the same point in the interaction as the Closed Approach, listContentItems.aspx.

```
1. Dim run As Integer = contentManager.recognise()
2. Response.Write("<form id='main'>")
3. Response.Write("<block>")
4. If run = 1 Then
     5. Response.Write("<prompt>Ok. One second!</prompt>")
     6. Response.Write("<var name='position'  expr='1'/>")
     7. Response.Write("<var name='filename'  expr='0'/>")
     8. Response.Write("<submit next=listContentItems.aspx'
        namelist='position filename />")
9. Else
    10. Response.Write("<prompt>I'm sorry, I did not understand, I'll
        try again</prompt>")
     11.  Response.Write("<var name='count' expr='" +
        Request.QueryString.Item("count").ToString + "' />")
     12.  Response.Write("<submit next='informationStart.aspx' />")
13. End If
```

Code Excerpt 18: Speech Recognition On Saved .Wav File

7.7 VoiceBrowse Implementation: Delivery of Online Content

Table 7.4 presents the pseudo code for the relevant scripts needed to output the matched documents and content extracted from online sources:

Filename	Pseudo Code
listContentItems.aspx	Receive input from disambiguate.aspx, and depending on which dialogue strategy is use: either match the input to the titles of all the documents in the document space; or use a COSIM similarity function to match input to similar <description> elements. The output in either case represents a list of introductions that will be output to the user. Create a VoiceXML form that outputs the matched documents from the document list, three at a time with an associated story number. Use a loop counter to go from its value + 3. The loop counter will be incremented below, if the user wishes to navigate forward in the document space. A XML grammar file will allow the user to request further information about a particular story's introduction by use of its number, repeat the three introductions, go back in the list, or move on to the next three introductions. If the user requests more information, pass control to fetchStory.aspx, along with the number of the story being requested, and the title of the feed currently in use to identify the relevant source of the story in the document space. If the user requests to go forward through the list of matching documents, then pass control back to listContentItems.aspx with an incremented count variable for use in the output loop and grammar creation.
fetchStory.aspx	Receive the title of the current feed and the story number request by the user from the previous file. Use the story number to identify the relevant story in the document space, and retrieve its source URL. Access the source URL, and download the HTML. Use HtmlTidy to clean and parse the downloaded HTML, and

	extract the text paragraphs. Use a loop counter to go from its value + 3, outputting 3 paragraphs of text from the main body of content. The loop counter will be incremented below, if the user wishes to navigate forward throughout the story's body. A XML grammar file will allow the user to repeat the three paragraphs, go back in the story's body, or move on to the next three paragraphs. If the user requests to go forward through story's body, then pass control back to fetchStory.aspx with an incremented count variable for use in the output loop and grammar creation.

Table 7.4: VoiceBrowse Pseudo Code For Outputting Content

The execution of listContentItems.aspx is dependent on the dialogue strategy in use: if the closed approach is in use the <document> elements in the document space are identified according to their <title> matching any part of the user's utterance; or identified by performing a Cosine Similarity function on the <description> elements of the document space with the user's speech if the open approach is in use. The former is similar to disambiguate.aspx which matches the content of the <title> elements to the user's input. The call to the Cosine Similarity function in listContentItems.aspx is shown as Code Excerpt 19.

The execution begins by reading into memory the recognised text from the recognition function, the result of which was stored as a local text file (line 1). Starting with the inner most part of the nested brackets in line 4, the recognised words are first converted to lowercase, as is the document space, trailing and leading edge spaces are trimmed from the text, and this is then passed to a function to remove the stop words from the text. Not shown here, removeStopWords() is a rule based function that contains a list of

```
1.     Dim oFile As New
       System.IO.FileStream(System.Configuration.ConfigurationManager
       .AppSettings("temp") + "recognised.txt", IO.FileMode.Open)
2.     Dim oRead As New System.IO.StreamReader(oFile)
3.     Dim query As String = oRead.ReadToEnd
4.     filename =
       f.getSimilarDocuments(f.removeStopWords(query.ToLower.ToString
       ).Trim.ToString)
```

Code Excerpt 19: Cosine Similarity Function of Content Spotter

well used stop words, of which the recognised text is filtered through and removed. The result is then passed into getSimilarDocuments() which performs the Cosine Similarity function - the code for which is included on the attached Code CD.

Independent of the dialogue strategy in use, the execution is essentially similar after matching documents have been fetched, by returning the result set to the user, and allowing the user to navigate through the result list and to select a particular story for VoiceBrowse to retrieve (Code Excerpt 20).

```
1. For i = 1 To Request.QueryString.Item("position") - 1
    2. xmlFeedIt.MoveNext()
3. Next
4.     Response.Write("<prompt>I have found stories. I will read them
       3 at a time.  Say the story number to proceed, or say repeat,
       next or back.")
5. i = 1
6. While xmlFeedIt.MoveNext
     7.  If i > 3 Then
         8.  Exit While
   9.    Else
         10. Response.Write("story " + (i +
             Request.QueryString.Item("position") - 1).ToString + ":
             ")
         11. Response.Write(xmlFeedIt.Current.Value.ToString)
         12. i = i + 1
     13.     End If
 14. End While
15.  Response.Write("</prompt>")
```

Code Excerpt 20: Outputting Matched Documents To The user

The current ‘position’ in the result set of returned documents is key to enabling the user to navigate through the documents. A variable ‘position’ is initialised to 1, and incremented by the number of documents to be returned to the user, currently three documents at a time. So, the first time listContentItems.aspx is accessed, the documents from position 1 to 3 will be output to the user, then if the user requests the next three documents, documents 4 to 6 will be output, and so on. It is important to understand that listContentItems.aspx is reloaded by the VoiceXML every time the user requests the next three documents, and so to control the current value of the position element, it is submitted as a ‘namelist’ attribute, part of the <submit> specification, to listContentItems.aspx, the value of which is then accessed in line 1 by requesting its parameter value. This is used by a For loop to move the XML pointer to the required position in the list of matched documents (lines 2 and 3).

Line 4 specifies the <prompt> element that outputs to the user the matched documents. It is the While loop, from lines 6 – 14, which outputs the current document description to the user and advances to the next, if less than three documents have been outputted to the user (lines 7 – 9), and if there are still documents in the returned list to output. Line 10 therefore outputs the text “Story N” to the user, where N is the story number, calculated by adding the values of the counter variable i that will contain a value 1 -3, to the value of the starting position variable. For example, if after outputting the first three stories the user responds with ‘next’, then the position variable is 3. Once the document has been reloaded, the calculation of line 10 will result in Story 4, Story 5 and Story 6 (1 + 3, 2 + 3 and 3 + 3). Finally it is line 11 that writes the content of the <description> element out to the user. It is important to remember that the <description> element from the document space is created from the <description> element of the RSS feeds.

```
1.     xmlFeedIt =
       xmlNav.Select("/Documents/document/source[contains(../title,'" +
       Request.QueryString.Item("category").ToString + "')]")
2.     For i = 1 To Request.QueryString.Item("story")
   3.     xmlFeedIt.MoveNext()
4.     Next
5.     filename =
       contentManager.fetchMainContent(xmlFeedIt.Current.Value.ToString)
6.     filename = filename.Split(".").GetValue(0)
```

Code Excerpt 21: Fetching Content Body From URL

If the user wishes to access the full story from the provider's web site, then VoiceBrowse must access the web page located at the documents <source> element.

The function to fetch the body of an article's web page is handled by the Content Manager (Code Excerpt 5), and the call to this function is contained within fetchMainContent.aspx (Code Excerpt 21). The source URL of the story is retrieved from the document space using XPATH (line 1). The values of the <source> elements from the document collection are accessed by the XPATH '/Documents/document/source'. Only the <source> elements from the documents that match the active category are retrieved, by using the XPATH String function 'contains()'. In the open version, as an XML document has been created with a rank list of documents, there is no need for the inclusion of this function.

Lines 2 to 4 then simply moves the XML pointer from the start of this list of <source> elements to the correct one, by iterating to the value contained in the query string parameter 'story'. The value of this parameter was passed to fetchMainContent.aspx by the previous document, listContentItems.aspx, and is the value of the VoiceXML <field> that was created in listContentItems.aspx to accept the story number from the user. With the XML pointer now located at the correct <source> element, the

```
1.     xpathDoc = New System.Xml.XPath.XPathDocument("C:\extractedHTML\"
       + filename + ".xhtml")
2.     xmlFeedIt = xmlNav.Select("//xhtml:p[position()>=" +
       (Request.QueryString.Item("position") - 1).ToString + "]", nsMgr)
3. Response.Write("<form id='main'>")
4. Response.Write("<field name='story'>")
5. Response.Write("<prompt>")
6. i = 1
7. While xmlFeedIt.MoveNext
     8. If i > 3 Then
         9. Exit While
    10. Else
        11. Response.Write(xmlFeedIt.Current.Value.ToString + ". ")
        12. i = i + 1
    13. End If
14. End While
15.    Response.Write("Please say repeat, next, back, or main
       menu.</prompt>")
```

Code Excerpt 22: Outputting Main Content Body To User

element's value is passed to the fetchMainContent() function, which creates a local copy of the remote story for VoiceBrowse to use (refer to Code Excerpt 5).

Once this function has completed, VoiceBrowse must output the fetched content out to the user. Code Excerpt 22 shows this function, and it is similar in execution to listContentItems.aspx, in that three paragraphs of text are output at a time to the user instead of three stories - to which the user can again respond next, back or repeat, to navigate through the body of text.

Line 1 sets the XHTML file to be output to the resulting XHTML from the fetchMainContent() function, passed as the variable 'filename'. XPATH is then used to extract all the <p> elements from the XHTML file (line 2), once again using the XPATH node function 'position()' to set the .NET XML pointer to the relevant <p> element in the file. The operation of this 'position' element is similar in function to that in listContentItems.aspx – it is incremented with each execution, and passed as a query string variable to itself if the user requests 'next'.

The result of lines 1 and 2 is then a set of XML Nodes, each containing a paragraph of text to be output to the user. Lines 3 to 5 simply set out the VoiceXML <form> specification to be used for the input and output of information, and it is the While loop from lines 7 to 14 that then outputs the paragraphs of text to the user. The condition of the While loop in line 7 is met if there are still nodes, or paragraphs, of text to be output, and if so, the preceding If statement (line 8) controls the conditions of outputting only three paragraphs at a time to the user. If both conditions are satisfied, line 11 outputs the paragraph text, accessed by the current node's 'Value' attribute, to the user. The text itself has been wrapped inside a VoiceXML <prompt> statement, which concludes with the allowable instructions for the user, as defined during the prompt design phase of development (section 7.3).

7.8 VoiceBrowse Implementation: Task Based Dialogues

Table 7.5 overleaf shows the pseudo code required to handle task based dialogues with VoiceBrowse. Dialogue control is passed to apiCollectRequest.aspx from informationStart.aspx if VoiceBrowse proposes that the current query is to be handled by an API (Code Excerpt 23). Task based dialogues are more difficult to handle with regard to language understanding than information based dialogues as the user's input cannot be anticipated . Due to the generic nature of VoiceBrowse, the Content Manager must allow the interaction with APIs from many different domains completing many different tasks. The inputs therefore have to be treated just as generic – one task based dialogue could be ordering computer parts of eBay whilst the next could be booking a flight from Belfast to London.

Filename	Pseudo Code
apiCollectRequest.aspx	Receive the API chosen by the user from disambiguate.aspx. Identify the API's specification, and extract the list of parameters that must be elicited from the user. Output the parameters one at a time, and pass control to saveAPIvariables.php to recognize the user's input. Confirm the parameter's value before moving on to the next. Once all parameters have been collected from the user, proceed to apiResponse.aspx.
apiResponse.aspx	Receive the parameters elicited from the user in apiCollectResponse.aspx. Insert the values into the relevant paths in the API Schema. Submit the API Schema via HTTP request to the relevant URL. Receive the API response, and extract the results. Output the results three at a time with an associated result number. Use a loop counter to go from its value + 3. The loop counter will be incremented if the user wishes to navigate forward in the result set. A XML grammar file will allow the user to request a particular result number, repeat the three results, go back in the list of results, or move on to the next three results.
saveAPIvariables.php	Receive the data from apiCollectVariables.aspx, and save it as a .wav file. Pass control to apiCollectVariables.aspx to recognize the user's speech and confirm.

Table 7.5: VoiceBrowse Pseudo Code For Task Based Dialogues

```
1.     Dim api As String =
       xmlNav.Evaluate("string(feeds/feed/filename[contains(../title,'"
       + Request.QueryString.Item("provider").ToString + "')])")
2.     Response.Write("<record name='query' beep='true' maxtime='5s'>")
3.     Response.Write("<prompt>")
4.     Response.Write("Please tell me the " +
       xmlfeedIT.Current.GetAttribute("name", "").ToString)
5.     Response.Write("</prompt>")
6.     Response.Write("</record>")
7.     Response.Write("<filled>")
8.     Response.Write("<submit method='post' enctype='multipart/form-
       data' namelist='query " + namelist + "'
       next='http://localhost:9990/saveAPIvariables.php'/>")
9.  Response.Write("</filled>")
```

Code Excerpt 23: Collecting API Parameter From User

To cater for the input generically, the recognition method similar to that of the open system was used – use a <record> element to allow the user to specify their answer, which is a parameter of the API, and then use the Content Manager's recognise() function to save the audio to text and proceed to the next parameter.

Firstly, the Dialogue Manager identifies the correct API in use by accessing the query string variable 'provider' that contains the title of the API, specified by informationStart.aspx (line 1). Once retrieved, this is then passed into an XPATH contains() function to identify the filename of the API specification to be used, stored as a string variable 'api'. This filename can be used to load the API specification into memory, needed to access the required parameters for which values must be collected from the user. Not shown in Code Excerpt 23, but included on the attached Code CD, is a count variable that keeps track of the current parameter being output to the user. Once confirmed by the user that their input has been recognised correctly, the count variable increments by 1 and moves the pointer onto the next parameter to be output.

```
1.     contentManager.recognise(0)
2.     Response.Write("<field name='confirm' type='boolean'>")
3.     Response.Write("<prompt>Did you say:")
4.     Dim oFile As New
       System.IO.FileStream(System.Configuration.ConfigurationManager.Ap
       pSettings("temp") + "recognised.txt", IO.FileMode.Open)
5. Dim oRead As New System.IO.StreamReader(oFile)
6. Response.Write(oRead.ReadToEnd)
7. Response.Write("</prompt>")
8. Response.Write("<filled>")
9. Response.Write("<if cond='!confirm'>")
10. Response.Write("<submit next='apiCollectRequestVXML.aspx'/>")
11. Response.Write("<else/>")
12.    Response.Write("<var name='position'  expr='" +
       Request.QueryString.Item("position").ToString + 1) + "'/>")
13. Response.Write("<submit next='apiCollectRequestVXML.aspx'/>")
14. Response.Write("</if>")
15. Response.Write("</filled>")
16. Response.Write("</field>")
```

Code Excerpt 24: Confirming API Parameter Uttered By User

Lines 2 – 7 specify the <record> element that will allow the user to input their answer to the <prompt> (line 4). Generic wording is used to ensure the function can be used in any task based dialogue, and the specific parameter to be provided by the user is inserted into the prompt by using the 'GetAttribute' method of the XML Node Class in ASP.NET. The <filled> element specified from lines 7 – 9 then submits the audio data to the PHP script saveAPIvariabhle.php, which saves the audio data to a wave file. Control is then passed back to apiCollectRequest.aspx.

The next step is then to get the user's confirmation of the recognition process, shown in Code Excerpt 24. Line 1 invokes the recognise function() of the Content Manager, which translates the audio .wav file produced by the PHP script into a text file containing the recognised words by the Microsoft recogniser. To confirm the recognition, VoiceBrowse is required to read in the text from the file, and output this to the user in a <prompt> element (lines 4 to 6). The main construct is a boolean <field>

```
1.     res = contentManager.apiRequest(paramPath, paramValue,
       feedNav.SelectSingleNode("feed/api/request/schema").Value.ToString
       , feedNav.SelectSingleNode("feed/api/request/url").Value.ToString)
2.     apiResponse.Load(res)
3.     Dim list As String =
       feedNav.SelectSingleNode("feed/api/response/parameter[@name='list'
       ]")
4.     Dim xmlIT As System.Xml.XPath.XPathNodeIterator =
       responseNav.Select(list)
5.     For count = 1 To Request.QueryString.Item("position") + 3 - 1
 6.     xmlIT.MoveNext()
7.     Next
8.     While xmlIT.MoveNext
   9.  If count > 3 Then
     10. Exit While
   11. Else
     12. responseString = responseString + "Result " + (count +
           Request.QueryString.Item("position") - 1).ToString +
           xmlIT.Current.Evaluate("string(normalize-
           space(string(translate(.,'£$*%^&*()!@:;<>=-
           _#~`¬¦€|\/',''))))").ToString
     13. xmlIT.MoveNext()
     14. count = count + 1
   15. End If
16.    End While
17.    Response.Write("There are " + xmlIT.Count.ToString + " results")
18.    Response.Write(responseString)
19.    Response.Write("Please say the result number, repeat, next, back,
       or main menu.</prompt>")
20.    Response.Write("<filled>")
21.    Response.Write("<submit
       next='http://localhost:9990/collectPersonalDetails.vxml'/>")
22.    Response.Write("</filled>")
```

Code Excerpt 25: Outputting API Results To user

element (lines 2) that accepts variations of ‘yes’ or ‘no’ answers from the user. Based on this input execution takes one of two paths: if the user does not confirm the recognition (line 9), control is passed back to apiCollectRequest.aspx to ask for the parameter’s value once more; or, if the user does confirm the input, then the ‘position’ variable is incremented by 1 before control is passed back also to apiCollectRequest.aspx, which will then ask for the next parameter’s value from the user.

Once all the parameters from the API specification have been elicited from the user and confirmed, control is passed to apiResponse.aspx which performs the HTTP Request

and outputs the response to the user (Code Excerpt 25). Line 1 calls the function apiRequest, part of the Content Manager which accepts four arguments, the first two being arrays of parameters paths and values, the third containing the API schema to be submitted, and the fourth containing the URL to which the API request is made. The parameters paths are fetched from the API's specification and the array of values uttered by the user are inserted into the appropriate place in the schema, using the array of paths as a direction.

Once the request has been made, the filename of the XML results file is stored in the variable 'res'. This XML file is then loaded into VoiceBrowse (line 2), and the results are extracted from the XML (line 4) by using the appropriate path that was specified with the API specification (line 3). The result is then an XML Node Iterator of API results, which can be output to the user. This is currently done three at a time, controlled by the use of a 'position' variable to store the current position in the list that is incremented with each result that is output. To set the XML pointer at the right place in the list, the For loop specified in lines 5 - 7 increments the XML pointer from the start of the list to the required position.

It is the While loop from line 8 - 16 that iterates through the next three results in the list and creates the String object to be output to the user. The If statement (line 9) prevents more than three results being output, otherwise XPATH is used to evaluate the result of normalisation() and translate() functions based on the result node's value, to remove space and erroneous characters from the result (line 12). The value is added to the 'responseString' variable that also contains the result number for that value. The end result of this loop is a variable containing 3 results and their results number, such as "Result 4, British Airways, Result 5, Iberia, Result 6, easyJet" for example.

The output itself begins in line 17 with the total number of API results uttered to the user, followed by the output of the 'responseString' variable (line 18), and lastly the instructions guiding the user to an appropriate response (line19). Not shown in Code Excerpt 25 is a VoiceXML <if> statement in the <filled> block that allows the user to hear the results again by using a <clear> element to delete the user's utterance from the <field>, or allowing the user to proceed to the next three results, in which case control is once again submitted to apiResponse.aspx with the incremented 'position' variable. Otherwise, if the user utters a result number, control is passed to the collectPersonalDetails.vxml form to collect the user's details for completion of the task.

7.9 VoiceBrowse Implementation: Challenges and Issues Encountered

Throughout implementation, numerous challenges were encountered that had to be overcome to enable VoiceBrowse maintain its generic nature. Problems arose from different areas of the VoiceBrowse environment and the technologies used for implementation; these can be categorised as Prophecy, VoiceXML, XPATH, Microsoft Speech Recogniser and APIs.

Common implementations of VoiceXML are based on Web Technologies and associated architectures, including the use of scripting languages to produce the VoiceXML during execution. With the choice of the Voxeo Prophecy platform, this paradigm is supported by the use of PHP, or other scripting languages with the incorporation of an appropriate web server, for example ASP.NET in the case of VoiceBrowse. When using web based technologies, standard techniques for passing variable values between pages include the use of 'session' variables – variables that hold their value whilst the current session, or interaction, is active.

The design of VoiceBrowse called for communication to occur between VoiceXML pages: passing the user's input from informationStart.aspx to disambiguate.aspx, for example; or passing the position variables from page to page as the user navigated their way through the document space. It had been planned to use session variables to accommodate these exchanges, however Voxeo Prophecy is currently unable to support the use of session variables in its current version. This is due to a new session being created with each new VoiceXML file that is accessed, and so at the point where execution has finished with one VoiceXML file and transitions to the next, all session variables are lost as the session itself has closed. Possible use of Voxeo Prophecy could be somewhat limited for developers needing to facilitate the use of session variables, and other session related attributes.

To overcome this limitation, it was possible to transmit variable values as query string variables, another common technique used in web based programming. Query string variables can often be found in the URL or Address bar of graphical browsers, appearing after the '?' character in an address string. For example in 'http://www.google.co.uk/search?q=vxml', the variable 'q', short for query, contains the value 'vxml' – here the query string variable 'q' contains the search query 'vxml' which is used by the 'search' page of www.google.co.uk.

A similar mechanism is available with Voxeo Prophecy. However this can lead to more complex and even confusing code; variables have to be requested from the query string to be used, and then VoiceXML variables containing the variable's value created, which are then submitted to the next document using the 'namelist' attribute of a <submit> element. If the number of variables to be passed as query string variables becomes relatively large, this can become quite a convoluted solution to the problem.

Problems with the VoiceXML language itself were also encountered - some due to the specification, and others due to incorporating web site information with VoiceXML. The biggest issue with regard to the VoiceXML specification was the lack of facility to save <record> audio data to a local sound file. A number of ad hoc solutions are available through the use of scripting languages, but a more desired solution would have this provision built into the VoiceXML language.

To overcome this problem in VoiceBrowse, and to recognise the words from the <record> element, a combination of PHP and VisualBasic.Net was used. By using the System.Speech 3.0 namespace, Visual Basic could be used to invoke the 'dictation' grammar on the saved .wav file using Microsoft Speech Recognition Engine 5.1, and the results then saved as a text file. Additionally, due to the nature of VoiceBrowse's functionality, a rule based method was needed to add new words, in particular proper nouns, to the dictation grammar[46]. The ability to do this however is not available in the current implementation of the System.Speech 3.0 namespace and Microsoft Speech Recognition Engine 5.1, and so a workaround had to be devised. This included invoking two grammars at the same time on the saved audio file; one being the dictation grammar, and the other being a finite state based grammar containing a list of <item> elements, each of which contained a particular word not in the current dictation grammar. Although this solved the original problem, it led to high word error rates, and so there is a need for a method to produce N-Gram based language models that can be

[46] http://msdn.microsoft.com/en-us/library/system.speech.recognition.dictationgrammar.aspx

used with the Microsoft Speech Recogniser, similar to that of other recognisers available with appropriate language model tools[47].

Further constraints were encountered with regard to the VoiceXML grammar capabilities. Current specifications of the VoiceXML language limit language understanding to finite state grammars. Using VoiceXML with statistical language models, such as N-Gram models, is not currently incorporated into the specification, although this can be made possible with the use of scripting languages and certain speech recognisers (Larson 2005b). This limits the possible use of VoiceXML to dialogue based on system initiative, reducing the potential application and deployment of a standard language for creating dialogue systems.

Although some shortcomings of the VoiceXML language have been identified, it is however a mature and well developed language in other areas. Its elements and execution algorithms are well specified and documented. Normally handwritten by developers, <grammar> and <prompt> elements are therefore created using legal characters with regard to the VoiceXML specification. VoiceBrowse however extracts the information from online sources for use within the <prompt> and <grammar> elements. This led to the problem of illegal characters being used as the content for these elements, such as '<' and '(' for example. A rudimentary solution developed to prevent this was to 'filter' the words to be used, removing any illegal characters as defined by the VoiceXML language.

One other language used during implementation with restrictions that had to be overcome was the XPATH query language. XPATH was developed alongside XML

[47] http://cmusphinx.sourceforge.net/html/cmusphinx.php & https://cmusphinx.svn.sourceforge.net/svnroot/cmusphinx/trunk/SimpleLM/

and there are numerous functions and operations available for XPATH to query an XML document.[48] Planned use of the contains() string function had been during the disambiguate.aspx state, to match the user's input onto available feed titles. However, this function matches exact phrases, meaning that to identify a particular feed, a user would be required to say the words of the feed's title in exact order – the user query 'Northern Ireland News' would not be matched to 'BBC News Northern Ireland' in the feed list for example. Suggestions for future implementations of the XPATH language would be to allow further refinements of string functions such as contains() with common options available in many search functions, such as 'Match EXACT phrase' or 'Match ANY words'.

An anticipated problem was the inclusion of APIs in the VoiceBrowse environment to handle the task based dialogues. Due to each API having its own specification, and handling different content types, they cannot be treated generically, and so in this implementation of VoiceBrowse the developer is required to specify certain parameters when adding an API into the system. Furthermore, regarding the output of API responses, it is clear that different content types should be handled differently, although knowledge of this from VoiceBrowse's perspective is somewhat limited due to the generic nature of the interaction. Future work in this area could see the automation of adding APIs to the environment, coupled with better language understanding and grammar creation abilities. However, the generic preparation of different outputs will be evaluated from a user's point of view in the next chapter.

---

[48] http://www.w3.org/TR/xpath-functions

7.10 VoiceBrowse Implementation: Example Dialogues

The realisation of the VoiceBrowse architecture has resulted in a dialogue system capable of generically browsing online content. To illustrate the functionality and dialogues that can be achieved with VoiceBrowse, Table 7.6 overleaf introduces three sample dialogues with the system, including comments to aid understanding of the functionality.

Utterance	Utterance	Comment
S1	Welcome to VoiceBrowse. To return to this point, say main menu at anytime.	Welcome prompt
S2	Say help for assisstance. Please say the provider, and also the category if you wish, or if you want to know what is available, say list.	Initial direction prompt
U1	BBC	User wishes to access information from the BBC website
S3	You have requested BBC. I have found 10 different feeds. Options include BBC News, Business, Entertainment, Sports, Football, Northern Ireland, Politics, Home Page...	As a result of matching the user input 'BBC' to the Feed Titles, 10 possible feeds were found. Disambiguate.aspx outputs these to the user, and awaits a response.
U2	Entertainment	The user requests Entertainment. The query now actually reads BBC Entertainment.

S4	There are 23 stories. I will read them 3 at a time. Say the story number to access the full story, or say repeat, next or back.	Disambiguate.aspx now finds a unique feed when matching 'BBC Entertainment' to the Feed Titles. The controls available to the user to navigate through the resulting documents are iterated to the user
S5	Story 1, talksport host james whale is fired for telling listeners to vote for london mayoral candidate boris johnson. Story 2, babyshambles frontman pete doherty is released from prison after serving 29 days of a 14 week sentence. Story 3, hollywood actor robert downey junior signs up for the film sequel to iron man, which is to be released in 2010.	The first 3 <description> elements of the BBC Entertainment RSS Feed are output by listContentItems.aspx and the system awaits a response from the user.
U3	3	The user responds with number '3', indicating that they wish VoiceBrowse to access the full web page related to story 3.

S6	Ok, you want story 3, one moment	Acknowledgement from VoiceBrowse to indicate that they have understood the user's request At this point the Content Manager has received a <source> URL from the Dialogue Manager, and downloads and 'tidies' the HTML from that address. The resulting HTML is saved as an XML file, which the Dialogue Manager will use to output to the user.
S7	You can say repeat, next or back at anytime	Standard introduction outputting applicable navigational controls to the user whilst the main story is being output.
S8	Hollywood actor... (story narrated from web site)	The Dialogue Manager outputs 3 <p> elements at a time to the user, retrieved from the resulting XML document from system utterance S6
S9	Welcome to VoiceBrowse. To return to this point, say main menu at any time.	Welcome prompt

S10	What is your query?	This is the open version of VoiceBrowse. For illustration purposes, the same task as in the above sample dialogue will be completed.
U4	Is Robert Downey Junior doing another Iron Man movie?	An open ended question from the user regarding the same story as above. The users utterance is captured as a VoiceXML <record> element, which is then stored as a .wav file by the PHP script. The Microsoft ASR is then used to convert the audio to text, which is then output to and confirmed by the user (not shown here).
S11	I have found 4 stories.	The Dialogue Manager here removes the stop words ‘is’, ‘doing’, ‘another’ and leaves the key terms of the query. The query is passed to the Content Manager which performs the COSIM function. The result is a ranked list of 4 documents, or <description> elements.
S12	Story 1, hollywood actor robert downey junior signs up for the film sequel to iron man,...	The Dialogue Manager reads the resulting <description> elements to the user, 3 at a time, in ranked order.

U5	1	The user has barged-in and has requested more information about the first story.
S13	You can say repeat, next or back at anytime	See S7
S14	Hollywood actor... (story narrated from web site)	See S8
S15	Welcome to VoiceBrowse. To return to this point, say main menu at anytime.	Welcome prompt
S16	Say help for assisstance. Please say the provider, and also the category if you wish, or if you want to know what is available, say list.	Initital direction prompt
U6	Travel Fusion	In this example, the user wishes to complete a task based dialogue, specifically book a flight by accessing the 'Travel Fusion' website.
S17	Ok, I have found Travel Fusion.	Disambiguate.aspx matches the user's input against the Feed Titles, finds only 1 unique feed, and so can proceed.

S18	Please tell me the origin.	The Content Manager, as a result of finding only 1 unique feed, has identified the feed as an API by the inclusion of its <type> element. Therefore, it accesses the named parameters from the API, in this case four parameters, and passes this list back to the Dialogue Manager to collect the information from the user. The first parameter is to collect the origin from the user.
U7	Belfast	The user's utterance is captured as a VoiceXML <record> element, which is then stored as a .wav file by the PHP script. The Microsoft ASR is then used to convert the audio to text, which is then output to and confirmed by the user (not shown here).
S19	Please tell me the destination.	Second parameter to be collected from the user
U8	London	See U7
S20	Second parameter to be collected from the user	Third parameter to be collected from the user
U9	26th August 2008	See U7

S21	Please tell me the return date	Fourth parameter to be collected from the user
U10	28th August 2008	See U8
S22	Ok one second. There are 62 results	With all parameters collected, they are passed back to the Content Manager which inserts them into their respective place in the API schema. This is then submitted to the relevant <URL> for that API. The response is received as an XML document, and it is passed to the Dialogue Manager for output to the user.
S23	Result 1 British Airways. Result 2 Aer Lingus. Result 3 EasyJet. Please say the result number, next, repeat or back.	Using the <response> path specified with the API, the Dialogue Manager can extract the relevant results from the API response, which are output to the user 3 at a time.

Table 7.6: Sample Dialogues With VoiceBrowse

7.11 Summary

This chapter has examined and explored in detail the implementation phase of VoiceBrowse. The conceptual architecture, design, process and use case diagrams presented previously were transformed into a VoiceXML Call Flow Diagram. Available dialogue technologies and platforms were considered for each of the main components of a spoken dialogue system, and the technologies chosen to realise VoiceBrowse were discussed along with the rationale for their selection.

Incremental implementation of the system was then discussed, and a detailed account of the various stages presented. Challenges and issues faced during implementation have been documented, including suggestions for overcoming current shortcomings with regard to the technologies. The next chapter describes the evaluation of the VoiceBrowse system.

Chapter 8: VoiceBrowse Evaluation

With VoiceBrowse implemented and tested, the next step of the research was to evaluate the system. As defined at the requirements stage, VoiceBrowse offers contributions to two different aspects of dialogue research: technical advances with regard to generic dialogue systems utilising unstructured online content in multiple domains; and usability advances with regard to browsing the Internet through speech.

Evaluation of both these aspects was taken into consideration during the design of the evaluation, in order to identify relations between the two. A detailed discussion of the evaluation design follows, followed by a presentation and discussion of the results and findings.

8.1 VoiceBrowse Evaluation: Design

The evaluation process should measure the performance and efficiency of the implemented system against the original requirements, and both qualitative and quantitative mechanisms were used to achieve this.

To evaluate the effect of the two different dialogue strategies on dialogue usability, it was initially proposed to present both systems to each user, and compare and contrast the user's subjective measurements given for each system. However, referring to the definition of usability given in Section 2.8 of the thesis, it would be difficult to produce a measure of learnability for either system. Learnability refers to the ease of learning the system and its functionality by a user over a period of time, and as each user would only have limited time with each system, this could not be measured accurately.

Therefore in a revised test design, half the participants were presented with the closed version of VoiceBrowse, and half with the open version. To simulate an extended period of time within which learnability could be measured, the evaluation was done in three stages:

1. The user's performance with the system was measured whilst completing set tasks without having used the system before commencement and without explanatory sessions.
2. The user was offered some free time to interact with VoiceBrowse, and the opportunity to ask questions regarding the system's functionality.
3. The user would then complete similar tasks as they faced in stage 1, and the difference in performance was calculated to give a measurement of learnability.

Furthermore, as usability is also defined with respect to the needs of different users, the evaluation design caters for the needs of the different users with regard to age and computing experience. The latter is important with regard to the usability of VoiceBrowse as it is the functionality of the graphical web browser that VoiceBrowse is designed to replicate, and so the experience of using such an interface would be predicted to have an effect on the usability of VoiceBrowse.

The final evaluation design is included as Appendix B of the thesis and its design is reinforced by its use in previous dialogue evaluations. Den Os et al. (2005) used a similar evaluation design to evaluate the usability of two different dialogue implementations of a multimodal dialogue system for bathroom design.

The hypotheses that the evaluation was designed to test can be summarised as follows:

1 That open and closed dialogue strategies will have an effect on a user's performance and usability with VoiceBrowse. It is thought that an open dialogue strategy would be more suited to younger people than older people due to the flexibility and speed offered over a closed strategy - and that a closed dialogue strategy would be more suitable to older people than younger people due to more guidance being offered through the dialogue to task completion. Throughout the evaluation, the closed and open version of VoiceBrowse will be referred to as *'System 1'* and *'System 2'* respectively.

2 That prior use of graphical interfaces will have an effect on the user's judgement of usability and performance using VoiceBrowse. It is thought that those who are experienced with regard to web browsing will find VoiceBrowse more usable than those who are inexperienced, due to their prior knowledge of web technologies. The two user groups tested in this hypothesis will be referred to as '*Experienced'* and '*Inexperienced'*.

3 That age will have an effect on usability and performance. It is thought that young people will find VoiceBrowse more usable than older people, due to their heightened appreciation of technology. The two user groups tested in this hypothesis will be referred to as '*Young'* and '*Old'*.

4 That usability is directly proportional to experience with VoiceBrowse – that is, as users interact with VoiceBrowse over time, the usability of the system increases. This will be referred to as learnability, and it is expected that experienced users and younger users demonstrate a higher degree of learning than inexperienced and older users. The two user groups tested in this hypothesis will be referred to as '*Untrained'* and '*Trained'*.

5 That browsing the web generically through voice can be made possible through the use of encapsulated RSS and API feeds.

The term ‘usability’ in hypotheses 1 – 4 refers to the ease and satisfaction with which the user interacts with VoiceBrowse.

The same evaluation was given to all 32 participants. This is the recommended minimum number with which hypothesis testing can be performed (Kraemer & Thiemann 1987). The evaluation itself aimed to be representative of normal VoiceBrowse usage, and so two Scenarios of six Tasks were devised to be given to the participants. Scenario 1 was given to the users during the first part of the evaluation, and Scenario 2 was given after the period of free time with the system. The 6 Tasks in both Scenario 1 and 2 are similar in order and complexity, so that comparison between Scenario 1 and 2 can be made fairly. Furthermore scenarios were presented alternatively to each participant, so that half received scenario 1 first and half received scenario 2 first.

The evaluation Scenarios and schedule are also included as Appendix B. Included also is the questionnaire that was used after both scenarios, devised using the SASSI questionnaire as a guideline (Hone & Graham 2000, 2001). As the same questionnaire was used after each Scenario, any difference in answer should reflect the change of user opinion after becoming more familiar with the system, providing a means to obtain a quantitative measure of learnability. Dialogues were also recorded and then transcribed in XML format, annotated with standard interaction parameters such as dialogue length, prompt length etc., allowing additional quantitative analysis of the data.

Finally, due to the poor performance of the Microsoft Speech Recogniser demonstrated in the original tests (see Section 7.9), it was decided to simulate speech recognition in

the open version of VoiceBrowse during the initial open ended prompt; achieved by passing hand coded inputs for each scenario to the Content Manager instead of the speech recognition result. This was perceived not to be a cause for concern, as the evaluation was testing hypotheses related to the usability and functionality of dialogue components, and not the speech recognition performance. As dialogue executions continue in the same way after the initial prompt for both versions, a fair comparison can still be made by simulating the speech recognition in this initial phase of one system.

8.2 VoiceBrowse Evaluation: Results

The questionnaires utilised were derived from the SASSI questionnaire (Hone & Graham 2000, 2001), which also defines 5 categories of usability metrics from the questions asked: Efficiency, Annoyance, Cognitive Demand, Likeability and Accuracy. These are used to analyse the evaluation results, and Table 8.1 overleaf defines these categories in terms of question numbers from the questionnaire used.

Usability Aspect	Questionnaire Numbers
Efficiency	5.4 (reverse polling), 5.5, 5.6
Annoyance	6.6, 6.7, 6.8, 7.4
Cognitive Demand	6.2, 6.3 (reverse polling), 7.1 (reverse polling)
Likeability	4.2, 4.5, 5.3, 6.1, 6.4, 7.2, 7.7
Accuracy	2.1 (reverse polling), 2.4, 2.5 (reverse polling), 4.3 (reverse polling), 4.4 (reverse polling)

Table 8.1: Usability Category Definitions

Questions used for efficiency ratings are concerned with the length of the dialogue, the speed of the dialogue, and if interactions quickly lead to successful goal completion. A higher rating reflects that the user decided the dialogue system was quite efficient in completing the task.

The annoyance rating will be used to give an impression of how displeasing the user found the interaction, using questions that ask if the dialogue was boring, repetitive, frustrating and difficult to use. A lower rating here is better, indicating that the user did not find the interaction annoying.

Cognitive demand is a measure of how much concentration was required of the user when interacting with the system. This metric is calculated from questions relating to the state of relaxation of the user, the amount of concentration required during interaction and if the user had found the system difficult to use. A higher rating is better than a lower rating due to reverse polling i.e., a higher rating means the user was more satisfied with the amount of attention involved.

Likeability is concerned with the overall impression of contentment with the system. Questions used for its definitions include the friendliness and pleasantness of the system, error recovery, the degree of fun of the interaction, that they felt in control of the dialogue, that it was easy to learn and that they would use it again in the future. A higher rating of likeability is better than a lower one, indicating that the user likes the system more.

Lastly the accuracy rating is concerned with the response of the system to the user's commands. Questions regarding the actions taken after user's speech, the reliability of the system, if the system did what the user expected and if it made a lot of errors, are

used to provide a response accuracy rating. Again a higher rating reflects that that the user was more pleased with the response of the system.

To quantify the qualitative data, a metric scale from 1 to 5 was used to reflect the user's answer, 1 being the poorest answer available and 5 being the best answer available. The final set of results recorded from the user questionnaires and VoiceBrowse log files is too large to be included here, and so has been attached as Appendix C. However, for illustration and discussion purposes, the main findings are highlighted overleaf in Table 8.2 which presents a summary of the qualitative data used for usability analysis.

Table 8.2 shows firstly the overall comparisons of the Untrained and Trained scenarios for both Systems 1 and 2. Comparisons of the five usability aspects with respect to experience level a3nd age are then shown. With regard to experience, 'E' refers to the Experienced user group, and 'I' refers to the Inexperienced User group, and shows the data from left to right for Untrained Experienced Users System 1, Untrained Inexperienced Users System 1, Trained Experienced Users System 1, Trained Inexperienced Users System 1 and so on. Older and Younger user groups are differentiated as 'O' and 'Y' respectively, and a similar presentation is used. Additionally Table 8.3 presents a summary of the quantitative data, which is used for performance analysis.

To help illustrate findings, 'box and whisker' graphs were constructed from the above tables – the raw data itself from the main results was used to construct the graphs. Figures 8.1 – 8.23 show the distribution of the data for the five usability aspects under investigation, comparing each system overall, by age of user and then by experience of user. Comparisons of the overall interaction parameters between the two systems are also included (Figure 8.8).

	System 1				System 2			
	Untrained		**Trained**		**Untrained**		**Trained**	
	Overall							
	3.33		3.42		3.41		3.39	
Efficiency	3.67		3.75		3.77		3.79	
Annoyance	2.51		2.39		2.35		2.18	
Cognitive Demand	3.25		3.45		3.35		3.61	
Likeability	3.76		4.01		3.63		3.80	
Accuracy	3.31		3.58		3.80		3.83	
	Experience							
	E	I	E	I	E	I	E	I
Efficiency	4.04	3.29	4.00	3.50	3.88	3.67	3.83	3.75
Annoyance	2.38	2.66	2.09	2.68	2.40	2.31	2.31	2.06
Cognitive Demand	3.54	2.96	3.79	3.13	3.41	3.29	3.79	3.83
Likeability	4.09	3.43	4.23	3.79	3.61	3.66	3.70	3.91
Accuracy	3.33	3.30	3.60	3.56	3.68	3.93	3.75	3.93
	Age							
	O	Y	O	Y	O	Y	O	Y
Efficiency	3.75	3.58	3.88	3.63	3.75	3.79	3.88	3.71
Annoyance	2.34	2.69	2.00	2.78	2.56	2.16	2.19	2.19
Cognitive Demand	3.25	3.25	3.50	3.41	2.96	3.75	3.54	4.08
Likeability	3.79	3.73	4.09	3.93	3.43	3.84	3.77	3.84
Accuracy	3.60	3.03	3.96	3.23	3.55	4.05	3.85	3.83

Table 8.2: Questionnaire Results

	System 1		System 2	
	Untrained	**Trained**	**Untrained**	**Trained**
Scenario Duration	468322	366299	452438	286663
<noinputs>	2.063	0.438	0.938	1.88
Barge-Ins	6.625	8.375	1.375	1.688
Help Requests	1.063	0.125	0.000	0.000
New Query States	7.938	5.313	5.875	5.938
<nomatches>	2.438	1.000	0.188	0.063
Turns per Scenario	65.313	56.813	42.063	43.688

Table 8.3: Interaction Parameters

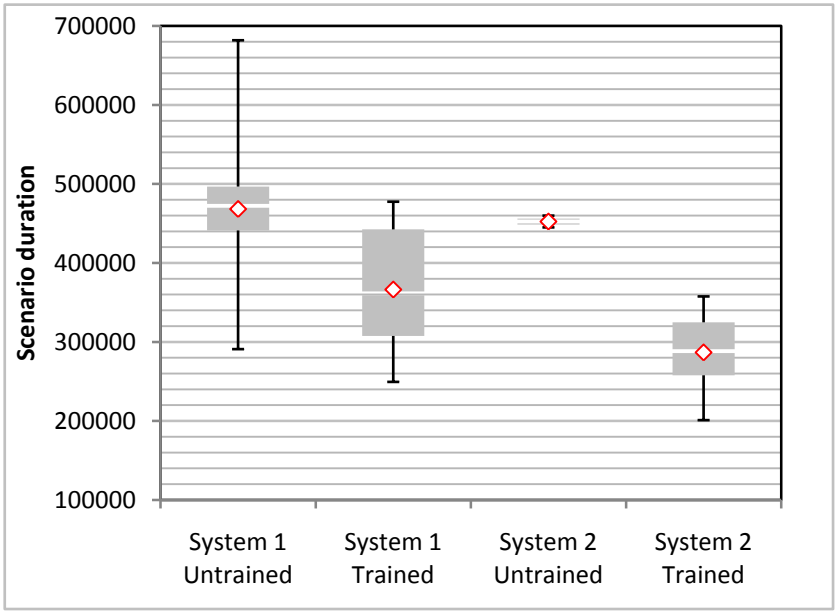


Figure 8.1: Scenario Duration w.r.t. User Group

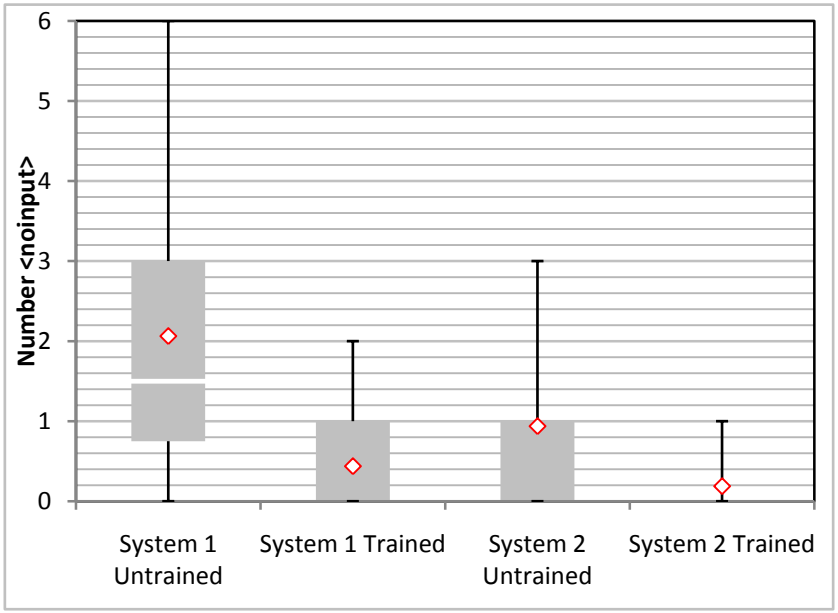


Figure 8.2: Number <noinput> w.r.t. User Group

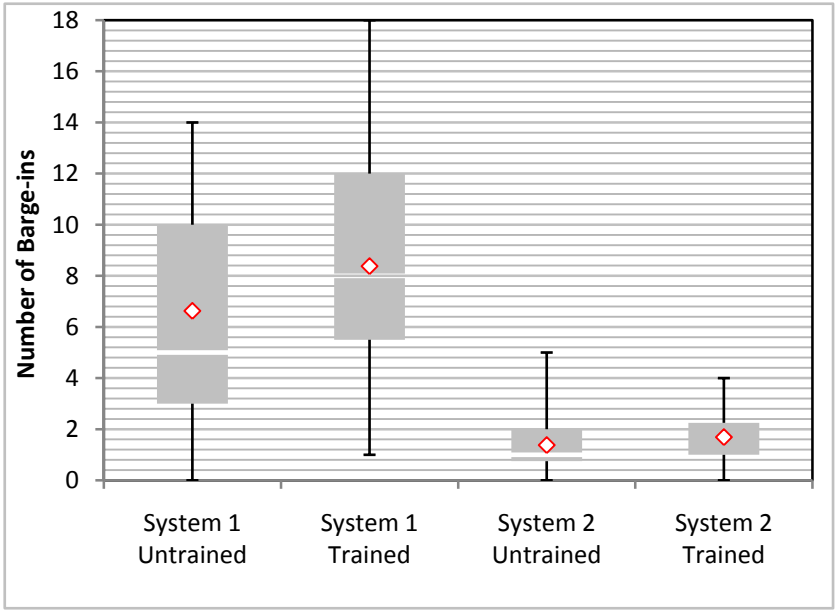


Figure 8.3: Number of Barge-Ins w.r.t. User Group

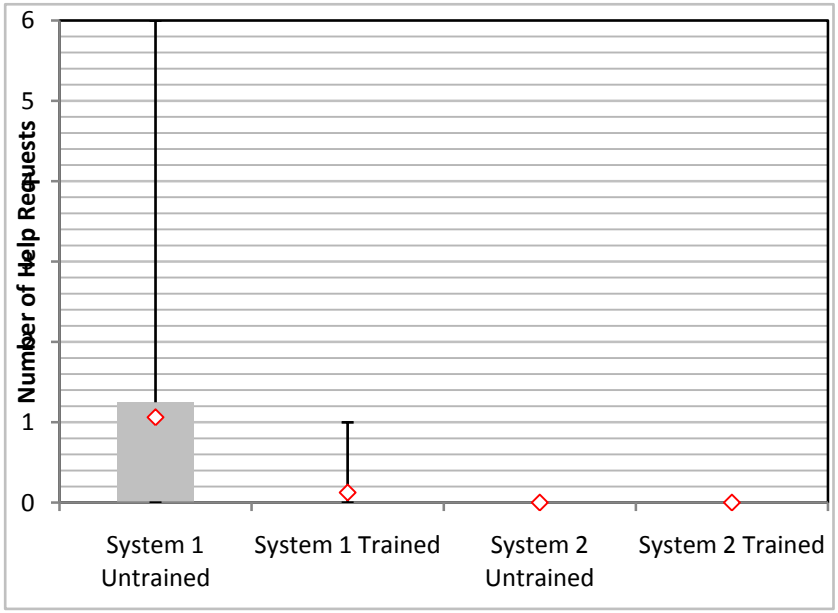


Figure 8.4: Number of Help Requests w.r.t. User Group

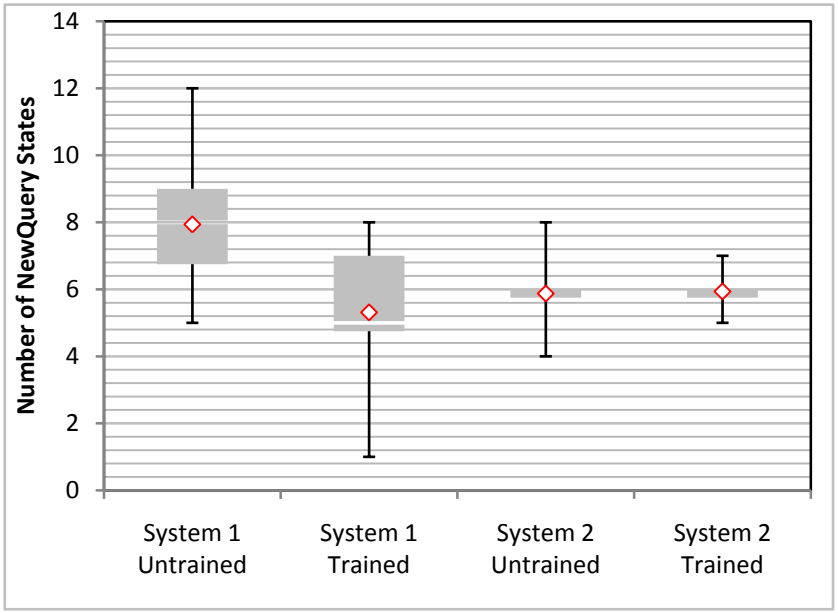


Figure 8.5: Number of New Queries w.r.t. User Group

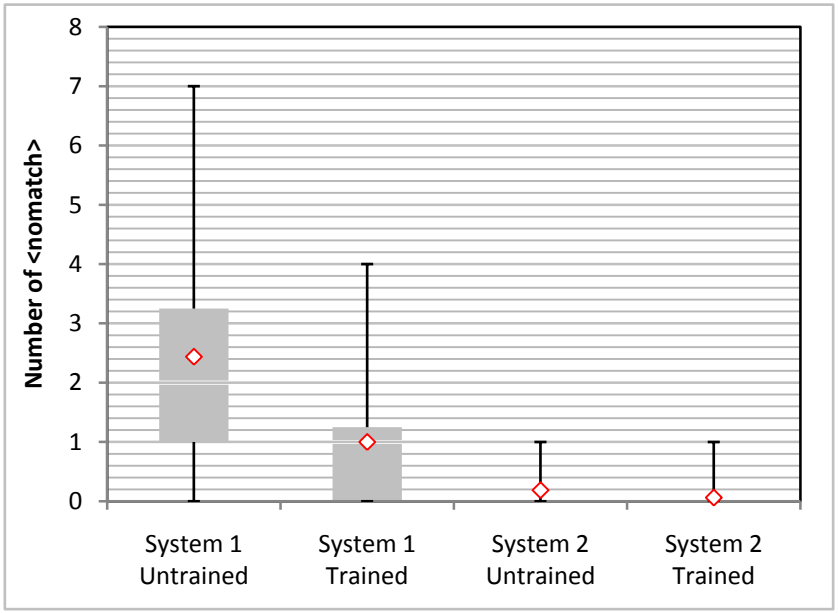


Figure 8.6: Number of <nomatch> w.r.t. User Group

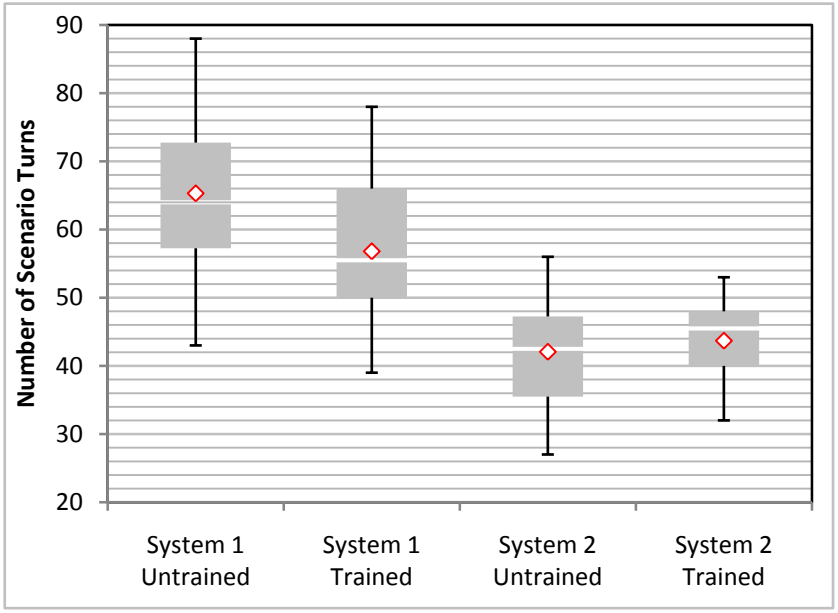


Figure 8.7: Number of Turns per Scenario w.r.t. User Group

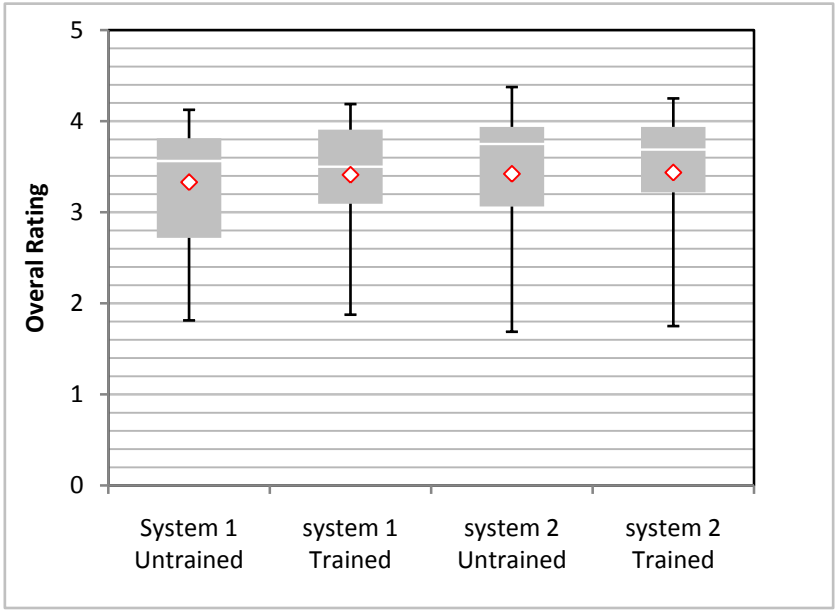


Figure 8.8: Overall Rating w.r.t. User Group

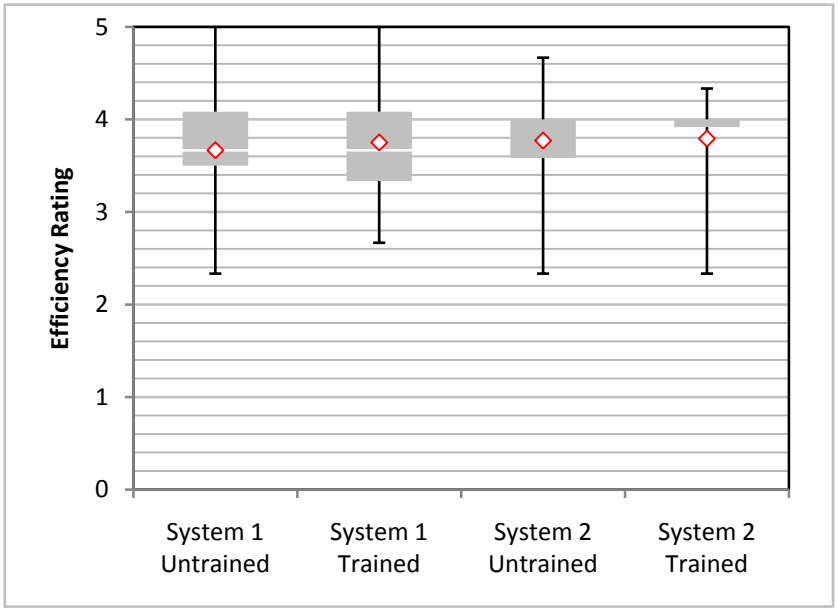


Figure 8.9: Efficiency Rating w.r.t. User Group

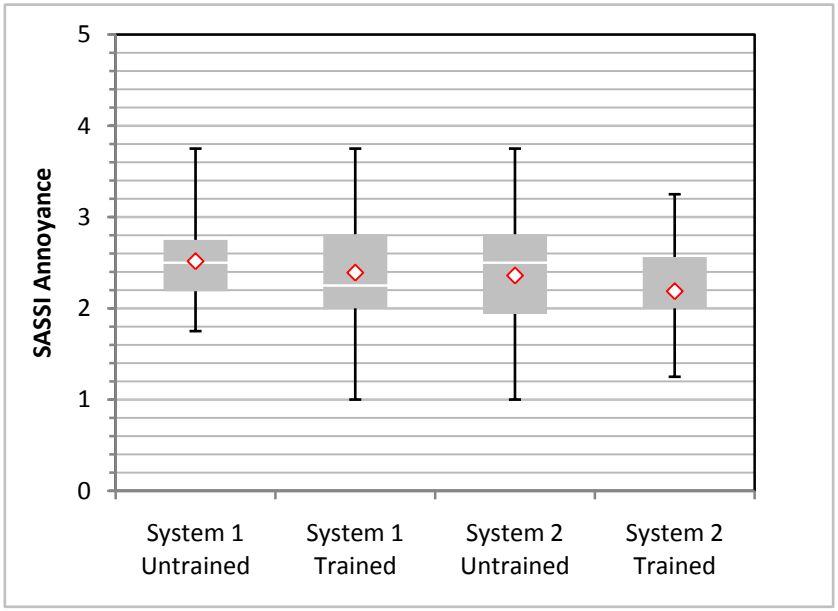


Figure 8.10: SASSI Annoyance w.r.t. User Group

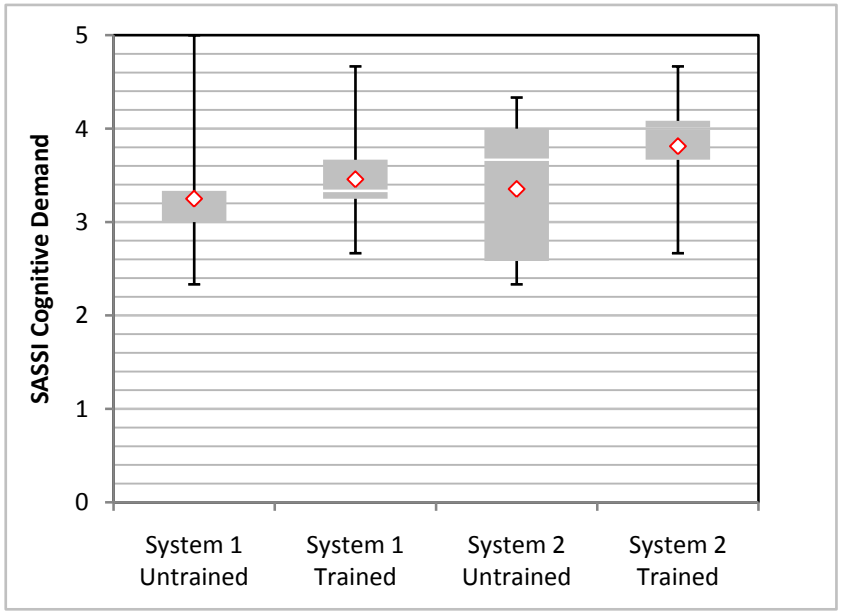


Figure 8.11: SASSI Cognitive Demand w.r.t. User Group

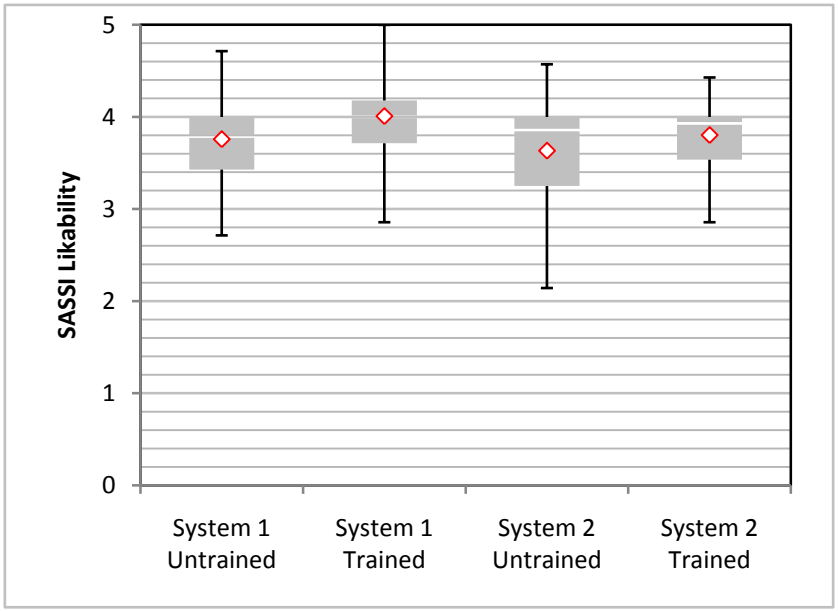


Figure 8.12: SASSI Likeability w.r.t. User Group

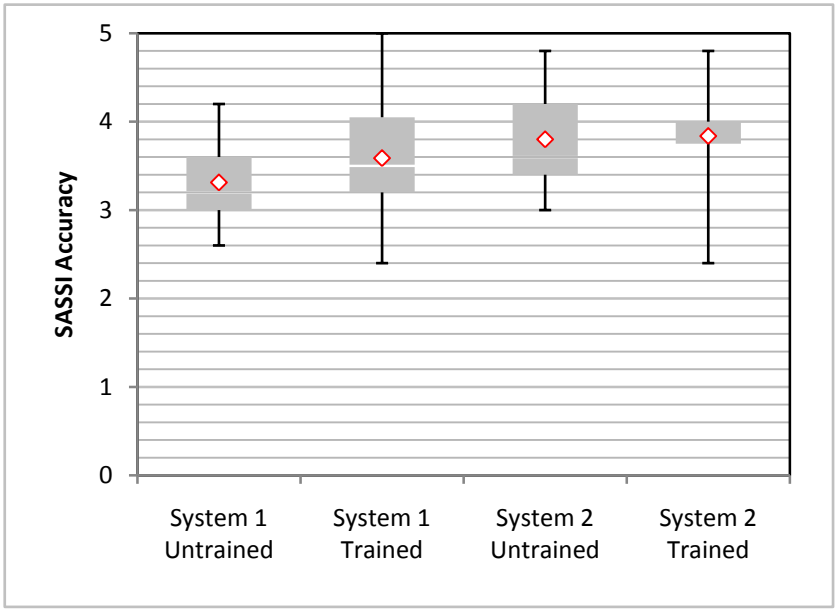


Figure 8.13: SASSI Accuracy w.r.t. User Group

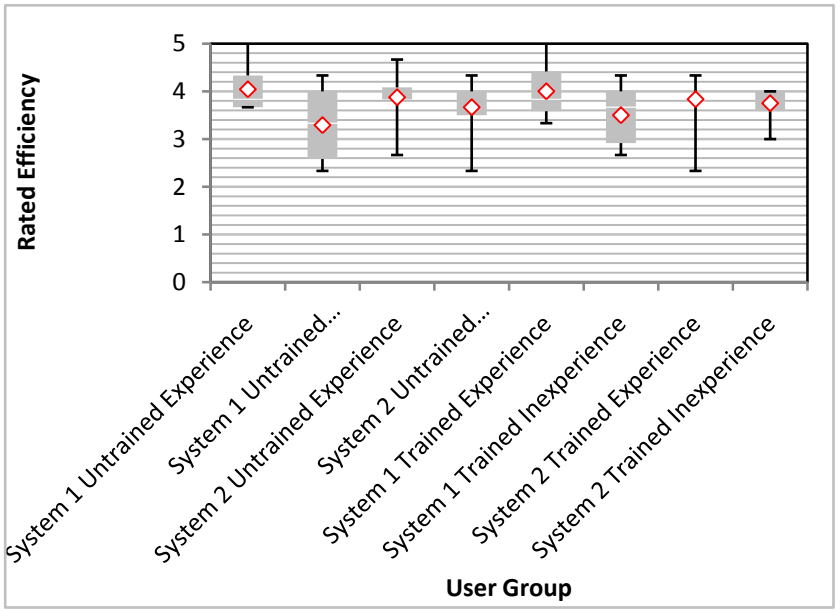


Figure 8.14: Efficiency Rating w.r.t. User Group

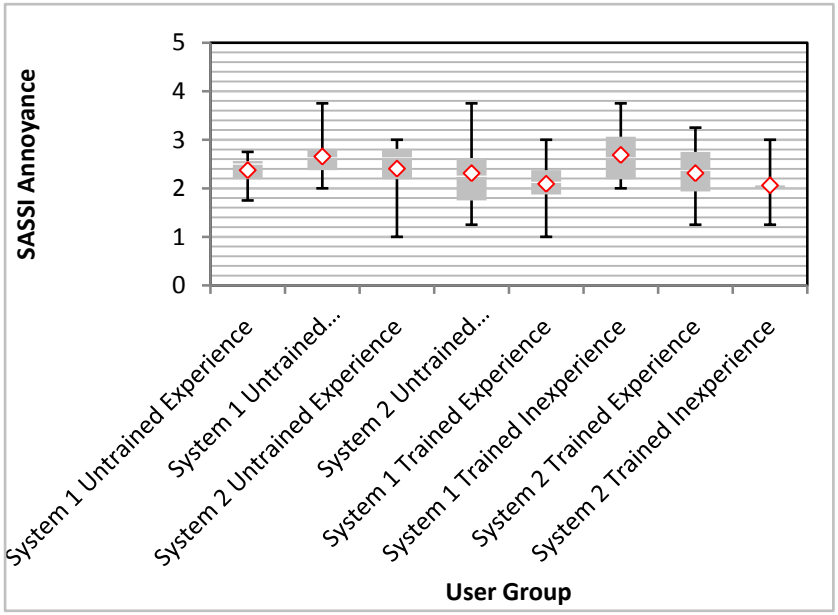


Figure 8.15: SASSI Annoyance w.r.t. User Group

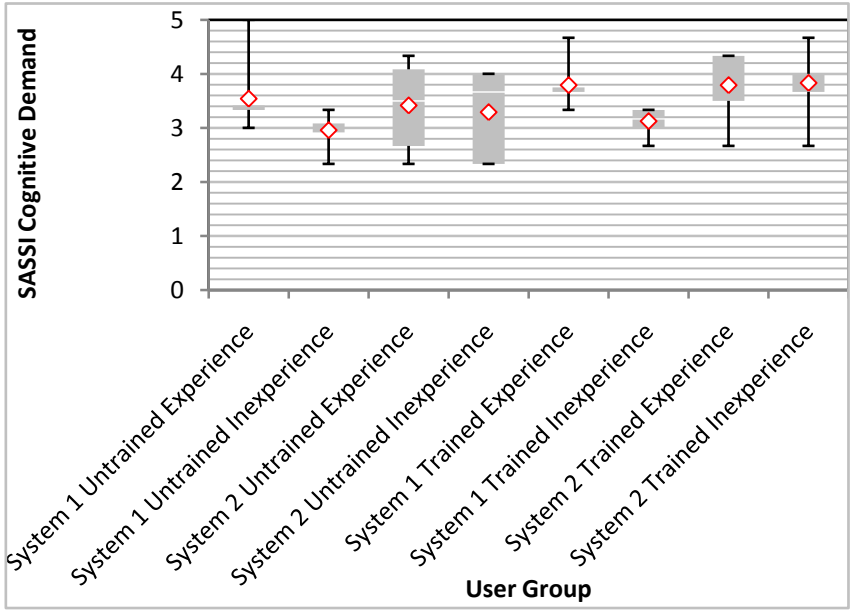


Figure 8.16: SASSI Cognitive Demand w.r.t. User Group

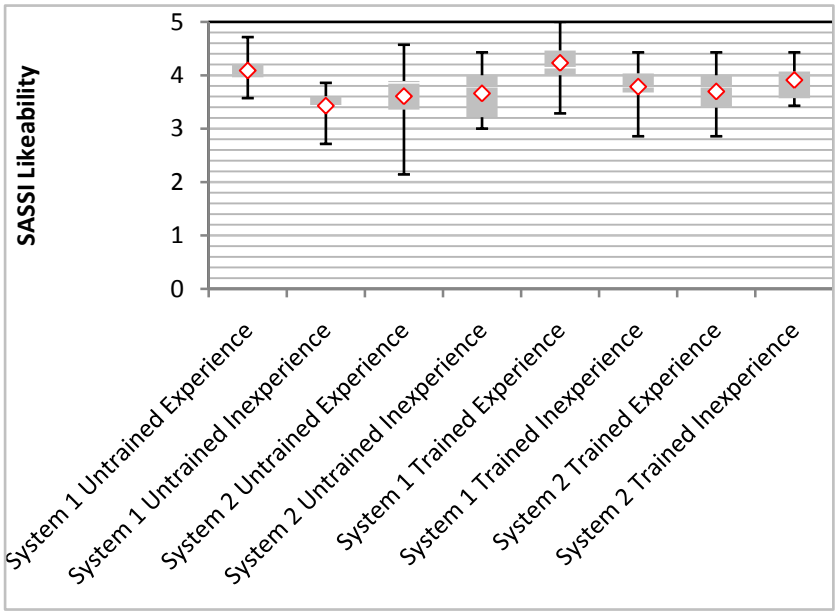


Figure 8.17: SASSI Likeability w.r.t. User Group

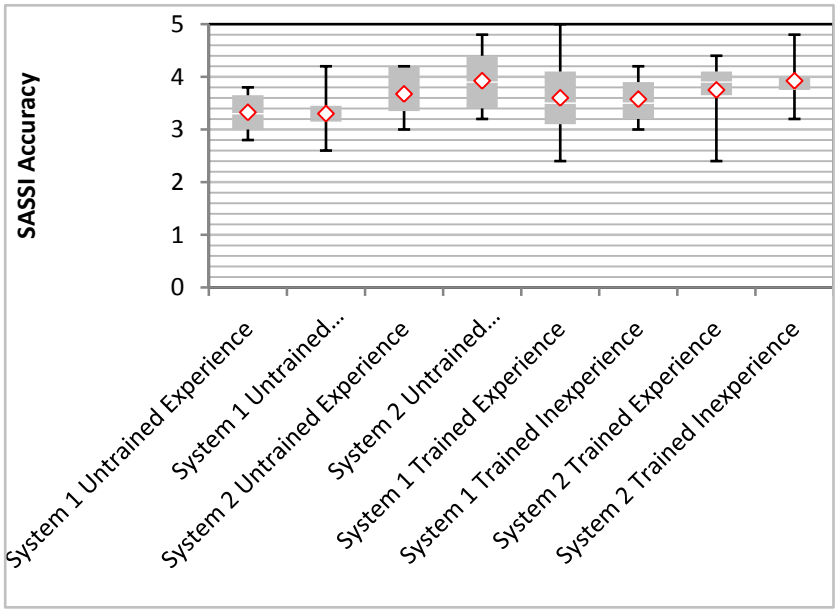


Figure 8.18: SASSI Accuracy w.r.t. User Group

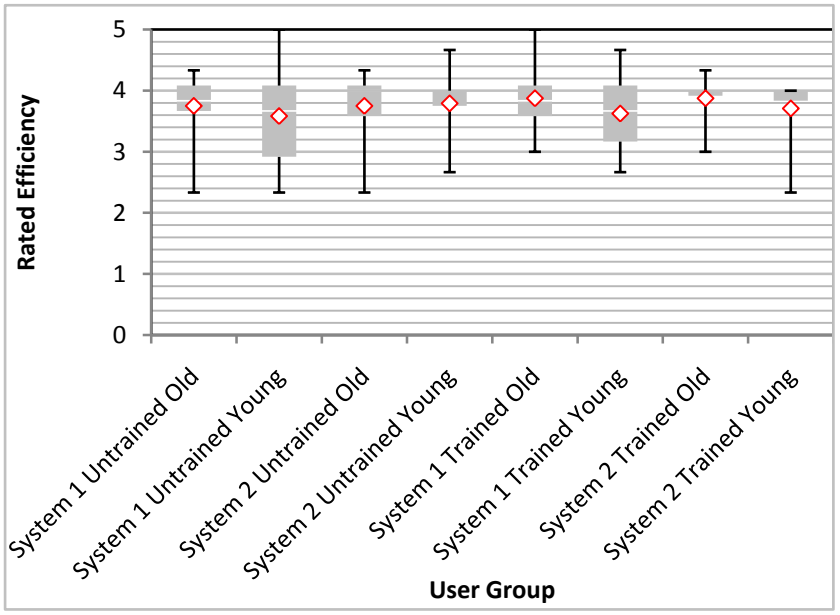


Figure 8.19: Efficiency Rating w.r.t. User Group

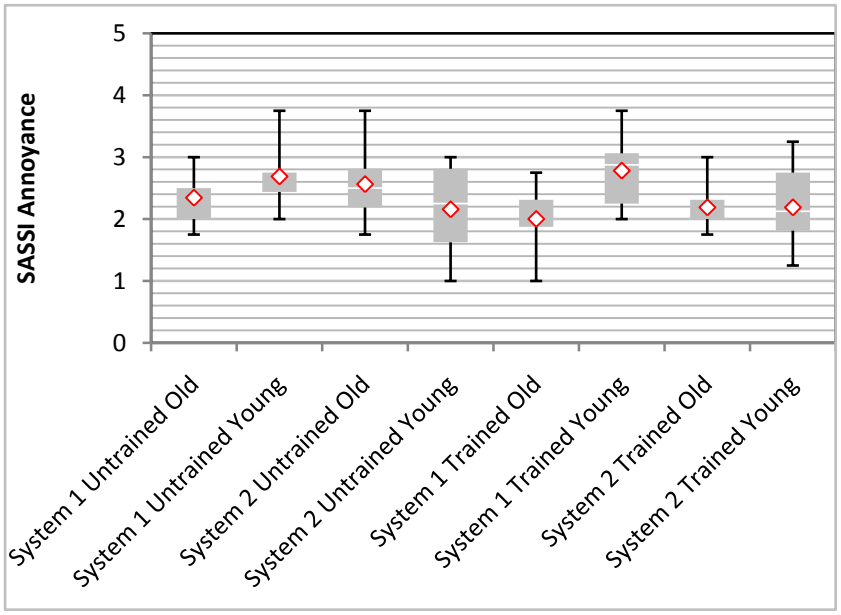


Figure 8.20: SASSI Annoyance w.r.t. User Group

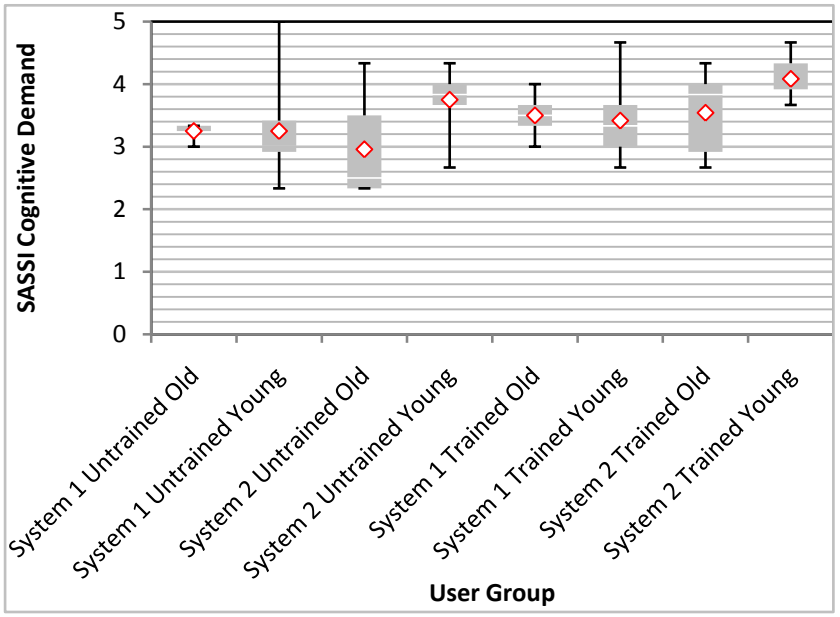


Figure 8.21: SASSI Cognitive Demand w.r.t. User Group

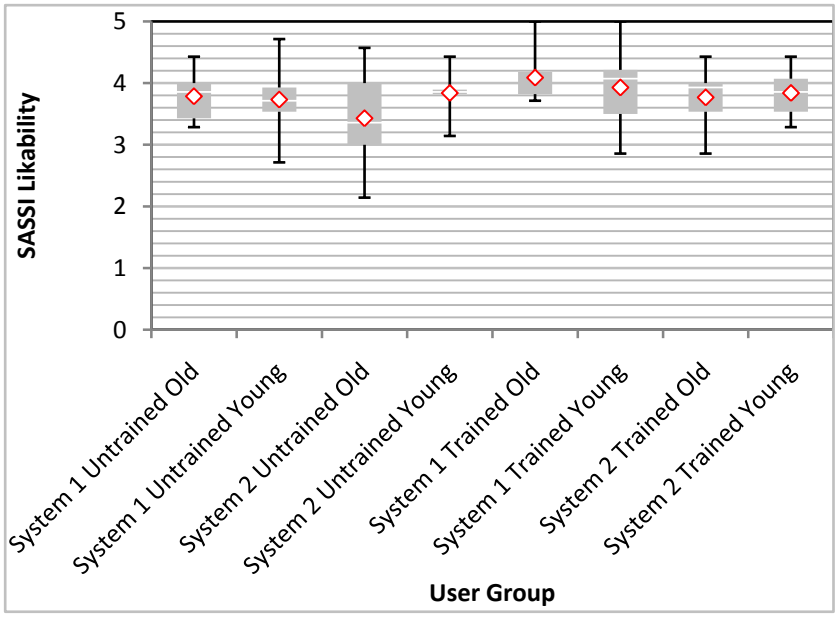


Figure 8.22: SASSI Likeability w.r.t. User Group

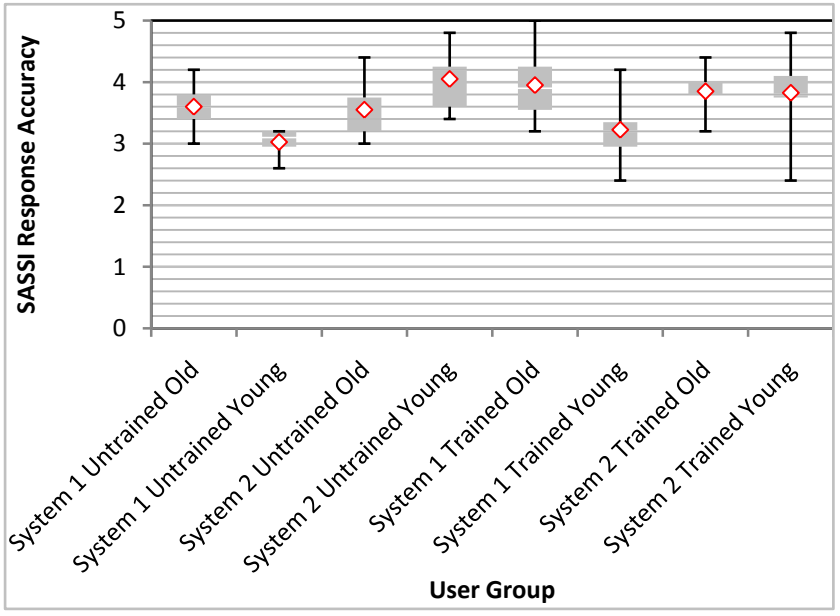


Figure 8.23: SASSI Response Accuracy w.r.t. User Group

Figure 8.1 shows the average duration with respect to the Trained and Untrained versions of Systems 1 and 2. It can be seen that training does have an effect on the scenario duration, particularly in System 1 where the distribution of durations is closer to the mean. As users became used to the interaction in both systems, tasks were therefore completed in a shorter time.

Figure 8.2 shows the number of <noinput> events for each of the Trained and Untrained versions of System 1 and 2. Once again training seems to have more of an effect on System 1, as the number of <noinput> events decreased dramatically in Scenario 2. During the evaluation it became clear that during initial interactions with both systems; users were often unclear at which points they could speak, and what they were required to say. As discussed later, the consequence of having the instruction prompts positioned before the main body of the prompt was that users were unclear that the system prompt had ended and that they had forgotten their options. Therefore, over time, as reflect on the Figure 8.2, the number of <noinputs> decreased dramatically due to the standard dialogue mechanisms being available and users learning these throughout the scenarios. Furthermore the number of barge-ins also increased after training for each system, shown in Figure 8.3. As users got to learn the available instructions in System 1 at each stage of the dialogue, they became increasingly confident with interrupting the system to progress the dialogue faster. Tangible examples of such a case include the listContentItems dialogue state – most users waited for the three content items to be output before speaking their desired story number to the system, whereas when their time with the system increased, users in general said the desired story number as it appeared in the output prompt.

This conclusion is reinforced by the decreasing number of help requests after training, shown in Figure 8.4. It was observed that, for users that requested use of the help facility early in the evaluation, they did not do so for the rest of the scenario, indicating both the effectiveness of the help function and the learnability of the system.

The number of 'NewQueryStates' is an important interaction parameter, as this indicates how troublesome users found the interaction. A 'NewQueryState' is entered at the beginning of each request for information, so if a user is having trouble interacting with the system, then the number of 'NewQueryStates' will be more than 1 as the system will be repeatedly asking the user for their information request. Consistent with the decrease of <noinputs> and <nomatches> shown in Figure 8.6, the decrease of 'NewQueryStates' is a further indication of the learnability of VoiceBrowse in a short period of time.

Figure 8.7 presents the average number of turns per scenario for each system group, which shows a slight decrease in System 1 after training - another indication of the system's learnability. However the average number of turns per scenario slightly increases after training in System 2, an indication that this system is more usable without training than System 1 without training. There exists more room for improvement with System 1 from the user's perspective, and so throughout Figure 8.1 to 8.7, System 2's minimal difference after training is not alarming.

Figure 8.8 reflects the computed overall scores from the user questionnaires, per user group. Overall however, Figure 8.8 shows a slight increase in mean satisfaction after training for both systems. The overall impact of pre and post training on each system with respect to efficiency, annoyance, cognitive demand, likeability and accuracy also show slight improvements, illustrated in Figures 8.9 – 8.13. However this overall

comparison offers only a high level representation of the usability of both systems, and the preceding figures illustrate a more accurate account of how the different users' age and experiences affected performance.

The next five figures clarify the five aspects of usability with respect to experience, categorised by system and by training group, leading to eight different user groups in total, allowing comparison to be made for each system, before and after training, and also between experience levels. Figure 8.14 shows the rated efficiency for each user group, and it can be seen that, in general, Experienced users recorded a higher efficiency rating than Inexperienced users. Overall the effect of training on rated efficiency for both Experienced and Inexperienced users is minimal, and as predicted, Inexperienced users showed a preference to interact with System 1 over System 2 due to the system directed dialogue guiding the user to task completion.

Figure 8.15 presents a less varied level of distribution of annoyance ratings with respect to user groups - a user's experience level seems to have no effect on level of annoyance, although Inexperienced users appear less annoyed when using System 1 than System 2. The effect of training on annoyance seems to be minimal.

Greater variation is observed in Figure 8.16 which shows the cognitive demand for each user group. Common across all groups is that Experienced users record a higher score than Inexperienced users which leads to a smaller cognitive demand due to reverse polling. Furthermore training seems to have an equal effect across all users groups, with each showing a similar increase in rating for Scenario 2. The cognitive demand does not appear to be affected by system type, signifying that content delivered through voice is cognitively independent of the dialogue strategy used.

A similar representation of data is observed in Figure 8.17, which displays the likeability ratings for each user group: Experienced users constantly record a higher likeability rating than Inexperienced users; the effect of training is consistent across all user groups, each showing similar increases; and likeability of each user group when compared to the same user group that interacted with the alternative system is inconclusive. The ratings of all user groups were normally quite high, indicating that although likeability does depend on both experience and training, all user groups tended to like interacting with the system initially.

Lastly in the comparison of experience levels is the accuracy ratings recorded by the different user groups (Figure 8.18). The observations with respect to accuracy were opposite to those witnessed with respect to cognitive demand and likeability – that is, Inexperienced users constantly recorded a higher accuracy rating than Experienced users. Additionally both training and system type had an effect on accuracy; it tends to increases after training; and the same user group in System 2 has an increased accuracy rating compared with the same user group in System 1. This is an interesting finding which could indicate that, due to a better knowledge and practice of graphical web browsing, Experienced users have higher demands than Inexperienced users with respect to browsing online content through voice. As the Inexperience users had little or no prior usage of graphical web browsers, the consequence was that they had no prior knowledge on which to base accuracy conclusions, and therefore recorded consistently higher accuracy ratings than their Experienced counterparts.

The last five diagrams illustrate the usability aspect with the last category of interest – the age of the user. Figure 8.19 shows that, with the exception of users interacting with System 2 during the Untrained phase of the evaluation, Young/Old comparisons

indicate that Older users constantly rate the interaction as more efficient. It is thought that, similar to the accuracy results from the Experienced/Inexperienced comparison (Figure 8.18), Older users have less prior knowledge of web browsing, and therefore a lesser benchmark to compare the interaction to. Younger users, however, who are expected to be frequent users of technology, or at least have a higher perception of technology, would therefore be comparing the efficiency of voice browsing to graphical browsing. This would explain therefore why Older users record a higher efficiency rating than Young users. Surprisingly System 2 was generally rated less efficient than System 1, and the effect of training is minimal on increasing the level of efficiency - beginning in an open dialogue in System 2, which could lead to a high expectation of the interaction, then switching to system directed after the initial query, perhaps impacts the perceived efficiency of the system. This indicates that dialogue strategy does have a direct effect on the perceived efficiency of a voice browse system.

Figure 8.20, which presents the annoyance level with respect to user group, shows no correlation between age and the level of annoyance during interaction. As expected however, Older users seems to be less annoyed with System 1 whilst Younger users seem to be less annoyed with System 2. Lastly the effect of training on annoyance levels is inconclusive, with both decreases and increases in rating being observed after training.

Similar to the findings of cognitive demand with respect to experience (Figure 8.16), Figure 8.21 shows also that Younger users consistently scored a better cognitive rating than Older users. A correlation between cognitive demand, age and system type can also be observed, as it can be seen that Older users seem to rate System 1 with a better cognitive demand than System 2, but Younger users however seem to rate System 2

with a better cognitive demand than System 1. This seems to prove the hypothesis that Older users will prefer System 1 whereas Younger users will prefer System 2. This will be tested in the next section for significance. Once again, training also seems to have an effect on cognitive demand, with all user groups recording an increased rated in Scenario 2.

Figure 8.22 shows the rated likeability with respect to the different user groups. No comments concerning differences between age groups can be made with any confidence; however it appears that likeability does increase after training, and that System 2 consistently scores slightly higher ratings than System 1. This effect of training and dialogue strategy will be tested for significance in the next section.

The findings from Figure 8.23, showing the response accuracy with respect to user group, are similar to that of rated efficiency, presented in Figure 8.18 – that is that Older users constantly record higher response accuracy than Younger users. This is arguably also due to their less demanding requirements regarding technology, and therefore rate voice browsing with more appreciation than Young users. Both training and System 2 led to the same or slightly improved response accuracies before training and with System 1, indicating that usage and dialogue strategy do have an effect on response accuracy, which will be tested for significance in the following section.

8.3 VoiceBrowse Evaluation: Discussion

The results of the evaluation, illustrated with respect to system type, experience level, and age, will now be further explored and interpreted, and observations will be tested for significance to give an accurate understanding of the results.

8.3.1 Discussion: Hypothesis 1, Hypothesis 4 and Hypothesis 5

The increasing mean overall rating of the interactions, shown in Figure 8.8, represents increased satisfaction across the four different user groups – Untrained System 1 users recorded the least overall score, followed by Trained users of System 1, then Untrained System 2 users, whilst the Trained System 2 user group recorded the highest overall score. This suggests that System 2 offers the most usable version of VoiceBrowse – even for Untrained users when compared to users who have been trained on System 1. However, the distribution of the overall scores is similar between the Untrained and Trained user groups of System 2, showing that increased usage of System 2 does not lead to increased performance on the system in general. This can be confirmed with certain t-tests showing that only scenario duration, the average system turn time, the number of time outs and cognitive demand show any significant improvement after training ($p=0.00$, $p=0.001$, $p=0.009$ and $p=0.005$ respectively). This is due to System 2 accepting the initial user's request in an open-ended manner, whereas there is more learning to be done by the users when initially using System 1, as they are required to request information in a specific way.

To further understand this comparison between systems with regard to usability and the effect of training, the overall rating was broken down and studied with regard to the categories mentioned previously, and Figures 8.9 to 8.13 present the distribution of results in these categories for the different user groups. The graphs show that users' ratings with regard to efficiency, annoyance, cognitive demand, likeability and accuracy increase from Untrained to Trained groups and from System 1 to System 2, as previously discussed. However, t-tests on the associated data show that none of these increases are significant between System 1 and system 2, although Accuracy is an

exception, which shows a significant difference between System 1 and System 2 during the Untrained phase (p=0.007), indicating that users without prior use of the system find that it is an open-ended system that responds better.

Regarding differences observed after training, a significant increase in both the likeability and response accuracy was also observed after training for System 1 (p=0.033 and p=0.01 respectively). A significant increase in cognitive demand was also observed after training for both Systems 1 and 2 (p=0.013 and p=0.005 respectively).

With respect to the performance of System 1 and System 2, Figures 8.1 to 8.7 present the key findings after analysis of the annotated dialogue data from the log files, and the interaction parameters vary more in difference than the subjective measures already discussed.

It can be seen that the scenario duration (Figure 8.1) decreases in both systems after users have received training, and as expected, System 2 has shorter scenario durations than System 1. This reduction is significant after training between System 1 (366299ms) and System 2 (286663ms) (p=0.002). In System 2, the duration decreased from 452438ms to 286663ms after training, and the decrease of the scenario duration in System 1 is highly significant (p=0.007), down from 468322ms to 366299ms, showing that the training phase does have an effect on performance for System 1. Once again it can be said that the dialogue strategy therefore has an effect on the task duration – in System 1 the closed-approach requires time from the user to learn how to interact with the system, and so as the number of interactions increase with the system, the average task time decreases.

The reduction in the number of <noinput> elements across user groups, shown in Figure 8.2, is confirmed to be significant between System 1 and System 2, decreasing from 2.063 to 0.938 for Untrained users (p=0.047). Furthermore, learnability is also evident as there is a highly significant reduction in the number of <noinput> elements after the training phase for both systems, decreasing dramatically to 0.438 and 0.188 respectively (p=0.007 and p=0.009) for Systems 1 and 2. Many users, for example, failed to provide input the first time that they encountered the navigational controls 'next', 'back' and 'repeat' during the output of the document descriptions. However, during subsequent attempts, they were successful in both speaking at the right time, and also using the right command.

This commonly occurred during Scenario 1 Task 2 – a task which asks the user to access more information regarding a story, deliberately chosen to be in the second set of three stories that are output to the user. The system iterates through the navigational commands available at the start of the prompt, followed by the first three stories. Consequently however, once the first three stories have been output, users often experienced a <noinput> error as they either did not realise that they could speak, or had forgotten what they could say due to the length of the preceding prompt. However, in subsequent tasks, no such error was experienced, as the user had quickly learnt how to proceed beyond the third story.

The same observation is also true for the reduced number of <nomatch> elements (Figure 8.6), which significantly reduced after training for System 1 from 2.438 to 1 (p=0.015), once again highlighting the initial time needed for users to adapt to the closed method of dialogue control. System 2 always had a lower number of <nomatch> elements when compared to System 1 during both the Untrained and Trained phase of

the evaluation, a difference tested to be significant in the trained dialogues (p=0.008). As a <nomatch> error was commonly thrown when the system could not identify a relevant feed to satisfy the user's request, this explains why the number of <nomatch> errors were always lower than with System 1, as System 2 has limited potential in a dialogue for a <nomatch> to occur.

The number of Barge-Ins (Figure 8.3) also increased with usage – increasing from 6.625 to 8.375 in System 1 and from 1.375 to 1.688 in System 2 after training. Although the increase was tested to be not significant after training (p=0.134 and p=0.289 respectively), this is a further indication of the high learnability associated with both systems. Additionally, the number of Barge-ins for System 1 was always greater than System 2, reflecting the lower number of opportunities in System 2 for the user to barge in.

The number of help requests (Figure 8.4) was 0 in both the Trained and Untrained dialogues for System 2. The mean of 1.063 help requests per dialogue in System 1 during the Untrained phase was shown to be significantly higher than System 2 in the same phase(p=0.045), however the number of help requests reduced after training for System 1 to 0.125, although not significantly (p=0.069). It was common for users interacting with System 1 to request the help functionality during initial use with the system – however, as time progressed, and their familiarity with the system increased, there was no need for the help functionality once again.

The lower number of new query states (Figure 8.5) in System 2 compared to System 1 was tested to be highly significant in the Untrained phase of the evaluation (p=0.001). This indicates that during initial interactions, users found requesting the correct information troublesome in System 1, often having to start a new query before being

able to proceed. After training however, the number of new query states decreases significantly for System 1 (p=0.001), arguably again due to the learnability of the system. Additionally, the significantly reduced number of user turns per scenario (p=0.048) in System 1 after training reinforces this argument, showing that the user has learnt in the short time available how to proceed in the dialogue to the relevant feeds.

The recorded number of turns per scenario (Figure 8.7) decreased significantly in System 1 from 65.313 to 56.813 (p=0.0035), however slightly increased in System 2 from 42.063 to 43.688 after training. As expected, System 2 always required a smaller number of turns per scenario, also tested to be a significant difference between the systems after training (p=0.001). This reinforces the emerging pattern that learnability is evident more so in System1 than System 2, due to the initial closed dialogue of System 1 requiring more learning from the user.

In conclusion the evaluation supports both Hypotheses 1 and 4 that system type (representing dialogue strategy) will have an effect on usability and performance, and that learnability will be evident in the system – evidence of the latter has been observed to be significant in many of the interaction parameters presented, such as the average scenario times being reduced, in particular for System 1, from 78s in Scenario 1 to 61s in Scenario 2 (p=0.007).

Many tangible examples can be provided to illustrate learnability – user 15 for example fails to complete Scenario 1 Task1, and gets confused interacting with the system, including after requesting help. However, a similar task in Scenario 2 Task 1 results in a successful completion in only 48 seconds. It is evident also that learnability is high within Scenario 1 itself – the same user who failed to complete Scenario 1 Task 1 did manage to complete Task 2 of the same scenario. Although this is an extreme example,

many of the dialogues show a progression of learnability within the first two to three tasks of the Scenario 1.

Furthermore the included help and disambiguation facilities of the Dialogue Manager also played a key role in the usability of the system. The use of the 'list' function was found to be quite common amongst the first tasks of the evaluation. Not knowing what to say during their first interaction, the key help word 'list' appears at the end of the initial prompt, and so is the last word that the user hears before the system expects user input. For example, user 5 during their first interaction uttered the word 'list'. Once they had heard the possible options, and uttered "done" to signal they had finished with the list function, their next utterance was then "BBC Northern Ireland" which allowed them to complete the first task. It is important to note that this list function was not demonstrated to any user before their evaluation, and so completing this important step toward using VoiceBrowse was based on their own intuition.

Regarding Hypothesis 5, criteria to confirm this hypothesis are somewhat more difficult to quantify. Each task presented to the users during the evaluation was unique, meaning that the above performance and usability evaluation was conducted based on content accessed from twelve different web sites and four different providers (BBC, Sky, eBay and Travel Fusion). The successful completion of all scenarios by each participant could itself be evidence that browsing the web through voice using RSS and API feeds is feasible.

8.3.2 Discussion: Hypothesis 2

The overall comparison contrasting systems presented in Section 8.3.1 does not give an accurate reflection of the usability and performance of each system, as the different user

groups were treated as one. Hypotheses 2 and 3, which categorise the data by experience and age of the users offer more insight into the usability and performance of VoiceBrowse, which is expected to differ for age and experience of user.

Figure 8.14 shows the efficiency rating for the different user groups, generally illustrating that Experienced users found the interaction more efficient than Inexperienced users – a difference that was found to be significant before training with System 1 ($p=0.041$). No other comparison between either system or experience level produced a significant difference, indicating that, surprisingly, efficiency does not depend on either experience or dialogue strategy.

As previously discussed, Figure 8.15 produces no obvious difference of annoyance rating with respect to either experience or system type, suggesting that generally this usability aspect also does not depend on either.

However, the difference reported in Figure 8.16 that Experienced users achieved a higher cognitive rating, resulting in a lower demand, is significant between Experienced and Inexperienced users interacting with System 1 ($p=0.033$). This difference is of an even higher significance after training ($p=0.001$). This was expected due to the prior knowledge of web browsing found in the Experienced group, and as users of technology, it is assumed they would be able to learn or adapt to a new medium quicker than Inexperienced users.

Independent of system however, the effect of training on the cognitive demand is significant, both systems recording a higher cognitive rating ($p=0.013$ and $p=0.005$) than before training. Furthermore a significant difference is also observed for Inexperienced users after training between System 1 and System 2, with Inexperienced

users who interacted with System 2 requiring less cognitive demand than users interacting with System 1 ($p=0.006$). Not only is this an indication of learnability in the two systems, it additionally proves that, independent of experience level, improvement is shown with either system in a short space of time, with System 2 producing decreases in cognitive demand for both Experienced and Inexperienced users ($p=0.038$ and $p=0.061$ respectively). Inexperienced users interacting with System 1 also produced a significant decrease ($p=0.08$) of cognitive demand whilst the difference before and after training for Experienced users of System 1 was not significant ($p=0.104$). This further suggests that all users had to concentrate initially during the interactions until learning had taken place, at which point all users recorded that they had to think less when using the system – however System 2 seems to facilitate this better than System 1.

Similar arguments can be made concerning the likeability of different experience levels – that is Experienced users significantly liked the interaction more than Inexperienced users ($p=0.002$). Once again this was expected, as it is assumed that Experienced people who are users of the Internet tend to like technology and new mediums for interaction more so than those who do not. However, although training had an increased effect on likeability for all user groups, only the Inexperienced users for both Systems 1 and 2 showed a significant increase ($p=0.01$ and $p=0.052$ respectively). This seems to suggest that initial impressions of Experienced users are more fixed that those of Inexperienced users, who are open to learning and improving their impressions of the system. Lastly, as noted before, comparison of the same user group interacting with the alternative system was inconclusive, however, a significant increase in likeability was found between trained Experienced users of System 1 and System 2, with users of System 2 rating the likeability higher ($p=0.062$). This further establishes the belief that

Experienced users would prefer a more open ended system than Inexperienced users, confirmed here to be significant after training has been received by both user groups.

The previously discussed difference of increased accuracy ratings by Inexperienced users when compared to Experienced users (Figure 8.18) was found not to be significant. Furthermore, with regard to the noted increase in accuracy after training, this was also found not to be significant, except for Inexperienced users interacting with System 1, who reported a significant increase of accuracy ($p=0.083$). This further suggests that the difference in usability before and after training for both user groups interacting with System 2 is minimal, due to the forgiving nature of the open ended dialogue, and that there is therefore more potential for learning in System 1. This is reinforced by a significant difference in accuracy that was recorded between System 1 and System 2 Inexperienced uses before training ($p=0.035$), although after training there was no significant difference.

Regarding interaction parameters with respect to Experienced level of users, Experienced users had significantly more <nomatches> than Inexperienced users in System 1 ($p=0.001$). This was not expected, and could be a side effect of the knowledge of web browsing Experienced users had brought to the evaluation, getting confused with the new interaction method, whereas the Inexperienced group who are not users of technology were more willing to listen to the system's instruction. After training however no significant differences could be found in either system, suggesting that the performance of browsing online content is not dependent on prior experience of web browsing, however usability is heavily affected by this user characteristic.

8.3.3 Discussion: Hypothesis 3

The observation that Older users rated the interactions as more efficient than Younger users has been tested to be not significant. Additionally, any differences observed after training and between System 1 and 2 were also not significant. Whereas experience level therefore had a significant effect on efficiency, age did not, and training and dialogue strategy did not have a significant impact on efficiency with regard to different age ranges.

The difference in annoyance levels between Older and Younger users (Figure 8.20) was shown to be significant after training, with Older users scoring a significantly lower annoyance rating than Younger users (less annoyed) ($p=0.016$). No significant difference was observed for the same user group before and after training, or when compared to the alternative system. This is an interesting finding, indicating that Older users were generally more accepting of the technology than Younger users, perhaps suggesting that they are more patient with a dialogue system than Younger users, who are perhaps familiar with the instant delivery of content through a graphical user interface.

In Figure 8.21, the observed difference between Older and Younger users regarding cognitive demand was tested to be significant for System 2 during the Untrained Scenario – that is Younger users rated a higher cognitive score, requiring less cognitive demand, than Older users ($p=0.0036$). This proves that, as expected, Older users found that they had to concentrate more initially during the dialogue. Training does have a significant effect however for Older users, who show a significant decrease in required cognitive demand (higher rating) in both System 1 and System 2 after training than before training ($p=0.002$ and $p=0.031$ respectively), reflecting further the learnability of

the system. Regarding Younger users after training, no significant decrease in cognitive demand was recorded, indicating that Younger users found that they had to concentrate just the same after training as before – however they did seem to significantly prefer System 2 over System 1, recording a significant decrease in cognitive demand ($p=0.018$) for System 2 compared with System 1. This reinforces the original hypotheses that Older users would show an increase in performance after training, and that Younger users would prefer System 2 to System 1 due to the flexibility of the dialogue.

As previously discussed, no obvious differences between age groups could be identified for likeability ratings (Figure 8.22), and t-tests proved that any differences in likeability were indeed not significant. Dialogue strategy also produced no significant difference in likeability for the different age groups, but the effect of training was significant for Older users interacting with System 2, who showed an increase in likeability after training ($p=0.026$). Once again, this could be an indication that Older users become more pleased with the system over time, whereas Younger users were perhaps expecting more from the system.

Regarding response accuracy (Figure 8.23), Older users recorded significant increases over Younger users in System 1 before and after training ($p=0.002$ and $p=0.024$ respectively). This could be a further indication that Younger users have a higher demand with regard to browsing the Internet though voice and Older users are therefore more pleased with the system than Younger users. The noted differences for the same user group after training and with System 2 over System 1 were shown to be not significant.

Further significant differences between Older and Younger users interacting with System 1 were observed after training that were not evident before. Already discussed are the significantly higher response accuracy and annoyance rating by the Older users ($p=0.024$ and $p=0.016$ respectively) – however Older users also took significantly fewer turns per scenario than Younger users ($p=0.016$), and had significantly fewer <noinputs> than Young users ($p=0.042$), after training – all of which resulted in significantly shorter durations than Younger users ($p=0.039$). Interestingly however, System 2 shows only one significant difference between age groups after training, with Younger users recording significantly more barge-ins than Older users ($p=0.001$) – this is a further indication that learnability is greater in System 1, specifically amongst Older users.

8.4 VoiceBrowse Evaluation: Conclusions

Analysis of the data has supported all five hypotheses regarding the usability and feasibility of browsing the Internet through Voice. Furthermore, due to System 2 being more usable, as discussed in Section 8.3.1, the effect of learning was minimal, with similar performance and user rating records for Scenario 2 of the evaluation. The only parameter to show a significant change was a decrease in the number of timeouts ($p=0.009$), a change also shown in System 1 ($p=0.007$).

In contrast the training phase for System 1, which was not as usable as System 2 initially, had a greater effect on usability. Shorter dialogue durations, a lower number of turns per dialogue, fewer <noinputs> and <nomatches> all confirm that, after users initially interacted with the system, learnability was evident, and within a short period

of time the performance of System 1 increased ($p=0.007$, $p=0.048$, $p=0.007$ and $p=00015$ respectively).

Furthermore after training, differences between Older and Younger users became more evident as Older users increased their usability ratings, sometimes significantly, more than Younger users. This suggests that usability levels recorded by Younger users generally remain static or improve slightly after training, whereas Older users improve more than Young people after Scenario 1, recording a correlated increase in scores such as likeability and cognitive demand.

Learning had the opposite effect on experience, with any difference between Experienced and Inexperienced users decreasing after training. This suggest that initial experience of web browsing is desirable in terms of usability, however, after interacting with the system for a short period of time, any inadequacies regarding experience level are generally compensated, and the performance thereafter is not significantly affected. However, this could argue for the need for adaptive techniques in the system, leading to a faster learning phase for Inexperienced users whilst not decreasing the performance of Experienced users.

Interestingly, the level of experience did have an effect on the rated accuracy and efficiency of the system, with Inexperienced users scoring these aspects higher than Experienced users. This suggests that associated demands of web browsing, such as the extraction and delivery of content, and the presentation through voice, are higher for Experienced users due to their already formulated perceptions of web browsing through graphical user interfaces, and the speed of content delivery through this medium. Without any prior web browsing through a graphical user interface, less experienced users were generally happy with the performance of VoiceBrowse.

While user performance generally gets better after training, some subjective ratings from users did deteriorate, although any difference that did occur was small and insignificant - indicating that users judged the system in their first Scenario slightly better than in the second, possibly reflecting the user's opinions of the system settling down after their initial first impressions.

Finally, mention must be made of the narrative comments recorded by participants following the evaluation. Although entirely subjective, providing no empirical measures for comparison, comments can offer accurate reflections on the users' thoughts regarding the interactions overall, including considerations of what they would like specifically improved in VoiceBrowse.

Generally the comments indicated that users were happy with the interaction, that they could see the potential of such a system, and that they felt comfortable completing the desired goals. Common responses to "Would you use VoiceBrowse?" included positive answers, with users recognising the 'hands free' possibility of voice browsing, allowing them to browse online content whilst doing other tasks that require hands on access – furthermore this was a common response to "What did you like most about using VoiceBrowse?"

Whilst positive comments highlighting the user's satisfaction with the system add weight to the argument for such a system, there is perhaps more benefit in exploring comments relating to what the users did not like about the system, and what they felt needed to be overcome in future versions. Shared feedback included users stressing that they did not feel in control of the interaction during initial usage, and that they had to concentrate more at the beginning to learn how to use the system for the remainder of the tasks – opinions that were quantified from available numerical data. One user went

on to recommend an adaptable system that offered more help during opening dialogues, reducing this guidance as time progressed, in response to their feelings that the system became too repetitive during the evaluation.

In summary, the evaluation consisted of 32 participants, split evenly between two experience and two age levels - the majority of whom have never used a spoken dialogue system before. Two scenarios were presented for completion, with the same questionnaire presented after each scenario. The evaluation has shown that generically managed dialogue is possible, driven by online content and knowledge extracted from various unstructured sources. All users groups were able to use the system, with Older users and Inexperienced users improving at a greater level than Younger users and Experienced users, leading to smaller differences between users groups after training.

Chapter 9: VoiceBrowse Conclusions

The following chapter concludes the thesis by summarising the research and its contributions. Finally future work to further the research is presented in Section 9.2.

9.1 VoiceBrowse Conclusions: Summary of Thesis

This thesis has been concerned with the development and evaluation of a dynamic dialogue system that can access unstructured online content for use in a dynamically evolving spoken dialogue system. Previous research on dynamic dialogue systems was explored, with the following shortcomings identified:

- Dynamic dialogue systems are generally developed to interact with one domain only.
- Dynamic dialogues systems require a well structured domain representation, normally created specifically for each system.
- Dynamic dialogue systems built to interact with online sources generally interact with only one web site or content provider.

Based on these current limitations, requirements were devised to realise a spoken dialogue system that would provide a dialogue interface to the Internet. The architecture of VoiceBrowse was then introduced and explored in detail, followed by a discussion of its implementation. Evaluation of the system was then undertaken with regard to its usability and technical performance, and results showed some significant findings, which can be summarised as follows:

- Training of users interacting with System 1 (closed approach) had a stronger effect on performance and ratings than with users of System 2.
- After training, differences in performance and usability between Older and Younger users became more pronounced.
- However, differences in performance and usability of Experienced and Inexperienced users became less pronounced after training.
- While user performance generally increased after training, the ratings (qualitative) showed no change or a slight decrease – this effect was not very strong or significant, and it is believed that there is a general tendency that users judged the system slightly better in the first dialogue than in the second after training.

9.2 VoiceBrowse Conclusions: Summary of Research Contributions

The contributions of the research can be divided into technical advances and evaluations of usability. Dynamic dialogue systems are generally dependent upon specifically crafted and well structured sources of domain knowledge that are readily accessible to the dialogue manager. VoiceBrowse however interacts with more than one type of domain knowledge from multiple sources that vary and are unstructured. This has furthered current dialogue research by the realisation of a generic dialogue manager that provides a dialogue interface to the Internet. The research has shown that, by making use of existing XML technologies, content from different online sources can be extracted and utilised in dialogue, overcoming the aforementioned shortcomings.

Additionally usability research to date, with regard to spoken dialogue systems, has concentrated mostly on task-based dialogues, and with assisting users of varying needs

to complete a set task in the quickest time that their skill level allows. Usability research for generic dialogue systems that interact with numerous domains in an opportunistic way is currently lacking, with no current in-depth study available to offer suggestions or knowledge in how to meet users' different needs within this application area. This research has shown that, although a generic dialogue manager for browsing the Internet is technically possible, both age and prior web browsing experience have an effect on usability – initial interactions highlight large differences in the needs of these different user groups, although after a short period of usage, the differences are minimised.

Younger users do seem to prefer the more flexible interaction that is possible with a user led initiative, whilst older users seem to prefer a more system directed initiative through the interaction. Furthermore, younger users tend to have higher expectations regarding the performance of such systems and therefore are less satisfied overall than older users when browsing online content through dialogue. To develop a generic dialogue manager that is truly usable and efficient for all users, it is clear that a single dialogue strategy is not the most efficient mechanism for doing so, and different strategies to meet the varying needs are therefore required. Consequently, one major contribution of the evaluation study has been to identify the need for adaptive spoken dialogue systems that cater for the needs of different user groups. Moreover, as differences between the groups tend to change over time, it is important that this adaptation evolves dynamically to meet the changing needs of the users.

9.2 VoiceBrowse Conclusions: Future Work

Potential future research could see work continue with VoiceBrowse in two directions: the realisation of those architectural components not focused on in this study; and the refinement of the current VoiceBrowse implementation.

A User Manager was included as part of the original conceptual architecture presented in Chapter 4, although not developed as it was outside the scope of the research focus and could indeed be the topic of a further dissertation. However a User Model would certainly enhance the current VoiceBrowse system, leading to more tailored interactions for each user. It is envisioned that, over time, a model would be constructed of the user's interests based on requests made during dialogues with VoiceBrowse. Further requests would then be passed to the User Model first to refine the query, before passing the request on to the Content Manager. Also, further web technologies could be used to build up an online repository of feeds available for VoiceBrowse, allowing Information Retrieval techniques such as Collaborative Filtering to suggest additional feeds for a user based upon the browsing habits of similar users.

A second area not developed in this research is the Device Manager, a component that would provide facilities for managing multiple devices and their capabilities within the VoiceBrowse environment. The current version of VoiceBrowse has been developed to operate on the medium of telephony - however, due to the multimedia nature of online content, there would be great potential for a system that could also provide video and graphical content. Once the Content Manager has returned content for output to the Dialogue Manager, the Device Manager could then infer if a more suitable device is available to output the content, and if so, switch the interaction to this device if desired by the user.

Regarding enhancements of the current functionality, there is scope to improve the post-delivery phase of the API results to the user, which suffers from a current lack of dialogue during post delivery. Typical task-based dialogue systems allow some degree of refinement of results as presented to the user, as the various means of the grouping and classification of the results in one domain can be set by the developer during the development stage. In a multi-domain system that has no prior knowledge of, and no active comprehension of the task-based data, the rules of query refinement are not known in advance by the dialogue manager. It is predicted that some form of language understanding component would be required to interpret the results and respond specifically to query modification – having a set of flight results for example, being able to resolve queries such as "Give me flights in the morning" would be needed, made more complex due to the multi-domain nature of VoiceBrowse. Furthermore an alternative, more automated way of API inclusion should be investigated to remove the manual work currently needed to add an API into the VoiceBrowse environment.

Also investigated briefly during the research were enhancements to the standard Cosine Similarity function used in VoiceBrowse, including the use of WordNet to allow for the inclusion of synonyms when matching a user's query to the document space. Although the preliminary results using WordNet were similar or slightly less accurate than the standard benchmark function, further study of using this resource and maximising its potential during the Content Spotting phase should be explored in more depth.

This thesis started with a quotation from Armstrong (1994) that asked the question "When should we start using such [speech] interfaces?" In 1994, Armstrong's belief was that that time should be 'now'. Perhaps the current lack of speech based interfaces in the public domain reinforces the difficulties of recognising, understanding and

generating speech, as well as usability questions that also arise when using such systems. Now that these issues are understood and to an extent addressed, the next stage of dialogue research has to see researchers thinking of dialogue as a generic interface to different domains and applications in order to increase the awareness and the usefulness of such systems.

Public perception of speech based system is generally that they should be capable of interaction in any domain, topic, or task, with robotic like accuracy in their speech recognition, understanding and generation. Current methodologies of hand crafting dialogue do not support such notions, and it will be the advancing techniques of utilising statistical approaches that become the future of dialogue research. However the current lack of standards and specifications for such systems could inhibit future progress and it is paramount that research concentrates on developing such standards.

To conclude: this research has identified current shortcomings regarding dynamic dialogue systems, and their reliance on a well structured and purposely crafted source of domain knowledge; and it has provided a generic solution in the form of VoiceBrowse, a spoken dialogue system that utilises existing XML technologies to realise generic interactions based upon online sources of varying content types. The usability of this approach was explored in a series of evaluation studies, with findings suggesting that dialogue strategy, age and experience all have significant effects to various degrees, before and after training. Future research based on the current work should address issues of user modelling and device management, and enhance the Content Spotter currently in use.

References

Anonymous, 2006. *RSS File Forma*. Available from: http://en.wikipedia.org/wiki/RSS_%28file_format%29 [Accessed 18th July 2006]

Allen, J., Ferguson, G. & Stent, A. 2001. "An architecture for more realistic conversational systems". *6th international conference on Intelligent UserInterfaces, 14-17 January 2001 Santa Fe, USA.* Pp. 1-8.

Andersen, O. & Hjulmand, C. 2005, "Access for all - a talking internet service". 9*th European Conference on Speech Communication and Technology, InterSpeech 2005, 4-8 September 2005 Lisbon, Portugal*. Pp. 457-460.

Armstrong, B. 1994, "Speech recognition application program interface committee". *In Proceedings of AVIOS 1994, 20-23 September 2004 San Jose, CA.* Pp. 19-26.

Ayres, T. & Nolan, B. 2006, "Voice activated command and control with speech recognition over Wi-Fi", *Science of Computer Programming.* vol. 59, no. 1-2, Pp. 109-126.

Bangalore, S., Hakkani-Tur, D. & Tur, G. 2006. "Introduction to the Special Issue on Spoken Language Understanding in Conversational Systems", *Speech Communication*, vol. 48, no. 3-4, pp. 233-238.

Batacharia B., Levy D., Catizone R., Krotov A. & Wilks Y. 1999. "CONVERSE: A conversational companion", M*achine conversations.* Edited by Y. Wilks. Boston/Dordrecht/London: Kluwer. Pp. 205-215.

Bernsen, N.O. 2003, "On-line user modeling in a mobile spoken dialogue system". *8th European Conference on Speech Communication and Technology, EuroSpeech 2003, 1-4 September 2003 Geneva, Switzerland.* Pp. 737-740.

Bernsen, N.O., Dybkjaer, H. & Dybkjaer, L. 1998. *"Designing Interactive Speech Systems: From First Ideas to User Testing".* New York, United States of America: Springer-Verlag.

Beveridge, M. & Milward, D. 2003, "Combining Task Descriptions and Ontological Knowledge for Adaptive Dialogue". *6th International Conference on Text, Speech and Dialogue, 8-12 September, Czech Republic. P*p. 341.

Black, L.-A., McTear, M., Black, N., Harper, R. & Lemon, M. 2005. "Evaluating the DI@l-log system on a cohort of elderly, diabetic patients: results from a preliminary study". 9*th European Conference on Speech Communication and Technology, InterSpeech 2005, 4-8 September 2005 Lisbon, Portugal*. Pp. 821-824.

Bohus, D. & Rudnicky, A.I. 2003. "RavenClaw: Dialog Management Using Hierarchical Task Decomposition and an Expectation Agenda". 8th *European Conference on Speech Communication and Technology, EuroSpeech 2003, 1-4 September 2003 Geneva, Switzerland.* Pp. 597-600.

Bohus, D. & Rudnicky, A.I. 2005a, "Error handling in the RavenClaw dialog management framework". *In proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing, 6-8 October 2005 Vancouver, Canada.* Pp. 225-232.

Bohus, D. & Rudnicky, A. I.2005b, "A principled approach for rejection threshold optimization in spoken dialog systems". 9*th European Conference on Speech Communication and Technology, InterSpeech 2005, 4-8 September 2005 Lisbon, Portugal*. Pp. 2781-2784.

Bohus, D. & Rudnicky, A.I. 2005c. "Sorry, I didn't catch that! -An Investigation of Non-Understanding Errors and Recovery Strategies", 6*th SIGdial Workshop on Discourse and Dialogue. 2-3 September 2005 Lisbon, Portugal.* Pp. 128-143.

Bohus, D., Puerto, S.G., Huggins-Daines, D., Keri, V., Krishna, G., Kumar, R., Raux, A. & Tomko, S. 2007. "Conquest–An Open-Source Dialog System for Conferences". *Proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics, 22-27 April 2007 Rochester, NY.*

Bollegala, D., Matsuo, Y. & Ishizuka, M. 2007. "An Integrated Approach to Measuring Semantic Similarity between Words using Information Available on the Web". *Proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics, 22-27 April 2007 Rochester, NY.* Pp. 340-347.

Bollegala, D., Matsuo, Y. & Ishizuka, M. 2007, "Measuring Semantic Similarity between Words using Web Search Engines", *16th International World Wide Web Conference, 8-12 May 2007 Alberta, Canada.* Pp. 757-766.

Boyer, L., Danielsen, P., Ferrans, J., Karam, G., Ladd, D., Lucas, B. & Rehor, K. 2000. *Voice eXtensible Markup Language (VoiceXML™) version 1.0*. Available from: http://www.w3.org/TR/voicexml/ [Accessed 18th July 2006].

Bray, T., Paoli, J. & Sperberg-McQueen, C.M. 1998. *Extensible Markup Language (XML) 1.0.* Available from: http://www.w3.org/TR/1998/REC-xml-19980210 [Accessed 17th July 2006].

Bronsted, T., Larsen, H.L., Larsen, L.B., Lindberg, B., Ortiz-Arroyo, D., Tan, Z.-. & Xu, H. 2005. "Mobile information access with spoken query answering". *Applied*

Spoken Language Interaction in Distributed Environments, 10-11 November 2005 Aalborg, Denmark. Paper 31.

Brusilovsky, P. & Tasso, C.E.R. 2004. "Preface to Special Issue on User Modeling for Web Information Retrieval", *User Modeling and User-Adapted Interaction,* vol. 14, no. 2-3, pp. 147-157.

Bühler, D., Minker, W., Haussler, J. & Kruger, S. 2002. "The SmartKom mobile multi-modal dialogue system". *Proceedings of ISCA Tutorial and Research Workshop on Multimodal Dialogue in Mobile Environment, 17-19 June 2002 Kloster Irsee, Germany.* Pp. 11.

Callejas, Z. & López-Cózar, R. 2005. "Implementing modular dialogue systems: a case of study". *Applied Spoken Language Interaction in Distributed Environments, 10-11 November 2005 Aalborg, Denmark.* Paper 13.

Callejas, Z. & López-Cózar, R. 2008. "Relations between de-facto criteria in the evaluation of a spoken dialogue system". *Speech Communication,* vol. 50, no. 8-9, pp. 646-665.

Capra III, R.G., Pérez-Quiñones, M.A. & Ramakrishnan, N. 2001. "WebContext: Remote Access to Shared Context". *Proceedings of Perceptive User Interfaces 2001, 15-16 November 2001 Orlando, Fl.* Pp. 1-9.

Capra III, R.G. 2003. "Mobile information re-finding as a continuing dialogue". *Proceedings of Human Factors in Computing Systems, 5-10 April 2003 Ft. Lauderdale Florida,* Pp. 664.

Capra III, R.G. & Pérez-Quiñones, M.A. 2005, "Mobile re-finding of web information using a voice interface: an exploratory study", *Proceedings of the Latin American Conference on Human-computer Interaction 2005, 23-26 October 2005 Cuernavaca, Mexico. P*p. 88-99

Carenini, G. & Moore, J. 2001. "A strategy for evaluating generative arguments". *1st International Conference on Natural Language Generation, San Diego, CA.* Pp. 1307–1314.

Carlson, R., Hirschberg, J. & Swerts, M. 2005, "Error handling in spoken dialogue systems", *Speech Communication,* vol. 45, no. 3, pp. 207-209.

Chen, B., Chen, Y., Chang, C. & Chen, H. 2005, "Speech retrieval of Mandarin broadcast news via mobile devices". 9*th European Conference on Speech Communication and Technology, InterSpeech 2005, 4-8 September 2005 Lisbon, Portugal*. Pp. 109-112.

Choi, O., Han, S., Abraham, A. 2005. "Integration of Semantic Data using a Novel Web Based Information Query System". *International Journal of Web Services Practices,* vol. 1, no. 1-2, pp. 21-29.

Chu, S.W., O'Neill, I., Hanna, P. & McTear, M. 2005, "An approach to multi-strategy dialgoue management". 9*th European Conference on Speech Communication and Technology, InterSpeech 2005, 4-8 September 2005 Lisbon, Portugal*. Pp. 865-868.

Chu-Carroll, J. 2000. "MIMIC: An adaptive mixed initiative spoken dialogue system for information queries". *6th Conference on Applied Natural Language Processing, 29 April-4 May 2000 Seattle WA. P*p. 97–104.

Chu-Carroll, J., & Carpenter, B. 1999. "Vector-Based Natural Language Call Routing". *Computational Linguistics,* vol. 25, pp. 361-388.

Churcher, G.E., Atwell, E.S. & Souter, C. 1997. "Dialogue management systems: a survey and overview". *School of Computer Studies, University of Leeds, UK.*

Claypool, M., Gokhale, A., Miranda, T., Murnikov, P., Netes, D. & Sartin, M. 1999. "Combining content-based and collaborative filters in an online newspaper". *ACM SIGIR Workshop on Recommender Systems: Algorithms and Evaluation, 19 August 1999 University of California, USA.*

Clark, H.H. 1996. *"Using Language".* UK: Cambridge University Press.

D'Haro, L.F., de Córdoba, R., Ferreiros, J., Hamerich, S.W., Schless, V., Kladis, B., Schubert, V., Kocsis, O., Igel, S. & Pardo, J.M. 2006. "An advanced platform to speed up the design of multilingual dialog applications for multiple modalities". *Speech Communication,* vol. 48, no. 8, pp. 863-887.

den Os, E., Boves, L., Rossignol, S., ten Bosch, L. & Vuurpijl, L. 2005, "Conversational agent or direct manipulation in human–system interaction", *Speech Communication,* vol. 47, no. 1-2, pp. 194-207.

Dybkjaer, L. & Bernsen, N.O. 2001, "Usability issues in spoken dialogue systems", *Natural Language Engineering,* vol. 6, no. 3-4, pp. 243-271.

Dybkjær, L., Bernsen, N. O., Dybkjær, H., 2005. "Usability evaluation issues in commercial and research systems". *Applied Spoken Language Interaction in Distributed Environments, 10-11 November 2005 Aalborg, Denmark.* Paper 29.

Feng, J., Bangalore, S. & Rahim, M. 2003. "WebTalk: mining Websites for automatically building dialog systems". *Automatic Speech Recognition and Understanding, 30 November-3 December 2003, U.S. Virgin Islands, UDS.* Pp. 168

Feng, J., Bangalore, S., Rahim, M. 2004. "Question-Answering in WebTalk: An Evaluation Study*".* 8*th International Conference on Speech Language Processing, InterSpeech 2004, 4-8 October 2004 Jeju Islands, Korea.* Pp. 2613-2626

Feng, J., Reddy, S. & Saraclar, M. 2005. "WebTalk: Mining Websites for Interactively Answering Questions". 9*th European Conference on Speech Communication and Technology, InterSpeech 2005, 4-8 September 2005 Lisbon, Portugal*. P2485-2488.

Feng, J., Hakkani-Tur, D., Di Fabbrizio, G., Gilbert, M. & Beutnagel, M. 2006. "WebTalk: Towards Automatically Building Spoken Dialog Systems Through Mining Websites". *31st Conference on Acoustics, Speech and Signal Processing, 14-19 May 2006 Toulouse, France.* Pp. 573-576.

Fensel, D. 2003. "*Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential*". USA: MIT Press.

Fischer, G.E.R. 2001. "User Modeling in Human-Computer Interaction". *User Modeling and User-Adapted Interaction,* vol. 11, no. 1 - 2, pp. 65-86.

Fischer, P., Österle, A., Berton, A. & Regel-Brietzmann, P. 2007, "How to personalize speech applications for web-based information in a car". *10th European Conference on Speech Communication and Technology, InterSpeech 2007, 27-31 August 2005 Antwerp, Belgium.* Pp. 2557-2560.

Fitzgerald, M. 2004. "*Learning XSLT*". 1st ed, USA: O'Reilly & Associates, Inc. Pp. ix.

Froumentin , M. & Ashimura, A. 2006. "*W3C "Voice Browser" Activity"*. Available from: http://www.w3.org/Voice/ [Accessed 14th July 2006]

Ganesan, P., Garcia-Molina, H., Widom, J. 2003. " Exploiting Hierarchical Domain Structure to Compute Similarity". *ACM Transactions on Information Systems,* vol. 21, pp. 64-93.

Garcia, E. 2006. *"The Classic Vector Space Model - Description, Advantages and Limitations of the Classic Vector Space Model".* Available from: http://www.miislita.com/term-vector/term-vector-3.html [Accessed 5th February 2007].

Giorgino, T., Azzini, I., Rognoni, C., Quaglini, S., Stefanelli, M., Gretter, R. & Falavigna, D. 2005. "Automated spoken dialogue system for hypertensive patient home management". *International Journal of Medical Informatic.*, vol. 74, no. 2-4, pp. 159-167.

Göker, M.H. & Thompson, C.A. 2000. "The Adaptive Place Advisor: A Conversational Recommendation System". *8th German Workshop on Case-Based Reasoning, 2-3 March 2000 Lamerbuckel, Germany.*

González-Ferreras, C. & Cardenoso-Payo, V. 2005, "Development and evaluation of a spoken dialog system to access a newspaper web site", 9*th European Conference on Speech Communication and Technology, InterSpeech 2005, 4-8 September 2005 Lisbon, Portugal*. Pp. 857-860.

Griol, D., Hurtado, L.F., Segarra, E., Sanchis, E. & i Computacio, D.S.I. 2008. "A Two Phases Statistical Approach for Dialog Management". *Perception in Multimodal Dialogue Systems: 4th IEEE Tutorial and Research Workshop on Perception and Interactive Technologies for Speech-Based Systems, 16-18 June 2008 Kloster Irsee, Germany.*

Gruenstein, A., Seneff, S. & Wang, C. 2006, "Scalable and portable web-based multimodal dialogue interaction with geographical databases". 9*th International Conference on Speech Language Processing, InterSpeech 2006, 17-21 September 2006 Pittsburgh, PA.* Paper 1095.

Haller, R. 2003. "The Display and Control Concept iDrive-Quick Access to All Driving and Comfort Functions". *In ATZ/MTZ Extra (The New BMW 5-Series), August 2003.* Pp. 51–53.

Hamerich, S.W., Schubert, V., Schless, V., de Córdoba, R., Pardo, J.M., d'Haro, L.F., Kladis, B., Kocsis, O. & Igel, S. 2004a. "The GEMINI platform: Semi-Automatic generation of dialgoue applications". 8*th International Conference on Speech Language Processing, InterSpeech 2004, 4-8 October 2004 Jeju Islands, Korea.* Pp. 2629-2632.

Hamerich, S.W., Schubert, V., Schless, V., de Córdoba, R., Pardo, J.M., d'Haro, L.F., Kladis, B., Kocsis, O. & Igel, S. 2004b/ "Semi-automatic generation of dialogue applications in the GEMINI project". 5*th SIGdial Workshop on Discourse and Dialogue, 1 May--30 April 2004 Boston, MA.* Pp. 31-31.

Hamerich, S.W., Wang, Y.H., Schubert, V., Schless, V. & Igel, S. 2003. "XML-Based dialogue descriptions in the GEMINI project". *Proceedings of the Berliner XML-Tage, 13-15 October 2003 Berlin, Germany.* Pp. 404-412

Hanna, P., O'Neill, I., & Liu X. & McTear, M. 2005. "Developing extensible and reusable spoken dialogue components: an examination of the Queen's communicator". 9*th European Conference on Speech Communication and Technology, InterSpeech 2005, 4-8 September 2005 Lisbon, Portugal*. pp. 1865-1868.

Hanna, P., O'Neill, I., Wootton, C. & McTear, M. 2007. "Promoting extension and reuse in a spoken dialog manager: An evaluation of the Queen's Communicator". *ACM Transactions on Speech and Language Processing),* vol. 4, no. 3, article 7.

Hardy, H., Baker, K., Bonneau-Maynard, H., Devillers, L., Rosset, S. & Strzalkowski, T. 2003. "Semantic and Dialogic Annotation for Automated Multilingual Customer Service". *8th European Conference on Speech Communication and Technology, EuroSpeech 2003, 1-4 September 2003 Geneva, Switzerland*. pp. 201-204.

Hardy, H., Biermann, A., Inouye, R.B., McKenzie, A., Strzalkowski, T., Ursu, C. Webb, N. & Wu, M. 2006. "The AMITIÉS system: Data-driven techniques for automated dialogue". *Speech Communication,* vol. 48, no. 3-4, pp. 354-373.

Hartikainen, M., Salonen, E.P. & Turunen, M. 2004. "Subjective Evaluation of spoken dialogue systems Using SER VQUAL Method". *8th International Conference on Spoken Language Processing, 4-8 October 2004 Jeju, Korea.* Pp. 2273-2273.

Hassel, L. & Hagen, E. 2005, "Adaptation of an automotive dialogue system to user's expertise", 6*th SIGdial Workshop on Discourse and Dialogue, 2-3 September 2005 Lisbon, Portugal.* Pp. 222-226

He, Y. & Young, S. 2006. "Spoken language understanding using the Hidden Vector State Model". *Speech Communication,* vol. 48, no. 3-4, pp. 262-275.

Hjalmarsson, A. 2005. "Towards user modelling in conversational dialgoue systems: A qualitative study of the dynamics of dialgoue parameters". 9*th European Conference on Speech Communication and Technology, InterSpeech 2005, 4-8 September 2005 Lisbon, Portugal.* Pp. 869-872.

Hocek, A. & Cuddihy, D. 2003. "*Definitive VoiceXML".* 1st ed. New Jersey, USA: Pearson Education, Inc.

Hone, K.S. & Graham, R. 2000. "Towards a tool for the Subjective Assessment of Speech System Interfaces (SASSI)". *Natural Language Engineering,* vol. 6, no. 3-4, pp. 287-303.

Hone, K.S. & Graham, R. 2001. "Subjective Assessment of Speech-System Interface Usability". *7th European Conference on Speech Communication and Technology, Eurospeech 2001, 3-7 September 2001 Aalborg, Denmark. P*p. 2082-2086.

Huang, X., Acero, A. & Hsiao-Wuen, H. 2001. "*Spoken language processing. A guide to theory, algorithm and system development*". New Jersey, USA: Prentice Hall PTR.

ISO 9241-11, 1998. "Ergonomic requirements for office work with visual display terminals (VDT)s – Part 11 Guidance on usability". Available from: http://www.miislita.com/term-vector/term-vector-3.html [Accessed 5th February 2007].

Jokinen, K., Kerminen, A., Kaipainen, M., Jauhiainen, T., Wilcock, G., Turunen, M., Hakulinen, J., Kuusisto, J. & Lagus, K. 2002. "Adaptive dialogue systems-interaction with interact". *3rd SIGdial Workshop on Discourse and Dialogue, 11-12 July 2002 Philadelphia, PA.* Pp. 64-73.

Jung, S., Lee, C., Kim, S. & Lee, G.G. 2008, "DialogStudio: A workbench for data-driven spoken dialog system development and management", *Speech Communication,* vol. 50, no. 8-9, pp. 697-715.

Jurafsky, D. & Martin, J.H. 2008. *"Speech and language processing. An introduction to natural language processing, computational linguistics and speech recognition".* 2nd ed. New Jersey, USA: Prentice-Hall Inc.

Kamm, C., Walker, M.A. & Litman, D. 1999. "Evaluating Spoken Language Systems", *Proceedings of AVIOS 1994, 20-23 September 2004 San Jose, CA.*

Kohrs, A. & Merialdo, B. 2000. "Using category-based collaborative filtering in the ActiveWebMuseum". *IEEE International Conference on Multimedia and Expo, 2 August-30 July 2000.* Pp. 351-354.

Komatani, K., Adachi, F., Shinichi, U., Kawahara, T. & Okuno, H.G. 2003. "Flexible spoken dialogue system based on user models and dynamic generation of VoiceXML scripts". *4th SIGdial Workshop on Discourse and Dialogue, 5-6 July 2003 Sapporo, Japan.*

Komatani, K., Ueno, S., Kawahara, T. & Okuno, H.G. 2005. "User Modeling in spoken dialogue systems to Generate Flexible Guidance". *User Modeling and User-Adapted Interaction,* vol. 15, no. 1, pp. 169.

Kobayashi, M. & Takeda, K. 2000. "Information retrieval on the web". *ACM Computing Surveys,* vol. 32, no. 2, pp. 144-173.

Kraemer, H.C. & Thiemann, S. 1987. *"How many subjects?".* California, USA: Sage Publications.

Lamel, L., Rosset, S. & Gauvain, J.L. 2000. "Considerations in the Design and Evaluation of Spoken Language Dialog Systems". *6th International Conference on Spoken Language Processing, 16-20 October 2000 Beijing, China*. Pp. 5-8.

Larson, J.A. 2003. "*VoiceXML: Introduction to developing speech applications"*. 1st ed. New Jersey, USA: Pearson Education, Inc.

Larson, J.A. 2005a. "VoiceXML: Industry perspectives and business opportunities". *Applied Spoken Language Interaction in Distributed Environments, 10-11 November 2005 Aalborg, Denmark.* Paper 43.

Larson, J.A. 2005b. "Voice User Interface Design for Novice and Experienced Users". *In "Practical Spoken Dialogue Systems. Text, Speech and Language Technology"*. Kluwer Academic Publishers: USA. Page 61.

Larson, J.A. et al. 2005. *"Ten Criteria for Measuring Effective Voice User Interfaces"*. *Speech Technology Magazine*, vol 10, num 6, issue 545, pp. 31-35.

Lee, C., Jung, S., Jeong, M. & Lee, G.G. 2006. "Chat and goal-oriented dialog together: a unified example-based architecture for multi-domain dialog management". *1st Workshop on Spoken Language Technology, 10-13 December 2006 palm Beach, Aruba.* pp. 194-197.

Leighton, H.V. & Srivastava, J. 1999. "First 20 precision among World Wide Web search services (search engines)". *Journal of the American Society for Information Science,* vol. 50, no. 10, pp. 870-881.

Lemon, O., Georgila, K., Henderson, J. & Stuttle, M. 2006. "An ISU dialogue system exhibiting reinforcement learning of dialogue policies: generic slot-filling in the TALK in-car system". *11th Conference of the European Chapter of the Association for Computer Linguistics, 3-7 April 2006 Trento, Italy.*

Litman, D., Singh, S., Kearns, M. & Walker, M. 2000. "NJFun: A Reinforcement Learning spoken dialogue system". *ANLP/NAACL Workshop on Conversational Systems, 4 May 2000 Seattle, WA. P*p. 17-20.

Litman, D.J., Walker, M.A. & Kearns, M.S. 1999, "Automatic detection of poor speech recognition at the dialogue level", *37th Annual Meeting of the Association for Computational Linguistics on Computational Linguistics, 20-26 June 1999 College Park, Maryland. P*p. 309-316.

López-Cózar, R. & Araki, M. 2005. *Spoken, multilingual and multimodal dialogue systems. Development and Assessment*". West Sussex, England: John Wiley & Sons Ltd.

López-Cózar, R., Callejas, Z., Gea, M. & Montoro, G. 2005. "Multimodal, multilingual and adaptive dialogue system for ubiquitous interaction in an educational space". *Applied Spoken Language Interaction in Distributed Environments, 10-11 November 2005 Aalborg, Denmark.* Paper 12.

McTear, M.F. 2004a. "*Spoken dialogue technology. Towards the conversational user interface*". London, United Kingdom: in Springer-Verlag.

McTear, M. 2004b. "New Directions in Spoken Dialogue Technology for Pervasive Interfaces". *Workshop on Robust and Adaptive Information Processing for Mobile Speech Interfaces, Satellite Workshop of 20th International Conference of Computer Linguistics, 28-29 August 2004 Geneva, Switzerland*. Pp. 57-64.

McTear, M., O'Neill, I., Hanna, P. & Liu, X. 2005. "Handling errors and determining confirmation strategies-An object-based approach". *Speech Communication,* vol. 45, no. 3, pp. 249-269.

Miller, M. 2002. "*VoiceXML: 10 Projects to voice-enable your web site".* 1st ed. New York, USA: Wiley Publishing, Inc.

Milward, D. & Beveridge, M. 2003. "Ontology-based dialogue systems". *3rd Workshop on Knowledge and Reasoning in Practical Dialogue Systems, 10 August 2003 Acapulco, Mexico.*

Mittendorfer, M., Niklfeld, G. & Winiwarter, W. 2002. "Making the VoiceWeb Smarter–Integrating Intelligent Component Technologies and VoiceXML". *2nd International Conference on Web Information Systems Engineering, 3-6 December 2001 Kyoto, Japan.* Pp. 126.

Möller, S. 2005a. "Evaluating Telephone-Based Interactive Systems". *Applied Spoken Language Interaction in Distributed Environments, 10-11 November 2005 Aalborg, Denmark.* Paper 42.

Möller, S. 2005b. "*Quality of Telephone-Based spoken dialogue systems*". Boston, USA: Springer. Pp. 27.

Möller, S., Englert, R., Engelbrecht, K., Hafner, V., Jameson, A., Oulasvirta, A., Raake, A. & Reithinger, N. 2006. "Memo: towards automatic usability evaluation of spoken dialogue services by user error simulations". 9*th International Conference on Speech Language Processing, InterSpeech 2006, 17-21 September 2006 Pittsburgh, PA.* Paper 1131.

Möller, S., Smeele, P., Boland, H. & Krebber, J. 2007. "Evaluating spoken dialogue systems according to de-facto standards: A case study". *Computer Speech & Language,* vol. 21, no. 1, pp. 26-53.

Möller, S., Engelbrecht, K. & Schleicher, R. 2008. "Predicting the quality and usability of spoken dialogue services". *Speech Communication,* vol. 50, no. 8-9, pp. 730-744.

Montoro, G., Alamán, X. & Haya, P.A. 2004. "A plug and play spoken dialogue interface for smart environments". *5th International Conference on Intelligent Text Processing and Computational Linguistics, 15-21 February 2004 Seoul, Korea.* Pp. 355-365.

Morrison, M. 2002. "*Sams teach yourself XML in 24 hours*". 2nd ed. USA: Sams publishing. Pp. 8-9.

Mueller, W., Schaefer, R. & Bleul, S. 2004. "Interactive multimodal user interfaces for mobile devices". *37th Annual Hawaii International Conference on System Sciences, 5-8 January 2004 Hawaii.* Pp. 286-295.

Nielsen, J. 1993. "*Usability Engineering*". London, United Kingdom: Academic Press Limited. Pp. 26.

Nielsen, J. & Landauer, T.K., 1993. "A mathematical model of the finding of usability problems". International *Conference on Human-Computer Interaction, 24-29 April 2004 Amsterdam, Netherlands*. Pp. 206-213.

Ng, C., Wilkinson, R., Zobel, J. 2000. "Experiments in Spoken Document Retrieval using Phoneme n-Grams". *Speech Communication, vol. 32, pp. 61-77*

O' Neill, I., Hanna, P., Liu, X., Greer, D. & McTear, M. 2005. "Implementing advanced spoken dialogue management in Java". *Science of Computer Programming,* vol. 54, no. 1, pp. 99-124.

Paek, T. 2001., "Empirical methods for evaluating dialog systems". 2nd *SIGdial Workshop on Discourse and Dialogue, 1-2 September 2001 Aalborg, Denmark.* Pp. 3-10.

Pargellis, A., Kuo, H.K.J. & Lee, C.H. 1999. "Automatic application generator matches user expectations to system capabilities". *ESCA Workshop on Interactive Dialogue in Multi-Modal Systems, 22-25 June 199 Kloster Irsee, Germany.* Pp. 37–40.

Pargellis, A.N., Kuo, H.-.J. & Lee, C.-. 2004. "An automatic dialogue generation platform for personalized dialogue applications". *Speech Communication*, vol. 42, no. 3-4, pp. 329-351.

Paris, C.L. 1993. "*User Modeling in Text Generation".* New York, USA: St. Martin's Press, Inc.

Pellom, B., Ward, W., Hansen, J., Cole, R., Hacioglu, K., Zhang, J., Yu, X. & Pradhan, S. 2001. "University of Colorado Dialog Systems for Travel and Navigation". *1st International Conference on Human Language Technology Research, March 2001 San Diego CA.* Pp. 1-6.

Pellom, B., Ward, W. & Pradhan, S. 2000. "The CU Communicator: An Architecture for Dialogue Systems". *6th International Conference on Spoken Language Processing, 16-20 October 2000 Beijing, China.* Pp. 723–726.

Perez-Quiñones, M. A.& Rode, J. 2004. "*You've Got Mail! Calendar, Weather and More: Customizable Phone Access to Personal Information".* USA: Virginia Tech, Technical Report TR-04-21.

Pieraccini, R. & Huerta, J. 2005. "Where do we go from here? Research and Commercial Spoken Dialog Systems". *6th SIGdial Workshop on Discourse and Dialogue, 2-3 September 2005 Lisbon, Portugal.* Pp. 1-10.

Platt, G.T. 2004. "VoiceXML versus salt: selecting a voice application standard". *Customer Interaction Solutions*, vol. 22, no. 11, pp. 50.

Polifroni, J., Chung, G. & Seneff, S. 2003. "Towards the automatic generation of mixed-initiative dialogue systems from web content". *8th European Conference on Speech Communication and Technology, EuroSpeech 2003, 1-4 September 2003 Geneva, Switzerland*. Pp. 193-196.

Polifroni, J. & Walker, M. 2006. "An Analysis Of Automatic Content Selection Algorithms For spoken dialogue system Summaries", *1st Workshop on Spoken Language Technology, 10-13 December 2006 Palm Beach, Aruba.* Pp. 186-189.

Popescul, A., Ungar, L.H., Pennock, D.M. & Lawrence, S. 2001. "Probabilistic models for unified collaborative and content-based recommendation in sparse-data environments". *17th Conference in Uncertainty in Artificial Intelligence, 2-5 August 2001 Seattle WA.* Pp. 437-444.

Rabiner, L. & Juang, B.H. 1993. "*Fundamentals of Speech Recognition".* New Jersey, USA: Prentice-Hall, Inc.

Raman, T.V. 1998. "*Audio System for Technical Readings"* Berlin: Springer.

Raman, T.V. 1997a. "*Auditory User Interfaces: Toward the Speaking Computer*". Norwell, USA: Kluwer Academic Publishers. Pp. 121.

Raman, T.V. 1997b. "Aural cascaded style sheets – the audible www". 6th International World Wide Web Conference, 7-11 April 1997 Santa Clara, CA.

Raux, A., Langner, B., Bohus, D., Black, A. W. & Eskenazi, M. 2003. "LET's GO: improving spoken dialog systems for the elderly and non-natives". *8th European Conference on Speech Communication and Technology, EuroSpeech 2003, 1-4 September 2003 Geneva, Switzerland*. Pp.753-756.

Raux, A., Langner, B., Bohus, D., Black, A. W. & Eskenazi, M. 2005. "Let's go public! taking a spoken dialog system to the real world". 9*th European Conference on Speech Communication and Technology, InterSpeech 2005, 4-8 September 2005 Lisbon, Portugal*. Pp. 885-888.

Raux, A., Bohus, D., Langner, B., Black, A. & Eskenazi, M. 2006. "Doing Research on a Deployed spoken dialogue system: One Year of Let's Go! Experience". 9*th International Conference on Speech Language Processing, InterSpeech 2006, 17-21 September 2006 Pittsburgh, PA.* Paper 1794.

Reithinger, N., Herzog, G. & Blocher, A. 2007. "SmartWeb-Mobile Broadband Access to the Semantic Web". KI-Künstliche Intelligenz, vol. 07, no. 2, pp. 30-33.

Richardson, R., & Smeaton, A. F. 1995, "Using WordNet in a Knowledge-Based Approach to Information Retrieval". 17th *BCS-IRSG Annual Colloquium, 4-5 April 1995 Crewe, UK.* .

Ringland, S. P. A. & Scahill, F.J. 2003. "Multimodality - The Future of the Wireless User Interface". *BT Technology Journal,* vol. 21, no. 3, pp. 181-191.

Rosset, S., Galibert, O., Illouz, G. & Max, A. 2006. "Integrating Spoken Dialog and Question Answering: the Ritel Project". 9*th International Conference on Speech Language Processing, InterSpeech 2006, 17-21 September 2006 Pittsburgh, PA.* Paper 1529.

Schubert, V. & Hamerich, S.W. 2005. "The dialogue application metalanguage GDialogXML". 9*th European Conference on Speech Communication and Technology, InterSpeech 2005, 4-8 September 2005 Lisbon, Portugal.* Pp. 789-792

Sharma, C. & Kunis, J. 2002. "*VoiceXML. Strategies and techniques for effective voice application development with VoiceXML 2.0*". New York, USA: John Wiley & Sons, Inc.

Sieg, A., Mobasher, B. & Lytinen, S. 2004. "Using Concept Hierarchies to Enhance User Queries in Web-Based Information Retrieval". *International Conference on Artificial Intelligence and Applications, 14-16 February 2004 Innsbruck, Austria*. Pp. 226–234

Sieg, A., Mobasher, B. & Lytinen, S. 2006. " Concept Based Query Enhancement in the Arch Search Agent". *4th International Conference on Internet Computing, 26-29 June 2006 Las Vegas, NV.*

Singh, S.P., Litman, D.J., Kearns, M.J. & Walker, M.A. 2002. "Optimizing Dialogue Management with Reinforcement Learning: Experiments with the NJFun System". *Journal of Artificial Intelligence Research,* vol. 16, no. 5, pp. 105-133.

Skantze, G. 2005. "Exploring human error recovery strategies: Implications for spoken dialogue systems". *Speech Communication,* vol. 45, no. 3, pp. 325-341.

Sturm, J. & Boves, L. 2005. "Effective error recovery strategies for multimodal form-filling applications". *Speech Communication,* vol. 45, no. 3, pp. 289-303.

Suhm, B. 2000, "Lessons learned for visual and verbal Interfaces from multimodal error correction". *W3C/WAP Workshop: the Multimodal Web, 5-6 September 2000,* Hong Kong.

Suhm, B., Myers, B. & Waibel, A. 2001. "Multimodal error correction for speech user interfaces". *ACM Transactions on Computer-Human Interaction (TOCHI),* vol. 8, no. 1, pp. 60-98.

Taylor, M.M., Néel, F. & Bouwhuis, D.G. 1989. "*The structure of multimodal dialogue".* Amsterdam, The Netherlands: Elsevier Science Publishers

Thompson, C.A., Goker, M.H. & Langley, P. 2004. "A Personalized System for Conversational Recommendations". J*ournal of Artificial Intelligence Research,* vol. 21, pp. 393-428.

Thompson, H. & le Hegaret, P. 2005. "XML Core Working Group Public Page." Available from: http://www.w3.org/XML/Core/ [Accessed 17th August 2006] .

Traum, D.R. & Allen, J.F. 1992. "A speech acts approach to grounding in conversation*". 2nd International Conference on Spoken Language Processing, 13-16 October 1992 Alberta, Canada.* Pp. 137-140.

Turunen, M. 2004. "*Jaspis–A Spoken Dialogue Architecture and its Applications".* Thesis (PhD). University of Tampere, Finland.

Turunen, M., Hakulinen, J., Raiha, K.-., Salonen, E.-., Kainulainen, A. & Prusi, P. 2004. "An architecture and applications for speech-based accessibility systems". *IBM Systems Journal*, vol. 44, no. 3, pp. 485-504.

Turunen, M., Salonen, E., Hakulinen, J., Kanner, J. & Kainulainen, A. 2005, "Mobile architecture for distributed multimodal dialogues". 9*th European Conference on Speech Communication and Technology, InterSpeech 2005, 4-8 September 2005 Lisbon, Portugal*. Pp. 849-852.

Veldhuijzen van Zanten, G. 1998. "Adaptive mixed-initiative dialogue management". *4th IEEE Workshop on Interactive Voice Technology for Telecommunications Applications, 29-30 September 1998 Torino, Italy*. Pp. 65-70.

Veldhuijzen van Zanten, G. 1999. "User modeling in adaptive dialogue management". *6th European Conference on Speech Communication and Technology, 5-9 September 1999 Budapest, Hungary.* Pp. 1183-1186.

Virzi, R.A. 1992. "Refining the test phase of usability evaluation: How many subjects is enough?". *Human Factors,* vol. 34, pp. 457-468.

Wærn, A.C.M. 2004. "User Involvement in Automatic Filtering: An Experimental Study". *User Modeling and User-Adapted Interaction,* vol. 14, no. 2, pp. 201-237.

Wahlster, W. & Kobsa, A. 1989, "*User Models in Dialog Systems*". New York, USA: Springer-Verlag. Pp. 4–24.

Waibel, A. & Lee, K.-. 1990. "Problems and opportunities". I*n Readings in Speech Recognition.,* edited by. A. Waibel & K.-. Lee. California, USA: Morgan Kaufmann Publishers, Inc. Pp. 7.

Walker, M.A., Litman, D.J., Kamm, C.A. & Abella, A., 1997. "PARADISE: a framework for evaluating spoken dialogue agents." *35th Annual Meeting of the Association for Computer Linguistics, 7-12 July 1997 Madrid, Spain.* Pp. 271–280.

Walker, M. A., Litman, D. J., Kamm, C. A. & Abella, A., 1998. "Evaluating spoken dialogue agents with PARADISE: Two case studies". *Computer Speech and Language*, vol. 12, pp. 317-347.

Weinschenk, S. & Barker, D.T. 2000. "*Designing effective speech interfaces*". New York, USA: John Wiley & Sons, Inc. Pp. 187.

Witten, I. H., Moffat, A. & Bell, T. C. 1999. "*Overview. In Managing Gigabytes - Compressing and Indexing Documents and Images".* 2nd ed. San Francisco, USA: Morgan Kaufmann Publishers.

Wittenbrink, H. 2005. "*RSS and ATOM: Understanding and implementing content feeds and syndication".* 1st ed. USA: Packt Publishing. Chapter 1.

Zhong, Y. & Gilbert, J. E. 2005, "A Context-Aware Language Model for Spoken Query Retrieval". *International Journal of Speech Technology,* vol. 8, no. 2, pp. 203-219.

Zirkle, P. 2003. "*Introduction to APIs and the event loop*". Available from: http://www.skyesurfer.net/keless/vgp/assignment1.htm [Accessed 18th September 2006]

Appendix A: Publications

Hanna, P., O'Neill, I., Wootton, C. & McTear, M. 2007. "Promoting extension and reuse in a spoken dialog manager: An evaluation of the Queen's Communicator". *ACM Transactions on Speech and Language Processing*, vol. 4, no. 3.

Wootton, C., McTear, M. & Anderson, T. 2007. "Utilizing online content as domain knowledge in a multi-domain dynamic dialogue system". *10th European Conference on Speech Communication and Technology, InterSpeech 2007, 27-31 August 2005 Antwerp, Belgium*. Pp. 122-125.

Appendix B: Evaluation Material

Evaluation Design

System 1 (16)																System2 (16)															
Experienced (8)								Inexperienced (8)								Experienced (8)								Inexperienced (8)							
Old (4)				Young (4)				Old (4)				Young (4)				Old (4)				Young (4)				Old (4)				Young (4)			
Male (2)		Female (2)		Male (2)		Female (2)		Male (2)		Female (2)		Male (2)		Female (2)		Male (2)		Female (2)		Male (2)		Female (2)		Male (2)		Female (2)		Male (2)		Female (2)	
Sc 1 Sc 2 (1)	Sc 2 Sc 1 (1)	Sc 1 Sc 2 (1)	Sc 2 Sc 1 (1)	Sc 1 Sc 2 (1)	Sc 2 Sc 1 (1)	Sc 1 Sc 2 (1)	Sc 2 Sc 1 (1)	Sc 1 Sc 2 (1)	Sc 2 Sc 1 (1)	Sc 1 Sc 2 (1)	Sc 2 Sc 1 (1)	Sc 1 Sc 2 (1)	Sc 2 Sc 1 (1)	Sc 1 Sc 2 (1)	Sc 2 Sc 1 (1)	Sc 1 Sc 2 (1)	Sc 2 Sc 1 (1)	Sc 1 Sc 2 (1)	Sc 2 Sc 1 (1)	Sc 1 Sc 2 (1)	Sc 2 Sc 1 (1)	Sc 1 Sc 2 (1)	Sc 2 Sc 1 (1)	Sc 1 Sc 2 (1)	Sc 2 Sc 1 (1)	Sc 1 Sc 2 (1)	Sc 2 Sc 1 (1)	Sc 1 Sc 2 (1)	Sc 2 Sc 1 (1)	Sc 1 Sc 2 (1)	Sc 2 Sc 1 (1)

Evaluation Schedule

Below is the schedule for a participant evaluation of VoiceBrowse. The evaluation will be carried out using a headset, although the computer will not be facing the participant so that they have no visual reference, mimicking real world usage.

Time	Activity
0 – 15 minutes	Introduction and consent forms Initial questionnaire Demonstration of VoiceBrowse by session leader
15 – 30 minutes	Scenario 1 (6 tasks)
30 – 50 minutes	Questionnaire 1
50 – 80 minutes	30 minutes free VoiceBrowsing. This is to simulate usage over time, improving the status of each participant from 'VoiceBrowse Novice' to 'VoiceBrowse Expert' status.
80 – 90 minutes	Scenario 2 (6 tasks) – These tasks will be similar to scenario 1 so that performance and learnability etc. can be compared between the two tasks
90 – 110 minutes	Questionnaire 2 – This will be the same questionnaire as questionnaire 1 above. The difference in responses will be used to compare usability between VB Novice and VB Expert status.
110 – 120 minutes	Final Questionnaire, to include also comparing real experience with expectations of participants stated at start of evaluation.

Pre-Screening questionnaire

How would you estimate your skills in using PCs/Notebooks?	How would you estimate your skills regarding getting information on the Internet (e.g. news pages, informational pages):	How would you estimate your skills regarding programming (e.g. C++, HTML, Delphi, Java, Perl, php, XML ...)	Knowledge (overall)
1= very few	1= very few	1= very few	Criterion:
2= few	2= few	2= few	
3= medium	3= medium	3= medium	Sum of all Items <6
4= good	4= good	4= good	= inexperienced
5= very good	5= very good	5= very good	
			Sum of all Items > = 6
			= experienced

Subject-No.: _____

Initial Questionnaire

Sex: ☐ male ☐ female

Age: ____ years

1. Do you have Internet at home?

   ☐ Yes ☐ No

2. Have you ever used a spoken dialog system?

   (e.g. Telephone banking, mobile phone mailbox, ticket ordering system)

   ☐ Yes ☐ No

2.1 If yes, what kind of system(s)?

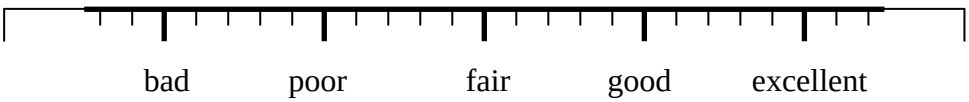
_____

_____

_____

Evaluation of the Interaction

1. Overall impression of the interaction with VoiceBrowse:



2. Achievement of Goals:

2.1 The system did not always do what I wanted.

strongly disagree ☐ disagree ☐ undecided ☐ agree ☐ strongly agree ☐

2.2 The information provided by the system was clear.

strongly disagree ☐ disagree ☐ undecided ☐ agree ☐ strongly agree ☐

2.3 The information provided was incomplete.

strongly disagree ☐ disagree ☐ undecided ☐ agree ☐ strongly agree ☐

2.4 Web browsing can be done efficiently with the system.

strongly disagree ☐ disagree ☐ undecided ☐ agree ☐ strongly agree ☐

2.5 The system is unreliable.

strongly disagree ☐ disagree ☐ undecided ☐ agree ☐ strongly agree ☐

2.6 The system provided the desired information.

strongly disagree	disagree	undecided	agree	strongly agree
☐	☐	☐	☐	☐

3. Communication with the System:

3.1 I felt that the system understood me well.

strongly disagree	disagree	undecided	agree	strongly agree
☐	☐	☐	☐	☐

3.2 I always knew what to say to the system.

strongly disagree	disagree	undecided	agree	strongly agree
☐	☐	☐	☐	☐

3.3 I had to concentrate to understand what the system was saying.

strongly disagree	disagree	undecided	agree	strongly agree
☐	☐	☐	☐	☐

3.4 The system voice sounded natural.

strongly disagree	disagree	undecided	agree	strongly agree
☐	☐	☐	☐	☐

3.5 The information was always provided in a meaningful way.

strongly disagree	disagree	undecided	agree	strongly agree
☐	☐	☐	☐	☐

3.6. The system always presented the right amount of information.

strongly disagree	disagree	undecided	agree	strongly agree
☐	☐	☐	☐	☐

4. Behaviour of the System:

4.1 The system responded too slowly.

strongly disagree	disagree	undecided	agree	strongly agree
☐	☐	☐	☐	☐

4.2 The system is friendly.

strongly disagree	disagree	undecided	agree	strongly agree
☐	☐	☐	☐	☐

4.3 The system did not always do what I expected.

strongly disagree	disagree	undecided	agree	strongly agree
☐	☐	☐	☐	☐

4.4 The system made a lot of errors.

strongly disagree	disagree	undecided	agree	strongly agree
☐	☐	☐	☐	☐

4.5 I was able to recover easily from errors.

strongly disagree	disagree	undecided	agree	strongly agree
☐	☐	☐	☐	☐

4.6 The system reacted like a human.

strongly disagree	disagree	undecided	agree	strongly agree
☐	☐	☐	☐	☐

4.7 The system behaved in a cooperative way.

strongly disagree	disagree	undecided	agree	strongly agree
☐	☐	☐	☐	☐

5. Dialog:

5.1 I easily lost track of where I was when interacting with the system.

strongly disagree	disagree	undecided	agree	strongly agree
☐	☐	☐	☐	☐

5.2 The dialogue was jerky.

strongly disagree	disagree	undecided	agree	strongly agree
☐	☐	☐	☐	☐

5.3 I felt in control of the interaction with the system.

strongly disagree	disagree	undecided	agree	strongly agree
☐	☐	☐	☐	☐

5.4 The dialogue was too long.

strongly disagree	disagree	undecided	agree	strongly agree
☐	☐	☐	☐	☐

5.5 The dialogue quickly led to the desired goal.

strongly disagree	disagree	undecided	agree	strongly agree
☐	☐	☐	☐	☐

5.6 The interaction with the system was fast.

strongly disagree	disagree	undecided	agree	strongly agree
☐	☐	☐	☐	☐

6. Personal Impression:

6.1 The interaction with the system was pleasant.

strongly disagree	disagree	undecided	agree	strongly agree
☐	☐	☐	☐	☐

6.2 I felt relaxed.

strongly disagree	disagree	undecided	agree	strongly agree
☐	☐	☐	☐	☐

6.3 A high level of concentration is required when using the system.

strongly disagree	disagree	undecided	agree	strongly agree
☐	☐	☐	☐	☐

6.4 The interaction was fun.

strongly disagree	disagree	undecided	agree	strongly agree
☐	☐	☐	☐	☐

6.5 Overall, I am satisfied with the system.

strongly disagree	disagree	undecided	agree	strongly agree
☐	☐	☐	☐	☐

6.6 The interaction was boring.

strongly disagree	disagree	undecided	agree	strongly agree
☐	☐	☐	☐	☐

6.7 The interaction was repetitive.

strongly disagree	disagree	undecided	agree	strongly agree
☐	☐	☐	☐	☐

6.8 The interaction was frustrating.

strongly disagree	disagree	undecided	agree	strongly agree
☐	☐	☐	☐	☐

7. Usability of the System:

7.1 The system is difficult to use.

strongly disagree	disagree	undecided	agree	strongly agree
☐	☐	☐	☐	☐

7.2 It is easy to learn to use the system.

strongly disagree	disagree	undecided	agree	strongly agree
☐	☐	☐	☐	☐

7.3 Web browsing via speech was comfortable.

strongly disagree	disagree	undecided	agree	strongly agree
☐	☐	☐	☐	☐

7.4 The system is too inflexible.

strongly disagree	disagree	undecided	agree	strongly agree
☐	☐	☐	☐	☐

7.5 The system is not helpful for browsing the web.

strongly disagree	disagree	undecided	agree	strongly agree
☐	☐	☐	☐	☐

7.6 I prefer to browse the web in a different way.

strongly disagree	disagree	undecided	agree	strongly agree
☐	☐	☐	☐	☐

7.7 I would use the system again in the future.

strongly disagree	disagree	undecided	agree	strongly agree
☐	☐	☐	☐	☐

7.8 Using the system was worthwhile.

strongly disagree	disagree	undecided	agree	strongly agree
☐	☐	☐	☐	☐

7.9 The functionality I would look for in such a system is provided by VoiceBrowse.

strongly disagree	disagree	undecided	agree	strongly agree
☐	☐	☐	☐	☐

8. Importance of System Aspects

Below, please label how important each of the aspects below are for your overall impression of the interaction by dividing the pie accordingly. You can allocate to each aspect as many pieces of the pie as you wish. More pie pieces for one aspect would mean that this aspect was more important for your rating of the interaction. All pieces added together should make up one whole pie.

Aspects:

A) System Personality (interaction style, friendliness, etc.)

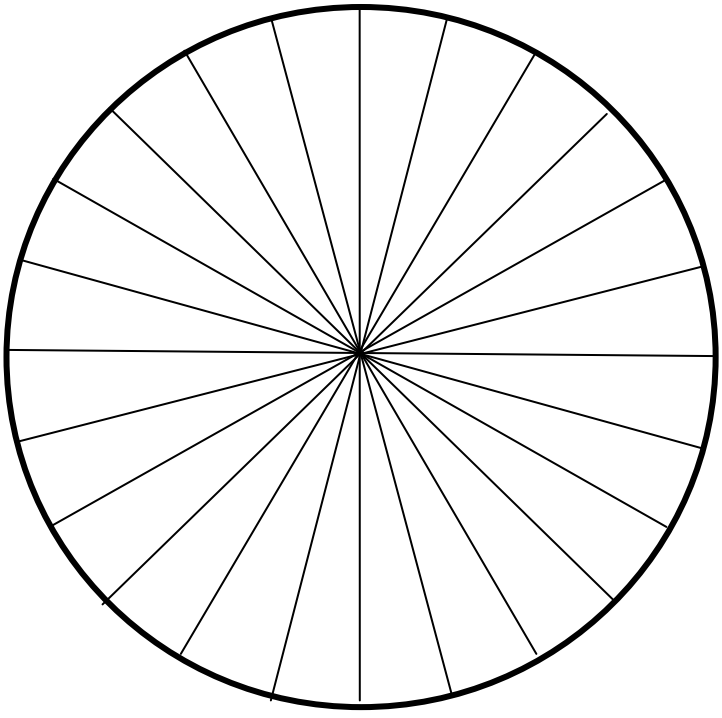
B) Output quality (intelligibility of system speech and quality of the voice)

C) Input quality (understanding errors, ease of input)

D) Learnability (quick and easy to learn)

E) Intuitivity (intuitively usable without learning)

F) Efficiency (get tasks done quickly)



Evaluation of the System

F.1 Overall impression of the VoiceBrowse system:

bad poor fair good excellent

F.2 I would use VoiceBrowse in special situations only.

no ☐ yes ☐

If yes, in which situations?_____

F.3 Browsing the web via speech in comparison to a graphical interface is:

more difficult easier

F.4 What did you like most when using VoiceBrowse? _____

_____

_____

F.5 What did you find most troublesome when using VoiceBrowse? _____

_____

_____

F.6 What suggestions do you have for improving the system? _____

_____

_____

F.7 Were you always aware of the back and next functions? _____

_____

_____

Thank you for participating!

Evaluation scenarios

Scenario 1

You have just returned home from a long plane journey and you are quite tired. However, you do want to check the news headlines before going to bed, as you have not heard any news whilst on the flight.

Task 1: Firstly, you are interested in hearing the Northern Ireland news headlines.

Task 2: You then wish to check the entertainment news headlines. You become interested in the story about the sequel to "Iron Man", and ask for more details about this.

Task 3: You then remember that in business news, oil prices were anticipated to reach new record levels today. Use VoiceBrowse to find out if they did or not.

Task 4: While on your flight today, you heard that the airliner currently has a sale on Belfast routes. You enjoyed the flight, and are required to make the same flight again in a few weeks time. Use VoiceBrowse to book a flight from Belfast to Manchester on the 22nd June 2008, returning on the 26th June 2008.

Task 5: You remember it is a friend's birthday at the end of the week, and your friend enjoys movies. Use VoiceBrowse to search eBay for a list of DVDs.

Task 6: Finally, before going to bed, you wish to hear the top 6 technology stories from provider PC PRO.

Scenario 2

Tonight you are having friends round for dinner. You have been away from the computer for quite some time, due to preparing dinner, cleaning the house and then showering and dressing for the party. Before your friends arrive however, you are interested in retrieving some information online, cannot use the computer as you have still to prepare the last stages of the meal. You use VoiceBrowse to attempt the following 6 tasks.

Task 1: You are interested on hearing the days' sports headlines.

Task 2: You then wish to check the football headlines from Sky Sports. You become interested in the story regarding Thierry Henry's future, so ask for more details about this story.

Task 3: You heard briefly today that the Labour party had a poor performance in the local election. You wish to find out what their leader, Gordon Brown, had to say about this.

Task 4: You know that flights for a particular route to Glasgow have just came on sale that day, and wish to book a flight on this route as early as possible to get the best fare. Use VoiceBrowse to book a flight from Belfast to Glasgow on the13th June 2008, returning on the 16th June 2008.

Task 5: One of your friends coming for dinner is interested in video games, and asked you to get a price for a particular console of eBay. Use VoiceBrowse to search eBay for a list of playstations.

Task 6: Finally, before your friends arrive, you wish to hear the top 5 entertainment stories.

Appendix C: Evaluation Results

Interaction Parameters (Quantitative) Results

Tables A and B overleaf show the Interaction Parameters recorded by the 32 participants during their evaluations. Table A shows the Interaction Parameters during the Untrained phase of the evaluation, and Table B shows the Interaction Parameters during the Trained phase of the evaluation. In both Tables A and B, the top sixteen participants are using System 1, and the bottom sixteen participants are using System 2. The legend used to describe the Interaction Parameters in the top row of both tables is:

sd – Scenario Duration	t – Scenario Time	ds – Number of Disambiguate States	to – Number of time Outs
std – System Turn Duration	nqs – Number of New Query States	ts - Task Success	no – Number of No Matches
ttd – User Turn Duration	fsr – Number of Feed Success States	bi – Number of Barge-Ins	hr – Number of Help Requests

	DIALOGUE 1 (UNTRAINED)											
user	**sd**	**std**	**utd**	**t**	**nqs**	**fsr**	**ds**	**ts**	**bi**	**to**	**no**	**hr**
15	681860	8622	1348	88	8	6	1	1	13	6	4	5
17	494280	7566	1320	66	8	6	0	1	3	1	2	0
8	290880	8065	1384	43	7	4	0	1	6	0	3	1
30	373460	7122	1569	52	10	4	0	1	2	1	5	0
1	489601	7407	1005	75	5	5	2	1	10	0	2	0
2	561200	11025	1368	87	12	6	1	1	14	1	7	0
6	372460	6600	1008	59	9	6	0	1	5	0	3	0
5	459920	6951	1053	72	10	6	0	1	10	2	4	2
25	471490	8303	1032	55	6	6	0	1	0	1	0	0
28	427700	8040	1233	55	6	6	0	1	5	3	0	0
26	448490	7537	1108	59	7	6	0	1	1	0	1	0
18	515820	7229	933	76	6	5	0	1	14	5	1	6
19	503730	8273	809	68	7	6	0	1	10	3	2	3
12	483590	7402	982	70	8	5	4	1	5	2	2	0
9	445620	8128	1000	58	9	4	1	1	3	3	2	0
22	473060	7534	857	62	9	6	0	1	5	5	1	0
7	459884	7364	1608	32	5	0	0	1	1	0	0	0
24	458891	6906	1863	51	6	0	0	1	2	1	0	0
29	457899	6377	3349	45	6	0	0	1	0	1	0	0
31	456906	6798	1388	34	5	0	0	1	2	1	0	0
23	455913	6691	1553	36	6	0	0	1	0	0	0	0
10	454920	5418	1752	46	6	0	0	1	2	0	0	0
3	453928	8239	1211	56	6	0	0	1	5	1	1	0
4	452935	8299	892	37	5	0	0	1	2	2	0	0
13	451942	7597	2335	52	6	0	0	1	1	3	1	0
27	450950	6619	3401	45	6	0	0	1	0	1	0	0
14	449957	5898	2664	43	8	0	0	1	1	1	1	0
32	448964	7363	2849	34	6	0	0	1	0	2	0	0
11	447971	6907	2133	41	6	0	0	1	2	0	0	0
20	446979	6331	2812	52	6	0	0	1	1	1	0	0
16	445986	6037	1439	42	7	0	0	1	1	0	0	0
21	444993	5924	1075	27	4	0	0	1	2	1	0	0

Table A: Interaction Parameters for 32 Users During Untrained Dialogue

	DIALOGUE 2 (TRAINED)											
user	**Sd**	**std**	**utd**	**t**	**nqs**	**fsr**	**ds**	**ts**	**bi**	**to**	**no**	**hr**
15	313500	6538	1330	56	3	6	4	1	12	0	1	0
17	304620	7015	945	55	5	4	8	1	13	1	2	0
8	387300	8640	1138	52	7	5	3	1	4	0	1	1
30	298760	7913	988	44	5	5	2	1	6	0	0	0
1	387380	6635	1400	70	1	6	1	1	10	1	0	0
2	308580	5970	1063	54	5	6	2	1	9	0	0	0
6	459780	7578	813	65	8	6	3	1	4	1	2	0
5	453910	6638	1096	78	6	6	5	1	18	1	4	0
25	254788	7917	624	39	5	5	1	1	1	0	0	0
28	335760	9592	1077	44	4	5	2	1	7	0	0	0
26	477520	7055	1250	69	8	6	4	1	6	0	1	0
18	249430	7388	849	42	1	5	3	1	9	0	1	0
19	466520	6820	806	74	8	5	4	1	15	2	3	1
12	334670	6914	746	56	7	5	2	1	12	0	1	0
9	438660	7803	999	56	5	4	1	1	2	0	0	0
22	389620	7834	844	55	7	6	2	1	6	1	0	0
7	308670	6241	1863	44	6	0	0	1	1	0	0	0
24	267640	6030	1787	41	6	0	0	1	1	0	0	0
29	246150	6476	1470	37	5	0	0	1	1	1	0	0
31	209430	6304	1377	33	5	0	0	1	1	0	0	0
23	201000	5658	2147	33	5	0	0	1	2	0	0	0
10	276260	5926	1548	45	6	0	0	1	2	0	0	0
3	322170	6922	1083	48	6	0	0	1	4	0	1	0
4	322420	6520	1396	50	7	0	0	1	4	0	0	0
13	357590	6716	1957	48	6	0	0	1	0	0	0	0
27	215220	6411	2380	32	5	0	0	1	1	0	0	0
14	346280	6267	2150	50	7	0	0	1	1	0	0	0
32	275700	5759	1476	44	6	0	0	1	0	0	0	0
11	332070	6295	2070	48	6	0	0	1	1	0	0	0
20	300460	5709	2857	46	6	0	0	1	2	0	0	0
16	343890	5452	1844	53	6	0	0	1	3	1	0	0
21	261660	4772	1798	47	7	0	0	1	3	1	0	0

Table B: Interaction Parameters for 32 Users During Trained Dialogue

Qualitative (Questionnaire) Results

Tables C and D overleaf show the qualitative data collected from questionnaires presented to the 32 participants. Table C shows the participants' answers after the Untrained phase of the evaluation, and Table D shows the participants' answers after the Trained phase of the evaluation. The first sixteen participants represent those interacting with System 1, and the second set of sixteen participants represent those interacting with System 2.

	15	**17**	**8**	**30**	**1**	**2**	**6**	**5**	**25**	**28**	**26**	**18**	**19**	**12**	**9**	**22**	**7**	**24**	**29**	**31**	**23**	**10**	**3**	**4**	**13**	**27**	**14**	**32**	**11**	**20**	**16**	**21**
Sex	M	M	F	F	M	M	F	F	M	M	F	F	M	M	F	F	M	M	F	F	M	M	F	F	M	M	F	F	M	M	F	F
Age	61	60	63	55	25	24	25	25	68	64	62	61	26	23	21	22	66	58	60	71	23	23	24	29	70	65	66	70	25	25	30	20
System	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
2.1	4	3	3	4	2	2	2	2	2	2	4	4	2	2	3	2	3	2	3	4	2	2	3	5	4	2	3	3	3	5	5	4
2.2	4	4	4	4	5	5	5	5	4	4	1	4	4	3	4	4	4	4	2	4	4	4	5	5	4	4	3	3	4	4	5	4
2.3	2	1	2	1	2	3	3	2	4	2	2	2	2	2	2	3	2	2	2	2	4	3	2	2	2	4	4	3	2	1	2	2
2.4	4	4	4	3	3	5	3	3	4	4	4	4	4	4	4	3	4	4	3	4	4	4	4	4	4	4	4	4	4	5	4	5
2.5	4	3	4	4	4	4	4	4	4	3	5	4	4	4	3	3	4	4	3	4	4	4	5	4	5	3	4	4	4	4	4	5
2.6	4	4	5	4	5	4	4	4	4	4	5	4	5	3	4	3	5	4	4	5	3	4	5	4	4	3	4	3	4	5	4	4
3.1	4	3	4	4	3	3	2	2	3	4	4	1	4	3	3	2	4	4	3	4	4	4	4	4	4	4	4	4	4	5	4	4
3.2	2	4	3	4	4	5	1	2	2	2	4	1	2	4	2	2	4	4	2	3	2	5	5	4	3	3	4	2	4	4	3	3
3.3	4	2	4	2	1	1	5	4	4	4	4	4	5	2	4	4	4	4	5	4	5	2	2	2	4	4	4	4	2	4	2	3
3.4	4	3	4	4	4	4	4	2	4	2	4	3	3	4	3	2	4	3	1	4	2	2	3	4	4	3	4	2	3	3	4	3
3.5	4	4	4	4	5	5	4	4	4	4	4	4	4	4	3	2	4	4	2	4	2	4	3	5	4	4	4	3	4	4	4	4
3.6	4	3	4	4	4	5	3	4	4	4	4	4	4	3	4	2	4	4	3	4	2	2	3	4	4	3	4	3	3	4	4	4
4.1	4	2	2	2	4	4	2	1	2	3	2	3	2	3	4	3	3	2	2	5	5	2	2	2	1	2	2	2	3	2	2	2
4.2	4	3	4	4	5	5	5	4	4	4	4	2	4	4	4	2	4	4	3	4	3	3	3	4	4	3	4	4	3	5	4	4
4.3	2	4	3	4	2	1	2	2	3	2	4	2	2	2	3	2	3	2	2	4	4	4	4	4	4	4	2	2	3	5	4	4
4.4	5	3	4	4	3	4	4	4	4	4	4	4	4	4	3	3	4	4	4	5	4	3	5	4	5	3	4	4	4	5	4	4
4.5	4	4	4	4	3	5	4	2	4	3	4	4	3	3	4	3	4	4	2	3	1	4	4	4	4	4	2	2	3	4	4	3
4.6	4	2	4	3	2	5	3	3	4	4	4	3	2	5	3	2	4	4	2	4	2	3	4	3	4	3	4	3	4	3	3	4
4.7	4	4	4	3	4	4	5	4	4	4	4	2	4	4	4	3	4	4	3	4	3	4	5	3	4	3	3	3	4	5	4	4
5.1	2	2	2	2	2	1	2	4	3	4	4	4	2	3	2	3	2	4	4	4	4	2	1	2	2	3	4	4	2	4	2	2
5.2	1	3	2	2	2	1	2	2	4	4	2	4	3	2	4	4	2	4	4	2	3	3	2	2	1	2	2	4	2	2	3	2

5.3	4	4	4	4	4	5	2	3	3	2	3	3	4	4	3	3	4	4	1	3	3	4	4	3	4	3	3	2	4	4	4	4
5.4	5	3	4	4	4	5	4	4	4	4	4	2	5	2	3	3	4	4	4	4	2	4	5	4	5	3	4	2	3	4	4	4
5.5	4	4	5	3	4	5	4	3	4	4	4	3	4	3	2	3	4	4	2	4	4	4	5	4	4	4	4	3	4	4	4	4
5.6	4	4	4	4	4	5	3	4	4	3	4	2	4	4	2	2	4	4	4	5	2	4	4	4	4	4	4	2	2	4	4	4
6.1	4	4	5	4	4	5	5	4	4	3	4	4	3	5	3	3	5	4	2	4	3	4	4	4	4	3	4	4	4	5	4	4
6.2	4	4	4	4	2	5	4	4	4	4	3	3	2	3	3	2	5	4	2	3	2	4	4	4	4	2	2	2	4	4	4	4
6.3	2	2	2	2	2	5	2	3	2	2	2	2	2	2	2	2	4	2	2	2	2	4	3	3	4	2	2	2	3	2	4	4
6.4	4	4	5	4	4	5	4	4	3	3	4	3	2	4	3	2	5	4	2	3	4	3	4	4	4	3	3	3	3	4	3	4
6.5	4	3	5	4	4	5	4	3	4	3	4	4	4	3	4	3	5	4	3	5	4	4	5	5	4	3	4	3	4	5	4	5
6.6	4	2	1	2	1	1	1	1	2	2	2	2	2	1	2	4	2	2	2	2	3	3	2	1	2	3	2	4	3	1	2	1
6.7	2	3	2	2	4	2	4	4	4	4	2	4	2	3	3	4	2	4	4	3	4	4	3	1	2	3	2	4	3	2	2	2
6.8	2	2	2	2	2	3	4	2	2	3	2	2	2	3	3	4	2	2	2	2	2	2	2	1	1	3	3	4	2	1	2	2
7.1	4	4	4	4	5	5	4	4	4	4	4	4	4	4	4	3	4	4	3	3	4	5	4	5	4	3	3	3	4	5	4	4
7.2	4	4	5	4	5	5	4	4	4	4	1	4	5	3	4	3	5	4	3	3	4	5	4	4	4	3	4	3	5	5	5	4
7.3	4	3	5	4	4	5	4	4	4	3	4	4	4	4	4	3	5	4	2	4	4	4	3	4	4	3	4	3	4	5	4	4
7.4	2	3	2	2	4	3	2	3	2	3	2	2	2	4	3	3	2	2	3	2	3	3	4	1	2	3	3	3	2	1	2	2
7.5	2	2	2	2	2	1	1	1	2	3	2	2	1	1	2	3	2	2	3	2	2	2	1	2	2	3	2	2	1	1	2	1
7.6	2	3	2	3	4	3	4	4	3	4	2	2	4	3	3	4	2	4	4	2	4	3	4	4	2	4	3	4	2	4	2	2
7.7	4	5	4	4	4	3	3	4	4	4	4	4	3	4	4	3	5	4	2	4	4	4	4	4	4	2	3	3	4	4	4	4
7.8	4	4	4	4	4	3	4	3	4	4	4	4	4	5	3	4	5	4	2	4	4	4	5	4	4	3	4	3	4	5	5	4
7.9	4	4	5	3	4	5	3	5	3	4	5	4	4	3	4	2	5	4	3	4	5	4	3	4	4	3	4	3	4	5	5	4
8a	3	8	5	4	2	2	2	1	5	3	5	3	3	1	3	2	5	2	4	5	5	2	3	3	5	4	4	4	3	3	3	4
8b	5	5	4	4	5	4	1	4	5	5	6	5	3	1	2	3	5	5	4	4	5	4	4	4	3	2	4	4	3	4	4	4
8c	3	4	5	4	3	2	7	5	4	4	3	3	4	7	5	4	4	3	3	3	5	6	2	4	0	3	4	4	5	3	3	4
8d	5	4	6	4	4	7	5	6	4	6	3	5	4	3	6	4	4	6	6	5	5	2	3	5	5	2	4	4	5	5	5	4
8e	3	2	2	3	4	6	5	3	3	3	2	4	5	5	4	5	3	2	3	1	2	2	7	3	4	5	4	5	3	5	5	4
8f	5	1	2	4	6	3	4	5	3	3	5	4	5	7	4	6	3	6	4	5	2	8	5	5	7	8	4	3	5	4	4	4

Table C: Qualitative Data After Untrained Scenario

	15	17	8	30	1	2	6	5	25	28	26	18	19	12	9	22	7	24	29	31	23	10	3	4	13	27	14	32	11	20	16	21
2.1	5	2	4	4	2	3	3	2	3	3	4	4	4	2	2	3	4	4	4	4	2	4	3	4	4	4	4	3	3	5	4	4
2.2	5	5	5	4	4	5	5	5	4	4	4	4	4	5	4	3	5	4	3	4	3	4	5	5	4	4	4	4	3	5	4	4
2.3	1	2	2	2	3	3	1	2	2	2	2	2	1	2	2	3	1	2	2	2	3	2	2	2	2	3	2	3	2	2	2	2
2.4	5	3	4	5	2	4	4	4	4	4	5	4	5	4	3	2	4	4	3	4	2	3	4	4	4	4	4	3	4	5	4	4
2.5	5	4	4	5	4	4	4	4	4	3	4	3	4	4	4	3	4	4	3	5	2	4	4	5	4	4	4	3	4	5	4	4
2.6	5	4	5	4	4	5	5	4	4	4	4	4	5	3	4	3	5	4	4	5	3	4	4	4	4	4	4	3	4	5	4	4
3.1	1	3	5	4	2	4	4	4	2	3	3	4	5	3	3	2	4	4	4	5	2	4	3	4	4	4	4	4	4	4	4	4
3.2	4	3	5	4	4	5	4	3	3	4	3	4	3	4	2	2	4	4	2	3	2	4	4	4	4	4	4	3	3	4	3	4
3.3	4	3	2	4	2	1	4	4	4	4	4	5	4	3	4	4	2	4	4	5	2	2	2	1	2	3	2	4	3	4	3	2
3.4	4	2	4	3	3	4	3	2	4	3	4	3	3	4	2	2	4	4	2	4	3	2	3	4	4	3	4	4	2	4	3	3
3.5	5	3	4	4	4	5	4	4	4	4	4	2	4	4	3	3	4	4	3	4	2	4	3	4	4	4	4	4	4	4	4	4
3.6	5	3	5	4	2	5	4	3	4	4	4	3	5	4	4	3	5	4	4	4	2	3	4	4	4	3	4	4	3	2	4	4
4.1	1	2	2	2	2	3	2	2	3	2	2	2	1	3	4	4	3	2	2	2	2	2	3	1	2	2	2	2	2	2	2	2
4.2	5	4	4	4	3	5	4	4	4	4	4	3	4	5	4	2	4	4	3	4	3	3	3	4	4	4	4	3	4	4	4	4
4.3	5	4	4	4	2	2	4	2	3	3	4	4	4	2	3	4	4	4	2	4	2	4	3	4	4	4	4	3	3	4	4	3
4.4	5	3	4	4	2	1	4	4	4	4	4	4	4	4	3	4	4	4	4	5	4	4	5	5	4	4	4	4	4	5	4	4
4.5	5	4	4	4	2	5	5	4	4	4	4	4	4	4	4	4	4	4	3	3	2	3	4	4	4	4	4	3	3	4	4	4
4.6	5	3	4	3	2	4	3	2	4	4	2	2	3	3	2	2	4	4	2	4	2	3	3	4	4	3	4	4	3	3	3	4
4.7	5	4	4	4	4	4	5	4	4	4	4	2	4	3	3	3	4	4	3	5	3	4	4	4	4	4	4	4	4	4	4	4
5.1	1	2	2	2	2	1	2	2	2	3	2	2	1	2	2	2	2	2	4	3	4	2	3	2	2	3	4	3	2	1	2	2
5.2	1	2	2	2	4	1	2	2	2	4	2	3	2	2	4	4	2	4	3	1	3	2	2	2	2	2	2	2	3	1	3	2
5.3	5	4	4	4	2	5	4	4	3	4	4	4	5	3	3	4	4	4	2	4	2	4	3	4	4	4	4	3	4	4	4	4
5.4	5	3	4	4	4	4	4	4	4	2	4	3	5	4	3	2	4	4	4	4	2	4	5	4	4	3	4	2	3	4	4	4

5.5	5	4	5	4	4	5	4	3	4	4	4	4	4	3	3	3	4	4	4	5	3	4	4	4	4	4	4	3	4	4	4	4
5.6	5	3	4	3	2	5	4	4	4	3	4	4	4	4	2	3	4	4	4	4	2	4	3	4	4	4	4	4	3	4	4	4
6.1	5	4	4	4	3	5	4	4	4	4	4	3	4	5	3	2	4	4	3	4	4	3	3	4	4	4	4	4	3	5	4	4
6.2	5	4	4	4	4	5	4	4	4	4	3	4	4	4	3	3	5	4	2	3	4	4	4	4	4	4	4	2	4	5	4	4
6.3	2	3	2	2	2	4	2	3	2	2	2	2	2	2	2	2	4	4	2	2	3	4	4	4	4	3	4	2	3	4	3	4
6.4	5	4	5	4	4	5	4	4	3	3	4	3	3	4	3	2	5	4	2	4	4	3	4	4	4	3	4	3	3	4	4	4
6.5	5	3	4	4	4	5	4	4	4	3	4	4	4	4	4	3	5	4	3	5	4	4	4	5	4	3	4	3	4	5	5	5
6.6	1	2	2	1	1	2	1	2	2	2	2	1	3	2	2	4	2	2	2	2	2	2	3	1	2	3	2	2	3	1	2	2
6.7	1	2	2	2	4	2	2	4	4	3	2	3	2	4	4	4	2	4	2	2	4	4	2	1	2	3	2	2	2	2	2	2
6.8	1	3	2	1	2	2	3	3	2	4	2	2	2	3	3	3	2	2	2	2	2	4	3	1	2	3	2	2	2	1	2	2
7.1	5	4	5	5	5	5	4	4	4	4	4	4	4	3	4	3	4	4	4	4	4	4	5	5	4	4	4	4	4	5	5	4
7.2	5	4	5	4	5	5	5	4	4	4	4	4	5	5	4	3	5	4	4	4	4	4	5	4	4	3	4	4	5	5	5	4
7.3	5	3	5	4	4	5	5	4	4	4	4	4	4	4	3	4	5	4	3	4	4	4	4	4	4	3	4	4	4	5	4	4
7.4	1	2	2	2	4	3	2	3	2	2	2	2	2	4	3	4	2	2	3	1	3	3	3	2	2	3	2	2	2	1	2	2
7.5	1	1	1	2	2	1	1	2	2	3	2	2	2	1	2	2	1	2	2	1	2	2	1	2	2	2	2	2	2	1	2	2
7.6	2	3	2	3	4	4	4	3	3	4	2	2	3	4	3	4	2	4	4	3	4	3	3	4	2	4	3	3	2	4	3	2
7.7	5	5	4	4	4	5	3	4	4	4	4	5	4	5	4	3	5	4	3	4	4	4	4	4	4	3	4	4	3	5	5	4
7.8	5	5	4	4	4	5	4	4	4	4	4	4	5	5	3	4	5	4	3	4	4	4	4	4	4	3	4	4	3	5	5	4
7.9	5	3	5	4	4	5	4	4	3	3	4	4	4	3	4	2	5	4	3	5	5	3	4	5	4	3	4	4	3	5	5	4
8a	3	8	3	4	3	2	3	1	4	5	4	3	3	2	3	1	4	2	4	6	4	2	5	3	3	4	8	4	3	3	3	4
8b	4	5	3	4	3	2	1	4	4	4	5	4	3	1	3	5	4	3	4	2	5	4	5	5	7	4	0	4	3	4	4	4
8c	3	3	3	3	3	3	6	4	4	3	3	5	4	5	5	2	4	2	3	2	5	6	2	5	1	3	4	4	4	3	3	4
8d	3	3	6	5	4	5	6	5	4	6	3	6	4	4	5	7	5	5	6	7	4	2	4	0	4	6	4	5	5	5	5	4
8e	5	3	4	4	4	6	4	5	4	3	5	3	5	5	3	6	4	5	3	4	3	2	2	4	4	4	4	3	4	4	5	4
8f	6	2	5	4	7	6	4	5	4	3	4	3	5	7	5	3	3	6	4	3	3	8	6	7	5	3	4	4	5	5	4	4

Table D: Qualitative Data After Trained Scenario