

A Corpus Based Approach to Generalising a Chatbot System

Bayan Aref Abu Shawar

**Submitted in accordance with the requirements for the degree of
Doctor of Philosophy**



**University of Leeds
School of Computing**

April 2005

The candidate confirms that the work submitted is his/her own and that appropriate credit has been given where reference has been made to the work of others.

This copy has been supplied on the understanding that it is copyright material and that no quotation from the thesis may be published without proper acknowledgement.

Abstract

Chatbot tools are computer programs which interact with users using natural languages. This technology started in the 1960's. Most developers built their systems aiming to fool users that they are talking with real humans. Up to now most chatbots serve as a tool to amuse users through chatting with a robot. However, the knowledge bases of almost all chatbots are edited manually which restricts users to specific languages and domains.

This thesis shows that chatbot technology could be used in many different ways in addition to being a tool for having fun. A chatbot could be used as a tool to learn or to study a new language; a tool to access an information system, a tool to visualise the contents of a corpus; and a tool to give answers to questions in a specific domain. Instead of being restricted to a specific domain or written language, a chatbot could be trained with any text in any language. Some of the differences between real human conversations and human-chatbot dialogues are presented.

A Java program has been developed to read a text from a machine readable text (corpus) and convert it to ALICE chatbot format language (AIML). The program was built to be general, the generality in this respect implies, no restrictions on specific language, domain, or structure. Different languages were tested: English, Arabic, Afrikaans, French, and Spanish. At the same time different corpora structure were used: dialogue, monologue, and structured text.

Chapter 1 is an introduction to thesis objectives and structure. Chapter 2 presents chatbot history, and corpus/computational linguistics. Chapter 3 discusses the corpora used in training the ALICE chatbot system, and chapter 4 presents the corpus-trained chatbots. Chapter 5 deals with evaluation and ideas for future work are presented in chapter 6.

Declaration

I declare that the work presented in this thesis is, the best of my knowledge in this domain, original, and my own work. Most of this thesis has been published in the papers listed below:

1. Abu Shawar B. and Atwell E. 2002. *A comparison between ALICE and Elizabeth chatbot systems*. School of Computing research report 2002.19, University of Leeds.
2. Abu Shawar B. and Atwell E. 2003. *Machine Learning from dialogue corpora to generate chatbots*. Expert Update journal, Vol. 6, No 3, pp 25-30
3. Abu Shawar B. and Atwell E. 2003. *Using dialogue corpora to retrain a chatbot system*. In Archer, D., Rayson, P., Wilson, A. and McEnery, T. (eds.) Proceedings of the Corpus Linguistics 2003 conference. Lancaster University, UK, pp. 681-690.
4. Abu Shawar B. and Atwell E. 2003. *Using the Corpus of Spoken Afrikaans to generate an Afrikaans chatbot*. SALALS Journal: Southern African Linguistics and Applied Language Studies. Vol. 21, pp. 283-294.
5. Abu Shawar B. 2003. *Review of CL2003, the International Conference on Corpus Linguistics*”, ELRA newsletter. Vol. 8, No. 2, pp. 4-6.
6. Abu Shawar B. and Atwell E. 2004. *An Arabic chatbot giving answers from the Qur'an*. In Bel, B & Marlien, I (eds.) Proceedings of TALN04: XI Conference sur le Traitement Automatique des Langues Naturelles. Vol. 2, pp. 197-202.
7. Atwell E.; Al-Sulaiti L.; Al-Osaimi S.; Abu Shawar B. 2004. *A review of Arabic corpus analysis tools*. In Bel, B & Marlien, I (eds.) Proceedings of TALN04: XI Conference sur le Traitement Automatique des Langues Naturelles. Vol. 2, pp. 229-234.
8. Abu Shawar B. 2004. *The NWCL Research Training Programme*. ELNews: The Newsletter of the European Network in Human Language Technologies. Vol. 13.1, pp. 14-15.
9. Abu Shawar B. and Atwell E. 2004. *A chatbot as a novel corpus visualisation tool*. In Lino, M.T., Xavier, M.F., Ferreira, F., Costa, R., Silva, R., Pereira, C., Cavalho, F., Lopes, M., Catarino, M. and Barros, S. (eds.) Proceedings of Fourth International Conference on Language Resources and Evaluation (LREC'04). Vol. VI, pp. 2057-2060.

10. Abu Shawar B. and Atwell E. 2004. *Evaluation of chatbot systems*. In Proceedings of Eight Maghrebian Conference on Software Engineering and Artificial Intelligence.
11. Abu Shawar B. and Atwell E. 2004. *Accessing an information system by chatting*. In Meziane, F. and Metais, E. (eds.) Proceedings of 9th International conference on the Application of Natural Language to Information Systems, NLDB 2004 Salford, UK. LNCS Lecture Notes in Computer Science, Springer, pp. 396-401
12. Abu Shawar B. and Atwell E. 2005. *Modelling turn-taking in a corpus-trained chatbot*. In Fisseni B., Schmitz H., Schröder B. and Wagner P. (eds.), Sprachtechnologie, Mobile Communication und Linguistische Ressourcen (GLDV05). Peter Lang Verlag, Bonn, pp. 549-561.
13. Abu Shawar B., Atwell E. and Roberts, A. 2005. *FAQchat as an information retrieval system*. In: Zygmunt V. (ed.), *Human Language Technologies as a Challenge for Computer Science and Linguistics: Proceedings of the 2nd Language and Technology Conference*, Wydawnictwo Poznanskie, Poznan, pp. 274-278.
14. Abu Shawar B and Atwell E. 2005. *A chatbot system as a tool to animate a corpus*. ICAME Journal. Vol. 29, pp. 5-23.
15. Abu Shawar B. and Atwell E. 2005. *Using corpora in machine-learning chatbot systems*. International Journal of Corpus Linguistics [forthcoming].

Acknowledgments

"Do not consider even the smallest good deed as insignificant; even meeting your brother with a cheerful face (is a good deed)."

Prophet Mohammed (pbuh)

In the name of God, the most gracious, the most merciful. I would like to begin by thanking my Lord, for providing me with the health, strength and patience, guiding me through all the difficulties to start this work and complete it. Initially, the most difficult problem that I was faced with, was being abroad for the first time. It was a big challenge, and I was uncertain of whether or not I would be able to withstand three years of being far away from my family, my relatives, my friends, and my home. Words can not describe how thankful I am to God for providing me with all the courage I needed to be able to stand alone, and for providing me with all the beloved people whom I met here and who gave me all the care and love I needed.

After my Lord, I would like to express my deepest gratitude to my parents. Thank you for your endless love, for your care, and for being with me in my heart at all times. I'm sorry mum for your tears, so dear to me, which fell from your precious eyes every time I said goodbye. I'm sorry dad for all the moments you felt worried about me during my stay abroad. This research is my gift to you, which I hope will make you happy and proud of your daughter. Thank you my sister Rana for all the lovely jokes and messages you kept sending to cheer me up, thank you for being there whenever I need you. Many thanks for my brothers Mohammed, Osama, and Moath, for their support and encouragement. And for my relatives, who keep asking about me and praying for me. I can not express more than this, how thankful I am to you all.

During the first month of being in the lab, Andy Roberts came to my desk, introduced himself, and said that our supervisor was on holiday, and if I need anything not to hesitate to ask him. It was so nice of him, and it was the first time I met him. Since that time, he has been the first person whom I turn to whenever I had any questions, or even a technical problem, and this is not only because he is so friendly, but also because luckily his desk was beside mine. Thank you Andy for being my friend, for being there throughout the three years. Many

thanks for the lovely time we had in the lab, chatting, laughing, and sharing thoughts and ideas.

It's time to thank all my friends in the lab, many thanks for Latifa Al-Sulati who encouraged me to exercise from time to time instead of sitting in front of the PC for long continuous hours every day. Also I want to thank Karim Djemame, Mohammed Haji, Abdullah Othman, Hazem Al-Hiary, Moath Al-Rajab and Akram Awad for being my brothers and for their company during Friday prayer. I should not forget to thank Savio Pirondini and Pritpal Rehal from the support team, who always solve my PC problems. Many thanks for the School of Computing for all the facilities it provides, and for all people listed in chapter 5, who helped with the evaluation.

Apart from the academic work, I'd like to thank Sabah Mohammed, Khaled Mahmoud, and their kids for the lovely family atmosphere, love and care they provided. My deepest gratitude for Abeer Saleh, for all the happiness and cheerfulness she brought to my heart. Thank you Abeer for all the interesting times we had together. Many thanks to Samar Jambi, for being my friend, and who opened her home to me when there was no place to go. Many thanks for Hala Zeid for being my sister and for sharing the same flat with me for two years, where we shared happiness and sadness, good and bad moments together.

Finally, leaving the best 'till last, thank you my dear supervisor, Eric Atwell, for your inspiration to do this research. It was an honour working with you and having you as a supervisor. All this was not to start or to finish without your support, without the regular excellent supervision and the significant advice you provided. Thank you for your help and your support every time I struggled. In addition to your academic mind, I would like to thank you for your kindness, your concern, and for believing in me to achieve this, which made it both motivating and interesting at the same time. I'm so grateful for all the times we discussed ideas, and all the times you let me in your office without an appointment, even though I was distracting you. My sincere appreciation for the lovely enjoyable times we had.

For all my beloved and even those whom I have not acknowledged, may God bless you and grant you with all the success and happiness you desire.

Contents

Abstract	ii
Declaration	iv
Acknowledgements	vi
Contents	vii
List of tables	xii
List of figures	xiii
List of abbreviations and acronyms.....	xv
1. Introduction	1
1.1 Overview	1
1.2 Chatbots and A.I. disciplines.....	2
1.2.1 Corpus and computational linguistics	2
1.2.2 Information seeking and knowledge acquisition.....	3
1.3 Chatbot evaluation.....	4
1.4 Thesis objectives	4
1.5 Thesis structure.....	6
2. AI Disciplines and Chatbot History	7
2.1 Introduction	7

2.2 Corpus linguistics and computational linguistics	8
2.2.1 Computational linguistics and language engineering.....	9
2.3 Chatbot history	12
2.4 Recent chatbots.....	13
2.5 Chatbot communication language	18
2.6 Chatbots and pattern-matching techniques.....	20
2.6.1 Information retrieval.....	21
2.6.2 Information extraction.....	22
2.6.3 Question answering systems	25
2.6.4 Summary of pattern-matching techniques.....	28
2.7 Machine learning techniques.....	28
2.8 ALICE chatbot system	30
2.8.1 Overview of AIML.....	30
2.8.2 Types of categories.....	32
2.8.3 The recursive categories	32
2.8.4 Preparation for pattern-matching in ALICE.....	35
2.8.5 The ALICE pattern-matching algorithm	37
2.9 Elizabeth system.....	41
2.9.1 Elizabeth's script file format	41
2.9.2 Dynamic processing in Elizabeth.....	42
2.9.3 Elizabeth's pattern-matching algorithm	43
2.9.4 Implementing grammatical rules in Elizabeth.....	45
2.10 A comparison between ALICE and Elizabeth.....	46
2.11 Evaluation methodologies	47
2.11.1 Evaluating spoken language dialogue systems	48
2.11.2 The Loebner Prize	49
2.12 Summary	53

3. Corpora to Train Chatbots.....	55
3.1 Introduction	55
3.1 The Dialog Diversity Corpus	56
3.1.1 MICAS corpus problems.....	57
3.1.2 CIRCLE corpus problems	57
3.1.3 CSPA corpus problems	59
3.1.4 The TRAINS dialog corpus problems.....	60
3.1.5 ICE-Singapore problems	61
3.1.6 Mishler Book.....	62
3.2 The Korpus Gesprooke Afrikaans corpus	62
3.2.1 Afrikaans corpus problems.....	63
3.2.2 Principles adopted to solve the KGA problems	64
3.3 The British National Corpus.....	65
3.3.1 The BNC problems.....	66
3.3.2 The principles adopted to solve the BNC spoken text problems	70
3.4 The Qur'an corpus.....	71
3.4.1 Problems of learning from the Qur'an text	72
3.4.2 The adopted principles to solve the Qur'an problems.....	74
3.5 The SoC Frequently Asked Questions (FAQ) corpus	74
3.5.1 The FAQ problems	75
3.6 Summary	78
4. Retraining ALICE	80
4.1 Introduction	80
4.2 Software developments	81
4.2.1 Software requirements specification	82
4.2.2 Software design and implementation	83
4.2.3 Software testing and evolution	83

4.3 The DDC prototype	85
4.4 The KGA prototype	86
4.4.1 Learning techniques	86
4.4.2 Elaborating the system architecture.....	88
4.5 The BNC prototype	90
4.5.1 The BNC frequency list.....	90
4.5.2 The alteration method.....	91
4.6 The Qur'an prototype	93
4.7 The SoC FAQ prototype (FAQchat)	96
4.8 Summary	99
5. Evaluation	100
5.1 Introduction	100
5.2 Human-to-human versus human-to-chatbot dialogues.....	101
5.2.1 Semantic comparison	102
5.2.2 Part-of-Speech comparison	103
5.2.3 Lexical comparison	104
5.3 Evaluating the Afrikaans version	105
5.3.1 Dialogue efficiency metrics.....	106
5.3.2 Dialogue quality metrics	107
5.3.3 User satisfaction	109
5.3.4 The methodology drawbacks.....	109
5.4 Evaluating the BNC versions	110
5.4.1 Results and discussion.....	110
5.5 Evaluating the Qur'an versions	116
5.5.1 Results and discussion of the Arabic version.....	116
5.5.2 Results and discussion of the English/Arabic version.....	117
5.5.3 Evaluation methodology and results	120

5.6 Evaluating the FAQ chatbot	123
5.6.1 Comparing the FAQchat with Google.....	124
5.6.2 The methodology.....	125
5.6.3 Results and discussions	127
5.6.4 Testing reliability	132
5.6.5 Samples of chatting	135
5.7 Summary	136
6. Conclusions	139
6.1 Summary of the work.....	139
6.2 Results	141
6.3 System drawbacks	143
6.4 Future work	144
6.4.1 Project aims	145
6.4.2 Project plans	145
6.5 Overall conclusions	146
7. References	147
8. Appendix A. FAQchat Evaluation Sheet.....	160

List of tables

Table 1. 1 Summary of system prototypes, corpora used, and the goals of implementations	5
Table 2. 1 Some recent chatbots.....	17
Table 2. 2 The normalisation process in ALICE pattern-matching processing	36
Table 2. 3 Producing the input path by the AIML interpreter.....	37
Table 2. 4 Tracing an Elizabeth output	45
Table 2. 5 A list of the Loebner prize winners from 1991-2004.....	53
Table 4. 1 The trainable corpora.....	82
Table 5. 1 Response type frequencies for “Afrikaana”, Afrikaans chatbot.....	107
Table 5. 2 Response type frequencies: subjective analysis of “Afrikaana” response	108
Table 5. 3 The BNC London teenager and loudmouth chatbots	112
Table 5. 4 Different BNC versions.....	113
Table 5. 5 Characteristics of the English/Arabic Qur’an chatbots	117
Table 5. 6 Proportions of answers generated from the Qur’an.....	121
Table 5. 7 Can you find the answer? Analysed per question	127
Table 5. 8 Proportion of users finding answers by FAQchat and Google.....	128
Table 5. 9 Which tool do you prefer? Analysed per question.....	129
Table 5. 10 The proportion of users’ preference tool, analysed per question	129
Table 5. 11 Number of matches found per user	130
Table 5. 12 The mean for giving answers and preference each user found	131
Table 5. 13 Paired t-Test for finding answers analysed per question and per user..	133
Table 5. 14 Paired t-Test for users preferred tool analysed per question and per user	134

List of figures

Figure 2. 1 The AIML format.....	31
Figure 2. 2 Samples of AIML categories, illustrates the usage of the recursive categories.....	34
Figure 2. 3 Log-plot visualisation of 24,000-state Graphmaster.....	37
Figure 2. 4 An example of an AIML code	39
Figure 2. 5 Tracing of ALICE matching process	40
Figure 2. 6 Elizabeth script command notations	41
Figure 2. 7 An example of Elizabeth script: illustrate different keyword types	44
Figure 2. 8 Grammar representation rules in Elizabeth.....	46
Figure 3. 1 Astronomy transcripts sample/MICASE corpus.....	57
Figure 3. 2 Algebra transcript sample/CIRCLE corpus.....	58
Figure 3. 3 Physics transcript sample/CIRCLE corpus.....	58
Figure 3. 4 CSPA corpus sample.....	59
Figure 3. 5 TRAINS corpus sample	60
Figure 3. 6 ICE-Singapore corpus sample.....	61
Figure 3. 7 Sample of the Korpus Gesprooke Afrikaans corpus.....	63
Figure 3. 8 A sample of a BNC spoken transcript.....	66
Figure 3. 9 A sample representing long monologue in the BNC	67
Figure 3. 10 A sample representing unclear utterances in the BNC.	68
Figure 3. 11 A sample showing the overlapping problem in the BNC	68
Figure 3. 12 Extra-linguistic entities cause problems in the BNC	69
Figure 3. 13 A sample of the Qur'an in Arabic and English.....	73
Figure 3. 14 The extra notations used within FAQ of SoC files for avigation	76
Figure 3. 15 Questions-answers denoted by <DIV CLASS="sect1"> and "<p>" tags.....	76
Figure 3. 16 FAQ questions-answers denoted by "question" and "answer" tags.....	77

Figure 4. 1 A sample of MICASE transcript with its corresponding AIML category	85
Figure 4. 2 An atomic category with its most significant word in 4 positions.....	87
Figure 4. 3 The program algorithm of the KGA prototype.	88
Figure 4. 4 A sample of Kilgariff unlemmatised frequency list.....	90
Figure 5. 1 Screenshot of the semantic comparison between ALICE (01) and CSPA (02)	102
Figure 5. 2 Semantic comparisons between ALICE and CSPA.....	102
Figure 5. 3 Screenshot of PoS comparison between ALICE (01) and CSPA (02) ..	103
Figure 5. 4 Part-of-Speech comparison between ALICE (01) and CSPA (02).....	104
Figure 5. 5 Screenshot of word comparison between ALICE (01) and CSPA (02).	104
Figure 5. 6 Word comparison between ALICE (01) and CSPA (02).....	105
Figure 5. 7 Dialogue efficiency: "Afrikaana" matching types relative probabilities	107
Figure 5. 8 The quality of the dialogue: "Afrikaana" response type relative probabilities	108
Figure 5. 9 A sample of chatting with the BNC chatbot	111
Figure 5. 10 A sample of chatting with Peter-BNC chatbot	113
Figure 5. 11 A sample of chatting with the sport-BNC chatbot.....	114
Figure 5. 12 A sample of chatting with the Arabic Qur'an chatbot	117
Figure 5. 13 A sample of chatting with the English/Arabic Qur'an chatbot.....	118
Figure 5. 14 A sample of matching a whole ayya via chatting with English/Arabic chatbot	119
Figure 5. 15 The Qur'an proportion of each answer type denoted by users	121
Figure 5. 16 FAQchat interface	126
Figure 5. 17 The proportion of finding answers by the FAQchat and Google.....	128
Figure 5. 18 Which tool do you prefer? Summary.....	130
Figure 5. 19 Comparing t-Test values fro finding answers.	133
Figure 5. 20 Comparing t-Test values fro the preferred tool.....	134
Figure 5. 21 Answers generated by FAQchat	135
Figure 5. 22 Answers generated by FAQchat and Google	136

List of abbreviations and acronyms

AI	Artificial Intelligence
ACASD	Automatic Content Analysis of Spoken Discourse
AIML	Artificial Intelligence Mark-up Language
ALICE	Artificial Linguistic Internet Computer Entity
BNC	British National Corpus
m.d.	The difference of mean between two groups
sd.	Standard deviation
K	The total number in the group
d.f.	degree of freedom
T	The value of the paired t-Test
t_{crit}	The critical value of t_{obt}
CIRCLE	Corpus of Interdisciplinary Research on Constructive Learning Environments
CLAWS	Constituent Likelihood Automatic Word-tagging System
CMC	Computer-Mediated Communication
COLT	The Bergen Corpus of London Teenage Language
CSPA	Corpus Spoken Professional American English
DDC	Dialogue Diversity Corpus
DIPRE	Dual Iterative Pattern Relation Expansion
FAQ	Frequently Asked Questions
GATE	General Architecture for Text Engineering
HMM	Hidden Markov Model
ICE	International Corpus of English
ICLE	International Corpus of Learner English
IE	Information Extraction
IR	Information Retrieval

IRC	Internet Relay Chat
ISLE	Interactive Spoken Language Education
KGA	Korpus Gesproke Afrikaans (Corpus of Spoken Afrikaans)
LOB	Lancaster/Oslo-Bergen Corpus
MICASE	Michigan Corpus of Academic Spoken English
ML	Machine Learning
NLP	Natural Language Processing
NLTK	Python Natural Language Toolkit
POS	Part-of-Speech
QA	Question Answering
RBB	Reuters Business Briefing
SLD	Spoken Language Dialogue
SoC	School of Computing
SPAAC	Speech Act Annotated Corpus
VRA	Virtual Research Associates
XARA	XML Aware Retrieval Application. Originally called XARA, and because of copy-right problems. Renamed XAIRA (an XML Aware Indexing and Retrieval Architecture).

1. Introduction

“Vague and nebulous is the beginning of all things, but not their end, And I fain would you remember me as a beginning.”

Gibran Khalil Gibran, “The Prophet”

1.1 Overview

Human-machine conversation is a technology integrating different areas to facilitate communication between users and computers using natural language. A related term to machine conversation is the *chatbot*, a conversational agent that interacts with users turn by turn using natural language. Different chatbots or human-computer dialogue systems have been developed using speech or text, for different purposes such as communication guidance, web site help, and for fun.

However, these chatbots are restricted to the knowledge that is manually “hand-coded” in their files and to the natural language, which is written or spoken. In this thesis an automatic process is presented to escape from these restrictions. The research focuses on a software implementation and the learning techniques that have been adopted to learn new rules. The software is a Java program that was implemented by the author, and has been used to develop different versions of a chatbot that “speak” different languages. The program is able to convert the machine-readable text (corpus) to the chatbot language model format. In this thesis, three main research questions are investigated:

1. What are the differences between real human-human conversations and human-chatbots dialogues, and can these be identified using computational tools?
2. Does the automatic process succeed in extracting input-output rules from a corpus to generate the knowledge base of a chatbot?
3. Is it possible to use an automatically-trained chatbot as a tool to access information and retrieve answers to questions?

The evaluation of the trained chatbots provides answers for these questions. Several techniques have been used such as measuring the “naturalness” of the generated dialogues; measuring the quality and efficiency of the learning techniques; and finally statistical evaluation to investigate the possibility of using the chatbot as a tool to retrieve answers from a frequently asked questions (FAQ) database as discussed in chapter 5.

1.2 Chatbots and AI disciplines

Artificial Intelligence (AI) is defined as “The study of how to make computers do things at which, at the moment, people are better” (Rich and Knight 1991). To do so, computers must be able to understand human language. Designing and building computer systems, which interpret and generate natural languages is a branch of AI known as natural language processing (NLP). Jurafsky and Martin (2000) define language processing as the “computational techniques that process spoken and written human language, as language”. The following sub-sections will present the relationship between chatbots and other AI branches.

1.2.1 Corpus and computational linguistics

Studying processing techniques for language and using these to analyse datasets of text (corpora) is a sub discipline of natural language processing known as corpus linguistics. Corpus linguistics “is not a branch of linguistics such as syntax, semantics and pragmatics that concentrate on describing or explaining some aspect of language use. It is a methodology that can be applied to wide range of linguistic study.” (Rayson 2003). Aijmer and Altenberg (1991) defined corpus linguistics as “the study of language on the basis of text corpora.”

A related field to corpus linguistics is computational linguistics, which focuses on “studying natural languages, just as traditional linguistics does, but using computers as a tool to model (and, sometimes, verify or falsify) fragments of linguistic theories deemed of particular interest.” (Boguraev et al. 1995). The first computational linguistics applications were the machine translation systems, which appeared in the late 1950s. Over decades, the increase in availability of computers, and the widespread use of the Internet, necessitate developing

different systems and tools that facilitate the retrieval, extraction, and searching for information or data within both the Internet and corpora.

One of these computational tools is Wmatrix (Rayson 2002), which provides a data-driven method to compare between different sized corpora at three levels: semantic, Part-of-Speech (POS) and word analysis. In this thesis Wmatrix was used to measure the naturalness of chatbot dialogues by comparing real human dialogues to chatbot ones, as discussed in section 5.2.

1.2.2 Information seeking and knowledge acquisition

Information seeking involves *data retrieval*, more generic *information retrieval*, and most specific *information extraction* processes. A data retrieval process aims to retrieve all objects from a well defined structure (e.g. relational databases) which satisfy clearly the user's defined conditions. Information Extraction (IE) and Information Retrieval (IR) are both related to natural language processing, where computational linguistic techniques and theories play a strong role. IR is the process of retrieving documents, which might be relevant or useful to the user request (Baeza-Yates and Ribeiro-Neto 1999).

Cowie and Lehnert (1996) describe an IE process as a method “for sorting through reams of text, linking relevant information while ignoring the irrelevant”, and Wilks (1997) claims that “any application of this technology is usually preceded by an IR phase”. In this respect, IR systems collect the relevant documents, and then IE systems use such documents to extract relevant information in a coherent framework. Grishman (1997) describes the output of an IE system as “a structured representation (such as data base) of selected information drawn from the text.”

Various learning algorithms and pattern matching techniques were applied to match the user's request with the knowledge base either to retrieve information or to extract specific pieces of data to fill a predefined template. Some of these methods are reviewed in section 2.5. In this thesis, the chatbot is also described as a tool to retrieve information from a structured database and is compared with the Internet search engine Google, to confirm its ability to generate responses, as described in section 5.7.

1.3 Chatbot evaluation

The Turing Test, proposed by Alan Turing (1950), originated the way of evaluating a machine in terms of intelligence. The intelligence of a software system could be evaluated by measuring how well it could emulate a human conversation. More recently, in an annual Loebner Prize contest¹, rival chatbots have been assessed in terms of strictly-defined evaluation criteria. A panel of human judges are given 10 minutes with each chatbot, and asked to rank them in terms of “naturalness”. The Turing Test and Loebner Prize contest are reviewed in section 2.8. Alternative evaluation methods, which are more appropriate to the practical information system applications, are advocated in chapter 5. The evaluation takes account of open-ended trials by real users, rather than artificial 10-minute trials.

1.4 Thesis objectives

The main objective is to build chatbots’ knowledge automatically instead of generating the knowledge-base manually, using a corpus based approach. Training the chatbot involves:

1. Building a general program that is able to extract input-output rules from different corpora using text learning techniques to give the best match.
2. Generating different versions of the chatbot that speak different languages, and serve in multiple domains by training the chatbot using numerous corpora including: unstructured, such as dialogue corpora and structured ones such as questions and answers.
3. Finding useful applications of chatbots in addition to being a tool for amusement such as:
 - Using a chatbot as a tool for unknown languages, where "unknown" means (i) unknown to the chatbot author/developer, and/or (ii) unknown to computational linguistics, that is, where there is a shortage of existing tools to deal with the languages.
 - Using the chatbot as a tool to visualise or animate a corpus.
 - Using the chatbot as a tool to retrieve answers for questions.

¹ <http://www.loebner.net/Prizef/loebner-prize.html>

- Using the chatbot as a tool to access and retrieve information from a corpus.

Despite the fact that existing chatbots are unable to hold the dialogue history (an important characteristic in many human conversations), this thesis shows that chatbots could still be used in many useful ways.

The overall objective, to explore the corpus-based approach to training chatbots, was implemented in a series of prototypes, using different corpora and learning techniques to address a range of sub-goals. Table 1.1 is a summary of system prototypes, training corpora used, and the goals of implementation.

Prototype	Learning technique(s)	Goals
The DDC	Simple approach: the first turn is the pattern and the second is the template.	Exploring the problems of using a dialogue corpus (Abu Shawar and Atwell 2003a)
The KGA	Simple approach, the first word approach, the first most significant word approach.	Conversational machine for a new language (Abu Shawar and Atwell 2003b).
The BNC	Simple approach, the first word approach, the first most significant word, and the second most significant words	Generating a large number of AIML categories (Abu Shawar and Atwell 2004a), and visualising/animating a corpus (Abu shawar and Atwell 2005a).
The Qur'an	The same as the BNC techniques.	Exploring problems in using Arabic language and non-conversational text (Abu Shawar and Atwell 2004b).
The SoC FAQ	Same as the BNC, and the first word with the first most significant word, and the second most significant word.	Exploring the use as a tool to retrieve answers from FAQ databases (Abu Shawar et al. 2005b).

Table 1. 1 Summary of system prototypes, corpora used, and the goals of implementations

1.5 Thesis structure

The remaining chapters of this thesis are as follows:

Chapter 2 presents chatbot history: when it began, the aim of developing such chatbots, and the language used within chatbot systems. Different pattern matching techniques are compared. Two chatbot systems, ALICE and Elizabeth, are compared in terms of knowledge representation and pattern matching techniques. The chapter includes background information on evaluation methods applied within spoken dialogue systems and the Loebner Prize contest which is used to evaluate chatbots. Corpus/computational linguistics fields and language engineering relationships are also reviewed.

Chapter 3 discusses the corpora used in training the ALICE chatbot system. These corpora have many problems, which are made clear in this chapter. Solutions are suggested to overcome these problems.

Chapter 4 describes the software implementation versions, each of which has its own corpus, and its own normalisation methods which cope with the corpus annotations. Different versions of ALICE were generated.

Chapter 5 is an evaluation of the different versions of the retrained ALICE, assessing whether the goals of each version were achieved. The evaluation is based on measuring naturalness, and measuring the success of the adopted learning techniques in giving responses based on users' feedback.

Chapter 6 provides a summary of the thesis and conclusions. The limitations of the work are presented and plans for future work are suggested.

2. AI Disciplines and Chatbot History

“...If you thought you must measure time into seasons, let each season encircle all the other seasons, and let today embrace the past with remembrance and the future with longing.”

Gibran Kalil Gibran, “The prophet”

2.1 Introduction

“Before there were computers, we could distinguish persons from non-persons on the basis of an ability to participate in conversations. But now, we have hybrids operating between person and non persons with whom we can talk in ordinary language.” (Colby 1999a).

Conversational machines have been developed by combining techniques and theories from many areas including natural language processing, corpus linguistics, computational linguistics, natural language engineering and recently multimedia technology, which aims to add animation, graphics, sounds during the man-computer interaction.

Section 2 of this chapter introduces a brief revision about related fields. Section 3 presents chatbots history. In section 4 some of the recent chatbots are discussed. Chatbot language and pattern-matching techniques are described in sections 5 and 6 respectively. Machine learning techniques are presented in section 7. The ALICE chatbot and AIML language are explained in section 8. Section 9 summarises the Elizabeth chatbot. A comparison between ALICE and Elizabeth is reported in section 10. Section 11 reviews different evaluation methodologies of chatbots and dialogue systems.

2.2 Corpus linguistics and computational linguistics

Sinclair (1991) defined the corpus as “a collection of naturally occurring language text, chosen to characterize a state or variety of a language.” Using a corpus based approach to study a language has been known as corpus linguistics.

With the rapid progress of the computer industry, corpora became increasingly available in a machine readable form. Sinclair (1996) defined a *computer corpus* as “a corpus which is encoded in a standardised and homogenous way for open-ended retrieval tasks. Its constituent pieces of language are documented as to their origins and provenance”. The first modern machine readable corpus was the Brown Corpus of Standard American-English (Kučera and Francis 1967). The corpus consists of approximately a million words of written American-English taken from 15 different text types published in 1961. In 1978 a corpus of British English was constructed in the University of Lancaster known as the Lancaster/Oslo-Bergen (LOB) corpus of British-English (Johansson 1986), which matches the Brown corpus in terms of its composition. Both the Brown and the LOB corpora are available for use in academic research.

Based on specific needs and inquiries, different corpora were built for different purposes. Sinclair (1995) classifies corpora into the following types:

- Reference corpus: designed to provide comprehensive information about a language, e.g. the British National Corpus (BNC) (Aston and Burnard, 1998), or a corpus for interstellar communication (Atwell and Elliot 2001).
- Monitor corpus: this type of corpus has a fixed size, and it is updated and refreshed from time to time, e.g. the Bank of English/Birmingham Corpus (Renouf 1987)
- Parallel corpus: collections of texts, translated to different languages, e.g. Al-Qur’an¹, the holy book of Islam, is translated to multiple languages; the CRATER corpus (McEnery et al. 1997).
- Comparable corpus: similar text in more than one language. For example two newspapers of the same domain in English and French are considered

¹ <http://www.oneummah.net/quran/quran.html>

as comparable corpora. A well known comparable corpus is the International Corpus of English (ICE) (Greenbaum 1996).

Hunston (2002) adds the following corpora types:

- Specialised corpus: texts are collected to represent a particular type of text, e.g. the Michigan Corpus of Academic Spoken English (MICASE 2003)
- Learner corpus: collections of texts produced by learners of a language, e.g. the International Corpus of Learner English (ICLE) (Granger et al. 2002), or the Interactive Spoken Language Education (ISLE) (Atwell et al. 2003).
- Historical corpus: texts from different periods of time, e.g. the Helsinki corpus, which consists of 1.5 million words, sampled from texts in old, middle, and early English periods (Kytö et al. 2000).

2.2.1 Computational linguistics and language engineering

In parallel with building machine readable corpora, a new bridge of cooperation between language engineering researchers and linguists was established to develop automatic software tools to replace manual analysis. This cooperation creates a new branch of natural language processing entitled computational linguistics. Grishman (1986) described computational linguistics as “the study of computer systems for understanding and generating natural language.” The computer is used to read the corpus and build different tools such as: annotation tools, concordancing and text analysis tools, and searching tools.

Annotation tools are used to annotate a text or a corpus in different terms such as:

- Part-of-speech (POS) tagging: assigns each word to its grammatical tag using for instance “V” for verb, “N” for noun. The Constituent Likelihood Automatic Word-tagging System (CLAWS) is an example of POS tagging tools (Leech et al. 1983, Garside 1987), which is used to annotate the LOB and BNC corpus. The AMALGAM tagger (Atwell et al. 2000a) can POS-tag according to several different corpus schemes.

- Grammatical parsing: annotates each sentence in the corpus with its phrase structural grammar (Atwell 1996a, Atwell et al. 2000a). The resulting corpora are known as “treebanks”.
- Semantic tagging: assigns each word to a semantic tag such as “E2” attached to “like” which denotes the category “emotional states, actions, events and processes”. Automatic Content Analysis of Spoken Discourse (ACASD) is an example of a word sense tagging system (Wilson and Rayson 1993). Demetriou and Atwell (2001) is another example of a domain-independent semantic tagger for the study of meaning association in English text.
- Speech-act tagging: annotates a dialogue corpora in terms of speech act categories, such as “decl” for declarative, “q-wh” for wh-question, “greet” for greeting. For example, Speech Act Annotated Corpus (SPAAC) is a new semi-automatic tool introduced by Leech and Weisser (2003); Churcher et al. (1997) presents the semantic/pragmatic annotation of an air traffic control corpus.

Concordancing and text analysing tools such as: Wordsmith (Scott 2004) and aConCorde (Roberts 2004). The concordance provides lists of words in a text with its context.

Searching tools such as XARA (an XML Aware Retrieval Application) (Burnard and Dodd 2003), which is a general tool for searching large XML corpora, with a particular focus on the needs of corpus linguistics. Users can search for substrings, words, phrases, or the tags which delimit XML elements and their descriptive attributes.

As well as specific tools, language engineering researchers also build generic natural language toolkits and platforms. Language engineering is a “discipline or act of engineering software systems that perform tasks involving processing human language.” (Cunningham 1999).

GATE, the General Architecture for Text Engineering, is a platform that aids in construction, testing, and evaluation of language engineering systems

(Cunningham et al. 1996, 1999; Gaizauskas et al. 1996; Cunningham 2002). GATE is composed of three basic elements:

- A database for storing information about the text and a database schema based on an object oriented model;
- A graphical user interface for launching processing tools on data and viewing and evaluating the results;
- A collection of wrappers for algorithmic and data resources that interoperate with the database and interface.

The GATE module set could be assembled to extract, retrieve, or even translate information. In addition to the main aim of developing GATE to advance research in the area of natural languages, Bontcheva et al. (2002) argued that the GATE could be used as an effective tool for teaching language engineering and computational linguistics. The authors justify this argument by pointing to tools built in GATE which offer `comprehensive Unicode-compliant multilingual support that allows student to create components in other languages, in addition to other tools for corpus annotations and performance evaluation.

The natural language toolkit (NLTK) is a set of libraries written in Python, designed for teaching corpus/computational linguistics. NLTK allows students to focus on natural language tasks instead of complex programming problems. NLTK provides “a simple, extensible, uniform framework for assignments, projects, and class demonstrations”, (Loper and Bird 2002). NLTK is made up of the following independent modules:

- The token module: provides basic classes for processing individual elements of text, such as words, or sentences;
- The tree module: defines data structures for representing tree structures over text, such as syntax trees and morphological trees;
- The probability module: implements classes that encode words frequency distributions and probability distributions;
- Other modules define data structure and interfaces for performing specific NLP tasks.

The above serves to illustrate computational linguistics and language engineering research and development. For more detailed overviews, see (Atwell 1999, Jurafsky and Martin 2000).

2.3 Chatbot history

The idea of chatbot systems originated in the Massachusetts Institute of Technology where Weizenbaum (1966, 1967) implemented the ELIZA chatbot to emulate a psychotherapist. Weizenbaum (1966) described ELIZA as a program which “makes natural language conversation with a computer possible.” The idea was simple and based on keyword matching. The input is inspected for the presence of a keyword. If such a word found, the sentence is mapped according to a rule associated with the keyword, if not a connected free remark, or under certain conditions an earlier transformation, is retrieved. To keep the conversation going, ELIZA has some fixed phrases in case no match found such as “Very interesting. Please go on.”, or “Can you elaborate on that?”

The next major program was PARRY, implemented in 1970. Colby (1973) described PARRY as a simulation of paranoid thought processing, representing beliefs, fears, and anxieties. In contrast to ELIZA, instead of simulating a psychotherapist, PARRY modelled a paranoid patient during an interview with his therapist. Psychiatrists were given transcripts of dialogues generated with PARRY and others with actual patients; the psychiatrists could not distinguish between them. PARRY’s knowledge is stored in condition-action production rules, where a semantic pattern is associated with an output response. The input is portioned into patterns, where each one is compared for similarities with the semantic patterns stored in the database. “Colby regarded PARRY as a tool to study the nature of paranoia, and considered ELIZA as a potential clinical agent who could, within a time-sharing framework, autonomously handle several hundred patients an hour.” (Güzeldere and Franchi 1995)

Saygin, Cicekli, and Akman (2000) noticed that, “Both ELIZA and PARRY use certain tricks to be able to successfully perform in conversations. ELIZA directs the conversation away from herself by asking questions... ELIZA uses parts of the user’s input in the output questions and seems to be following the

conversations. In addition to these techniques, PARRY has little stories to tell and tend to insert these in the conversation.”

In 1972, Terry Winograd implemented the SHRDLU system to simulate “the actions of a robot interacting with a “blocks world” of different colored and shaped blocks which could be placed on a table or put in a box.” (Stephens 2004). Winograd (1972) described communication with SHRDLU as “talking to a simple robot, with a hand and an eye and the ability to manipulate toy blocks on a table. The robot responds by carrying out commands (in a simulated scene on a display screen attached to the computer), typing out answers to questions, and accepting information to use in reasoning later on”. SHRDLU uses a top-down, left-to-right parser which analyses a pattern, identifies its structure, and recognises its relevant features and grammar. To understand the meaning, the sentence was represented in terms of objects, properties, and relations in a procedural representation form.

2.4 Recent chatbots

“The need of conversational agents has become acute with the widespread use of personal machines with wish to communicate and the desire of their makers to provide natural language interfaces” (Wilks 1999).

MegaHAL (Hutchens 1996; Hutchens and Alder 1998) was the first chatbot to use machine learning techniques in generating replies from a training corpus. It uses its own language model, which consists of two Markov models. The first is able to predict a word that will follow a given string; the other can predict which word precedes a given sequence. This model is used to generate chatbot replies. MegaHAL obtains a list of keywords from user’s input by tokenising and then removing any high frequency words, such as ‘the’ or ‘and’, etc. Each keyword is used as a seed for creating a possible reply. Starting at the keyword, the ‘forward’ model generates a sequence to complete a sentence, whereas the ‘backward’ model works from the keyword back to the beginning of a sentence. Many candidate replies are generated, and so a formula is used that defines the *highest information*, which makes the final decision as to which reply is output the user. Essentially, the highest information formula was constructed so that

MegaHAL actually gave the most surprising replies in order to keep the conversation original. When MegaHAL was entered in the 1996 Loebner contest (See 2.11.2), the judges were so surprised by MegaHAL's replies that they thought it was the most human-like program, and awarded the Prize to MegaHAL. However, many replies are ungrammatical and/or nonsensical, and subsequently judges realise this was characteristic of a Markov model of sentence-structure, so were not fooled in subsequent contests. MegaHAL showed that a ML (machine learning) chatbot could be worthwhile, but a learning approach should generate well-formed, "realistic" replies.

Going in Socrates' direction who remarked "the cure of the soul has to be effected by certain charms, and these charms are fair words", and years after ELIZA and PARRY, Colby (1999b) presented a cognitive therapy program entitled Overcoming Depression. The system provides a therapeutic opportunity for depressives in the form of a personal computer program, where users chat with it to treat depression. The program is made up of a text mode and a dialogue mode. The text mode provides all concepts and explanations relevant to depression therapy. The dialogue mode represents a virtual person designed to participate in the conversation by behaving as if it understands user's input, and responding appropriately. The system aimed to encourage users to express their feeling, beliefs, and thoughts using supportive words, and offering challenging questions and assertions. The database is stored in condition-action production rules, where semantic patterns are associated to output responses. Each semantic pattern is composed of a token-expression (words, idioms, phrases, etc.), which is assigned to a semantic category, e.g. the input "Right" is assigned to the assertion category. Responses are classified to three syntactic types (questions, requests, and assertions), and five semantic forms (beliefs, causes, feelings, desires, and actions). These responses are partially ordered in terms of semantic 'intensity'. This order is changed while the dialogue proceeds to point directly to the user's personal intentional system. User's input is portioned into patterns, where each one is compared for similarities with the patterns stored in the database. From the set of patterns matched, the best response is selected (the top one in the list). If no match is found, the program either returns to the text mode or offers a question or assertion about the topic under discussion.

Converse represents "Catherine, a 26 year-old female editor for a magazine like Vanity Fair, who was born in the UK, but currently lives in New York." (Wilks

and Catizone 2001). In order to control the conversation flow, Converse was built based on the idea of “having something to say” (Batacharia et al. 1999). The system is composed of three basic modules: input, action, and the generator module, where each one includes sub modules. The first is the input module, which reads the user input, reprocesses it to correct grammatical and spelling mistakes, and analyses the syntax structure to determine whether each sentence is a command, question or a statement, and extract useful information from the input. The useful information is then passed to the action module, which is responsible for: generating answers to questions; handling exceptional cases such as rude language by selecting the response from a canned set of phrases; and keeping track of topics dealt within the conversation, monitoring user input and trying to match it with the content words of available topics. Finally the generator collects text generated by the action module and merges it into a coherent message, which is output to the user.

SpeakEasy (Ball 1999) is a dialogue controller, which interprets conversational scripts that control how a computer character interacts with a user. The script holds exemplars of utterances that the user might generate, with definitions of the assistant’s reaction. Conceptual representations of the expected inputs are generated from the input exemplars, which are then matched against user utterances in order to detect acceptable paraphrases. Ball listed a set of dialogue script requirements that need to be achieved by any conversational agent; the most important are:

- Avoiding inappropriate repetition in the assistant’s dialogue.
- Accessing the referenced conversational history in order to respond appropriately.
- Having something to say when no match is found, in a way that controls the conversational flow.

SpeakEasy’s interface design allows independent flow of control to be created when the user changes the context. In the mean time, the previous contexts are still available to respond when user return back to them. When the same match occur more than one, different answers are displayed each time to avoid repetition. In the case of no match, SpeakEasy replies by “Sorry, I didn’t understand...” The output is presented using a general text-to-speech synthesizer and graphical character with a limited collection of pre-authored animations.

Alabiso and Kronfeld (1999) assume that “the point of each and every interaction between man and machine is to make the application do something”. They used this assumption in developing LEAP, which is a software tool that enables developers to add a natural language interface (including speech) to their products. LEAP converts spoken utterances into function calls that are dispatched to an application for execution as user requests. The idea is based on thinking of the world in terms of entities (objects), their properties (traits) and the relationships they participate in. User inputs are treated as tasks, and each task has three parts: specifying entities that are relevant to the task; specifying the properties that are needed to make functions’ call; and specifying a set of templates, each corresponding to a class of sentences that can be used to initiate a task. Each task is composed of two discourse segments: conversational template, which holds the knowledge about the task, and the conversational instance which holds the context of the current discourse segment. The discourse is composed of three interaction components. The first is the linguistic structure of the utterance. The second is the discourse intention, which is encapsulated by the tasks required. The third is the attentional state, which is the collection of entities, properties, and relationships that are relevant for the task. User’s input is analysed and matched with the appropriate template, and then each entity and property required for the task are instantiated and the task is executed. The method is similar to the idea used in building information extraction systems, but instead of filling the structure template using a corpus, the user input is used to match the template type. In order to use LEAP, the author must map the functionality of his application into an object-oriented formalism, which LEAP can read from author’s file. LEAP was used with the Bunny Express, which is an application that allows children to create lines, rectangles, ellipses, and text boxes in different colours on screen.

Aside from the fun of chatting with a machine, intelligent agents, and chatbots in particular are playing an important role in technology. Chatbots could speak different languages, serve in different domains, and have multiple personalities. Table 2.1 summarises some other current chatbots.

Chatbot	Application	Communication mode
Sanelma ²	A fictional person to talk with in a museum, which provides background information concerning a certain piece of art. Sanelma is a 26 year old woman from Helsinki of the 30's.	Textual mode
PC Therapist ³	Simulate a Rogerian therapist, inspired from ELIZA. Different personalities have been developed such as: PC professor, discusses men versus women; PC Politician, discusses Liberals versus Conservatives.	Spoken mode
Speak2Me ⁴	A female chatbot that is used to teach English language through chatting.	Input: textual mode Output: spoken and textual mode.
Marloes ⁴	A female Dutch financial advisor.	Spoken mode
MIA ⁵	A German advisor on opening a bank account.	Textual mode
Cybelle ⁶	A female avatar with body and uses gestures while talking. She directs you to discover the agent land, a new land where you can find more information about agents, what they are, how they work, how they could be useful for you.	Textual mode

Table 2. 1 Some recent chatbots

² <http://www.mlab.uiah.fi/mummi/sanelma/>

³ <http://www.loebner.net/Prizef/weintraub-bio.html>

⁴ www.speak2me.net

⁵ www.aitools.org

⁶ AgentLand.com

2.5 Chatbot communication language

Language is defined in the Collins English Dictionary as “a system for the expression of thoughts, feelings, etc., by the use of spoken sounds or conventional symbols” (Makins 1991). Language is a way of communication as addressed by Grishman (1986) “the primary vehicle by which people communicate and record information”. Human communication modalities can be classified into speech mode e.g., face to face communication, or telephone conversations; and text mode e.g., sending letters through mail, or by emails, etc. There are significant differences between these modalities, for example simple gaze behaviour and beat gestures play a primary role in face to face communication (Cassell et al. 1999); while stress, intonation and tone of the voice affect understanding over the phone.

In the same manner of using language for human communication, people need to use their language to communicate with computer machines. For example, speech communication is used to interact with a dialogue system, which is defined as “a computer system that interacts with users utilising connected natural language dialogue, where the use of natural language need not consist of predefined commands.” (Flycht-Eriksson 2001). A wide variety of system applications have been proposed for dialogue systems, ranging from “question-answering systems that answer one question at a time, to conversational systems that engage in an extended conversation with the user.” (McTear 2002). The best way to facilitate Human Computer Interaction (HCI) is by allowing users “to express their interest, wishes, or queries directly and naturally, by speaking, typing, and pointing.” (Zadrozny et al. 2000). This was the real trend beyond developing man-machine interfaces, which are defined as “the mediator between users and machines. It is a system that takes care of the entire communication process, is responsible for the provision of the machine “knowledge”, functionality and available information, in a way that is compatible with the end-user's communication channels, and translates the user's actions (user input) into a form (instructions/commands) understandable by a machine.” (Stephanidis and Sfyraakis 1995).

However, some differences are found when using the natural language to communicate with people, and the one used to communicate with computers.

Takahashi (2003) examined interaction via computer-mediated communication (CMC) within his paper *"Do we talk or write differently over the net?"* He reviewed previous work that described the characteristics of the language of CMC and the relationship between CMC and spoken and written language. These characteristics describe CMC as dynamic communication, classified into two types: synchronous such Internet Relay Chat (IRC), which maintains strong coherence in turn taking, like face-to-face conversation; and asynchronous such as emails. In order to classify the chatbot language, (Herring 1996) argues that computer mediated communication (CMC) "is typed, and hence like writing, but exchanges are often rapid and informal, and hence more like spoken conversation".

On the same context, Grondelares et al. (2003) described the language of Internet Relay Chat (IRC) as an example of "spoken language in written form", as it shares with spoken language the informality characteristic. Four features of informality were discussed. The first is the higher frequency of second person pronouns and vocatives. The second is a lot of abbreviation and ellipsis typed by the users. A third source is speaker related: age, gender, and the topic of the chat. The fourth factor is register related: that chatters might choose to sound vernacular, or to maintain a more formal standard.

Cheepen and Monaghan (1999) addressed that the degree of formality is an important consideration for designers of automated dialogue. Dialogue types are classified into two categories: transactional and interactional pole of discourse meaning. They decided that the more suitable for automation is the transactional one, as it is more obvious and has an observable goal, while the interactional reflects the effect of the conversation on the relationship of the speakers. Cheepen and Monaghan adopted the view that the automated systems are "not opportunities for the pleasurable give-and-take of essentially human interactional talk". Instead of trying to mimic the human dialogue, automated systems should take full account of the transactional speech event. However, the case is different with chatbots, sometimes people use it for fun, to chat when feeling lonely, other times the chatbot is used for a special purpose on a specific domain.

As most chatbots use text communication, we cannot say that the text is exactly like the written form, but it is a hybrid between spoken and text modalities. Since

dialogue corpora represent real human conversation, the dialogue corpora were selected to seed some of the generated versions in this research; regardless of the linguistic features involved within, which cannot be straight forwardly modelled within the text mode.

2.6 Chatbots and pattern-matching techniques

An important component in developing chatbots is the way the chatbot analyses the input, and finds the best match from the stored knowledge. Pattern-matching techniques can affect the users' belief that they are chatting with a real human. Some of pattern-matching techniques used within the previous chatbots are:

- Matching keyword and applying transformation rules such as in ELIZA and Sanelma.
- Storing the dialogue in a condition-action rule based system as used by the Overcome Depression system.
- Using Markov model as in MegaHAL.

Lee et al. (2001) claimed that “most conversational agents use simple pattern-matching techniques for their answer without considering user goals and thus give users an unsatisfactory response”. The authors proposed using Bayesian network to model users' intentions or goals to have satisfactory responses. The knowledge base scripts consist of a list of queries and responses represented as conditions-action rules. Each query is classified according to a particular topic. A user model is constructed as a graph of children-parent relationships, where the goal variables are the parent and children denoted by evidence variables. For example, “Location” is a parent node (goal), connected to “where”, “address”, or “place” children nodes (evidences). To find an answer for a query, the process is applied as follows:

1. Users' input is corrected in terms of spelling and a list of keywords and synonyms are generated using a reprocessing module.
2. Users' goals are identified through a goal inference module, which aims to identify the topic (goal of the query) by finding either the topic directly as using the word “location” or finding the evidence variables or its synonyms in the input. When goals are not found, users are asked to give more information.

3. During the pattern-matching process, the conditions of all topics are compared to the goal information, the query and the keyword list. As a result the topics are scored, in terms of goals and keywords matched, and the action of the highest score topic is returned as an answer.

This approach was applied in a specific domain and results showed better response in terms of consistency with users' goals. However, different pattern-matching techniques are also used in a wide range of applications and fields. The following sub-sections review the pattern-matching techniques, and corpus based approaches used in information retrieval, information extraction and question answering systems.

2.6.1 Information retrieval

Information retrieval (IR) systems retrieve related documents according to specific user query as described in section 1.2.2. Most IR systems use a vector-space model. In this model, each of the documents and user's request are treated as a vector of n-dimensional space, where n represents the number of meaningful tokens in the user's query and in the set of documents to be searched. The IR system analyses the user's query to its lexical terms, indexes the user's request and documents according to meaningful terms, and finally returns the documents that have the highest frequency of the important words appearing in the user's query. (Russell and Norvig 1995). Retrieval process is based on indexing, which indicates what documents are about. The aim of indexing is "to increase *precision*, the proportion of retrieved documents that are relevant, and to increase *recall*, the proportion of relevant documents that are retrieved." (Lewis and Jones 1996).

Information retrieval systems such as Google⁷ and other search engines depend on keyword matching. On the other hand, AskJeeves⁸ is an example of IR systems which uses syntax and semantics to analyse the input and give an output. It expects user to ask proper questions, whereas Google expects keywords.

⁷ <http://www.google.co.uk>

⁸ <http://ask.co.uk/home>

Vrajitoru (2003) merges pattern-matching technique used in ELIZA with indexing and query matching methods inspired from information retrieval in building a chatbot. The aim of the model is to “simulate particular personalities, fictional or real mostly taken from literature, film, or TV shows” (Vrajitoru 2003). Typical IR systems index a collection of documents, such as web pages, and return documents that contain keywords from the users’ input query. Vrajitoru used a similar model, except the “documents” were in fact individual sentences of a given personality. When a user of the chatbot converses, the chatbot acts like a search engine: locates the keywords from the input and then finds the most relevant match from its indexed documents. In this instance, the match will be a sentence, and therefore a conversation can take place. The chatbot is adapted even further to include a memory, so that it doesn’t return the same responses. Additionally, it utilises evolutionary algorithms to generate new sentences based on those returned from a query.

Unlike information retrieval systems, the chatbot versions retrained in this thesis, were able to retrieve responses using simple approaches. The main idea is based on tokenising the sentences, extracting the most significant words (which are the least frequent ones) and matching users’ query with these significant words to retrieve answers as will be discussed in chapter 4.

2.6.2 Information extraction

The goal of information extraction (IE) systems is to “pull specific information from large volumes of text and store this information in structured databases, where users could quickly query, aggregate, and otherwise analyze it.” (Taylor 2004).

In general, an IE system is composed of six phases as described by Cowie and Lehnert (1996). The first is the filtering process, which determines the relevant parts of a text. After that part-of-speech (POS) tagging is applied to denote words within their POS, e.g., verb, noun, adjective, etc. The third stage is the process of semantic tagging which recognises major phrasal unit and marks them with semantic information. In the fourth stage a lexical parser is used to show the relationship between the phrasal elements. Coreference resolution is applied in the fifth stage to recognise and unify referring expressions. Finally the extracted information is formatted to suit the output template. In order to estimate the

precision and recall as in IR systems, Cowie and Wilks (2000) reported that the evaluation of IE systems is based on comparing the automatic generated templates with ones produced by humans for the same texts. Precision is the percentage of the of slots filled correctly to the total number of slots the system filled, and recall is the percentage of slots the system found correctly to the slots in the answer key.

IE systems are used in different domains such as finding the names of all people and places in a document, retrieving information about people's ages, finding all reports of murders from a newspaper, or tracking medical patient records. Different learning algorithms are applied to look for patterns that include the required data within corpora. Some systems operate at the level of a sequence of words; others use syntactic analysis, such as identifying verb groups and noun groups, or patterns are stated in term of subject-verb-object relations. A selection of IE tools is outlined below:

- The VRA Reader (Virtual Research Associates) (King and Lowe 2003) is a software tool that deal with data from Reuters Business Briefing (RBB) newswire. The Reader “extracts the first sentence, or lead, from RBB articles and attempts to deliver a compact quantitative summary of all the events that are described in the lead” (King and Lowe 2003). The extraction process is composed of three stages: the first is the tokenisation and lexical processing which involves using word segmentation, part-of-speech tagging, and word-sense tagging. The second stage is syntactic parsing which analyses the grammatical structure of a sentence, and makes an extensive use of the Word-Net lexical semantic net work to solve the ambiguity problem. The last stage is domain analysis which involves resolving coreference ambiguity using domain information. coreference ambiguity problem arise when referring to the same entity in several different ways especially in dealing with names, e.g. deciding whether the “John Doe” is the same person as the “J.Doe”.
- The Textractor (Zavrel et al. 2000) is a Dutch job advertisements IE system. Developers adopted a text classification approach to extract slot fillers out of the text. The basic idea is to label the symbolic representation of the entities that will be extracted as categories, treat the

instantiation of these entities and the predictive co-occurring string as features, and segment the text into labelled sections. After that the memory based learning method is used to learn all of the classification tasks from labelled examples that are stored in memory as binary vectors. To learn a new category the nearest neighbour set is retrieved from memory and the most frequent category is obtained. Then, different feature-selection methods were used to search for good features. This includes features of different types: unigram (single), phrase (sequential unigrams), and combinations: conjunction of phrases and section labels.

- ECRAN (Poibeau 2000) attempts to offer generic and portable IE systems for French, English and Italian. It searches movies and financial databases. ECRAN is composed of four parts: shallow parsing module to produce a tagged text with part of speech and some morphological information; named entity recognition module to recognise entities such as dates, locations, and person names by applying a grammar of regular expressions; a module to extract other information from the structure of the text; and finally a module to link all these information together to produce a result template.
- Brin's DIPRE (Dual Iterative Pattern Relation Expansion) system (Brin 1998) uses a bootstrapping method to find patterns in un-annotated texts from the World Wide Web. The idea based on extracting a relation of books (author, title) pairs from the web. The methodology is initiated by a small seed set of relation pairs. All occurrence of relation (author, title) with its surrounding context is found within the web. Each occurrence is composed of 5 fields: the order of author, and title occurring in a text, the URL of the document where the book was found, the strings prefixing the author, the text between the author and title (middle), and the strings suffixing the title. Then the occurrence list is used to produce a patterns list, where each pattern matches all occurrences that hold the same order and middle fields, and matches the longest path of prefix, suffix, and URL. The World Wide Web can be searched using these patterns to obtain new books.

- Other researchers illustrated how to use topic relevance for bootstrapped search for patterns (Grishman and Yangarber 2000). The process started by seed patterns that characterise the topic of interest, then search to find the documents containing these patterns in relevant documents, ranked by their frequency in the relevant set compared to their frequency in the remainder of the corpus. The top-ranked patterns are added to the set of seed patterns and reused for extraction.

2.6.3 Question answering systems

Question answering (QA) systems are developed to accept user's question in natural language, and retrieve answers from either the Internet, or question-answer databases. "The goal of a question answering system is to retrieve 'answers' to questions rather than full documents or even best-matching passages as most information retrieval systems currently do." (Dumais et al. 2002).

START, is a question answer system with a web interface, developed by Boris Katz at MIT's AI lab in 1993 (Katz 1997). START's knowledge base is represented as subject-relation-object tuples. The system breaks up an English sentence into smaller units called kernel sentences, where each one contains only one verb. These sentences are analysed separately and keywords are extracted. The keywords are used to fit into one of question templates; after that one of the answer templates is returned. When no answer is found, the system replies that it does not know the answer (Costello and Smeaton 2004).

Kwok et al. (2001) split any automated QA system into three main components:

1. an information retrieval engine to return relevant documents which can answer the question.
2. a query formulation mechanism to map the user question into a query for the IR engine.
3. an answer extraction method to analyse the documents and extracts answers from them.

Based on these components, MULDER system was built as "a web question answering system" (Kwok et al. 2001), which accepts user's question, submits it

to Google search engine, and extracts the answers from the engine's output. The process is not that simple, the original user's question is parsed to obtain the syntactic tree structure. Then the tree structure is passed to a question classifier to determine the type of answer to expect. After that, the parsed question is translated to a series of search engine queries using a query formulation module. These queries are submitted to Google search engine to fetch the relevant documents. The answer extraction module takes the turn now to extract relevant snippets called "summaries" from the web pages then rank them in terms of having important keywords which are close to each other, and generate a list of possible candidate answers. Finally the answer selection module ranks the candidates according to how close they are to keywords; clusters the similar answers together; casts a final ballot for all clusters; and generates an ordered list of answers with the highest ranked one in the top.

AnswerBus⁹ is another question answering system which works in a similar way to MULDER. However, instead of restricting the user's question to be in English as MULDER, AnswerBus accepts German, French, Spanish, Italian and Portuguese questions; then a translation tool is used to generate the corresponding English ones. AnswerBus submits the question to more than one search engine including Google, Yahoo, WiseNut, and Altavista based on the question to retrieve the relevant documents. The Scoring mechanism is based on matching words and other factors such as using a specific dictionary for QA, named entity extraction, co reference resolution, and redundancy deletion.

Instead of retrieving the answers from the web as MULDER and AnswerBus, FAQ-Finder (Robin et al. 1997) uses files of frequently asked questions extracted from USENET News as its knowledge base to obtain answers. To improve the matching process, FAQ-finder applies syntactic parsing of the question, and performs semantic matching using the semantic knowledge base "WordNet". In contrast, Auto-FAQ system (Whitehead 1995) uses "shallow" natural language understanding techniques, which means "the matching is based on keyword comparison; the system performs no syntactic parsing of the question, it does not extract semantic concepts." (Sneiders 1999).

⁹ <http://www.answerbus.com/about/index.shtml>

Instead of focusing in matching the user's question with FAQ questions to extract answers, Kosseim et al. (2003) suggest another approach called "answer formulation". In this case, answer templates will be built manually for each question, and these templates will be used for matching instead of question matching to extract the missing information in a way similar to information extraction. For example, given the question "Who is the prime minister of England?", one expected answer formula is "The prime minister of England is <PERSON-NAME>". Then the World Wide Web is searched for this exact answer sentence, and the noun phrase will be extracted. Evaluation for this technique showed that the generation of good linguistic quality answer formulas allows the system to give a full complete sentence as an answer to the user.

In this respect, the natural language interface could be built as a tool for information retrieval, information extraction, and question answering (QA) systems. Machine conversations and almost all QA systems have a natural language interface to facilitate the communication with users.

BEE-SMART (Supekra et al. 2004) is a natural language interface system that allows users to retrieve knowledge and executes services over the web. BEE-SMART was developed by 15 graduate students under Dr. Yugyung Lee supervision at University of Missouri. The interface provides the following functionalities:

- Virtual chat, e.g., "how are you?" and "I am fine".
- Information retrieval, e.g. "Who is Pete Sampras?"
- Task execution, e.g. "Find the stock price for the stock ticker IBM"

OSCON (Mc Kevitt 2000) "is a natural language interface which answers English queries about computer operating systems." The main problem in developing a natural language interface as described by Mc Kevitt is that there are many ways to ask an English query. To overcome this problem, OSCON was designed to deal with three types of queries: request for information; request for explanation; and request for direction. OSCON is composed of the following modules: grammar parser, semantic parser, natural language generator and knowledge base.

2.6.4 Summary of pattern-matching techniques

This review shows that IR systems use surface lexical pattern-matching to match whole documents. Whereas the existing IE and QA systems use more sophisticated NLP to analyse input and retrieve answers, which restricts these systems to NLP modules. In this research, the aim is to investigate how well simple lexical pattern-matching can be extended to general, powerful chat generation and question-answering.

In this research, a tool called FAQchat (discussed in chapter 5) is built to give answers for users' questions related to FAQ at the School of Computing at the University of Leeds. FAQchat looks like question answering and information retrieval systems in terms of accepting natural language question, and returning an answer or a set of links to related answers. However, FAQchat differs than the previous systems in terms of its simplicity. FAQchat is similar to FAQ-Finder in terms of using FAQ files instead of retrieving from the World Wide Web as MULDER and AnswerBus do. On the other hand, FAQchat does not use any syntactic, semantic or even morphological analysis during the matching process. Therefore, FAQchat belongs to systems using shallow natural language understanding. FAQchat evaluation proved that FAQchat is practical and effective despite the absence of a real natural language understanding.

2.7 Machine learning techniques

Lozano-Pérez and Kaelbling (2003) defined the word *learning* in terms of having the ability to memorise something, learn facts through observations, improve cognitive skills through practicing, and organise new knowledge into an effective representation. The authors added that “the most common kinds of learning is the acquisition of information with the goal of making predictions about the future.” (Lozano-Pérez and Kaelbling 2003). The authors classified machine learning techniques into:

- Supervised learning, which includes finding a rule to predict the output of a new input using a set of example input/output pairs.
- Clustering, which includes grouping a set of given examples (with no associated labelling) into natural clusters, this is also called unsupervised learning.

- Reinforcement learning, which includes observing the real world and learning to take actions in such a way to obtain a lot of reward; some other machine learning researchers would say this is just a special case of supervised learning, adapting an initial model.

Murquez (2000) classified the machine learning in a different way as follows:

- Stochastic machine learning approaches, which describe the real world data as a probabilistic network that presenting the probabilistic dependencies between random variables, e.g., the Hidden Markov Model (HMM).
- Symbolic machine learning approaches, which include learning techniques that do not use probabilities. Examples of these approaches are: *decision trees*, which are a supervised way that transform the data in a compact hierarchal structure; *decision lists*, which are ordered lists of conductive rules; and *genetic algorithms*, which are used to infer word categories and syntactic structure from unannotated corpora.

Machine learning is widely used in corpus-based computational linguistics; see for example (Atwell 1996b, 2003, 2004). Despite the fact that most chatbots do not have the ability to learn, the advocated learning approach used in this thesis to build input/output rules, is similar to what is known as *instance-based learning (memory-based learning, case-based learning)* approach. In contrast to decision trees, and clustering which try to build a general representation form which is used during classification (predicting an answer), the instance-based learning is a form of supervised inductive learning, which keeps all the training data in its memory, and uses it to predict (classify) an answer for a new input by using for example the *k nearest neighbour* algorithm. In the instance-based learning approach, the data is represented a set of instances having specific attributes. For example the Weka (Witten and Frank 1999) tool visualises the data as a table where each row represents an instance and the columns denotes the attributes. These attributes are used in the matching process to classify a new instance.

In our approach, the corpus is considered as the original data. To extract input/output rules from the corpus, the following were applied:

- Each utterance pair (input/output pairs) is considered as an instance.
- The first word, and the most significant (least frequent) words extracted from each utterance are considered as attributes to classify an answer for a new input.
- Different outputs which have the same input are grouped in one instance with a random list of outputs.

This method will be described in detail in chapter 4. This technique provides a way of giving a nearest answer based on the most significant word matching.

2.8 ALICE chatbot system

ALICE (Artificial Linguistic Internet Computer Entity)¹⁰ (Wallace 2003) is a chatbot system that engages a human in dialogue, using AIML (Artificial Intelligent Mark-up Language), a version of XML, to represent the input/output rules of the underlying dialogue. The basic units of AIML objects are *categories*. Each category has a *pattern* which matches an input and associated *template* which implies the ALICE response.

The simplicity of AIML makes it easy for non-programmers, especially those who already know HTML, to get started writing chatbots based on ALICE; and it simplifies and streamlines the maintenance and extension of chatbots. One ambitious goal for AIML is that, if a number of people create their own chatbots, each with a unique area of expertise, the ALICE engine could literally merge-sort them together into a Superbot, automatically omitting duplicate categories. Both the source code and the ALICE content are freely available from the ALICE foundation website, in order to encourage others to “open source” their chatbots as well, to contribute to the Superbot.

2.8.1 Overview of AIML

AIML “describes a class of data objects called AIML objects and partially describes the behaviour of computer programs that process them” (Wallace 2003). Each AIML file contains the data objects, and starts with an `<aiml>` tag,

¹⁰ <http://www.alicebot.org/>

which represents the AIML version. AIML objects are made up of units called *topics* and categories. There are two optional tags called `<topic>` and `<that>`. The `<topic>` groups a set of categories together, and `<that>` tag holds the chatbot last response. The full AIML format is shown in figure 2.1.

The AIML pattern language is simple, consisting only of words, spaces, and the wildcard symbols `_` and `*`. The words may consist of only uppercase letters and numerals. A single space is used to separate between words, and the wildcard characters function like words. The first versions of AIML allowed only one wild card character per pattern. The AIML 1.01 standard permits multiple wildcards in each pattern.

The template is the AIML response. In its simplest form, the template consists of only plain, unmarked text. The template may also contain AIML tags which can save data, activate other programs, give conditional responses, and recursively call the pattern matcher to insert the responses from other categories. For example, the `<javascript>` tag allows arbitrary scripting inside the templates.

```

<aiml version="1.0">

<topic name="the topic">

    <category>

        <pattern>USER INPUT</pattern>

        <that>THAT</that>

        <template>Chatbot answer</template>

    </category>

    ..

</topic>

</aiml>

```

Figure 2. 1 The AIML format

2.8.2 Types of categories

There are three types of categories: atomic, default and recursive categories.

1. *Atomic categories* are those with patterns that do not have wildcards.

```
<category><pattern>10 DOLLARS</pattern>
<template>Wow, that is cheap!</template></category>
```

In this case if the user input is “10 dollars”, and the response will be “Wow, that is cheap!”

2. *Default categories* are those with patterns has wildcards. These patterns result from a reduction process while the chatbot is searching for an appropriate match. The wildcards match any input, but they are differing in their alphabetical order.

```
<category><pattern>5 *</pattern>
<template>It is five</template></category>
```

If the user input is “5 dollars”, and this category is the only match, then the response is “It is five.”

3. *Recursive categories* are properties of templates not patterns. The template calls the pattern matcher recursively using `<srai>` or `<sr>` operator.

2.8.3 The recursive categories

Many applications could be achieved using the recursive tags which affect its meaning of the acronym. The “A.I.” denotes artificial intelligence, where “S.R.” may denote “stimulus-response”, “syntactic rewrite”, “symbolic reduction”, “simple recursion”, or “synonym resolution”. Basically there are two main functionalities of the `<srai>` and `<sr>` tags in AIML:

1. Mapping user's input into another form, in order to:
 - Simplify the language. Since there are many ways to formulate a sentence, in which some of them are complex, the recursive process maps such inputs to a simpler standard pattern. For example, "Do you know who Socrates is ?" is mapped into "Who is Socrates"
 - Deal with synonyms. All inputs that have the same meaning, or the same response are mapped to the same pattern e.g., "hiya", "hi there!", or "hi" are mapped to "hello".
 - Correct spelling or grammar, e.g., "your a .." is a common mistake which is mapped to "you are".
 - Detect keywords. In fact some patterns are concerned with particular keyword regardless its prefixes or suffixes, so wherever this keyword is detected in the sentence, the same answer will return. In this case different categories are generated to deal with the keyword position, e.g., if the keywords is "Mother", then the following input patterns are generated, to handle the middle location "_ MOTHER *", to handle the first "MOTHER *", to handle the last "_Mother", and if it is only one word "MOTHER".
2. Partitioning: break down users' input into one or more parts, and then combine their responses back together, e.g., any sentence starts with "Yes" or "No" will be treated as a two separate sentences, the first is "Yes", and the second is the suffix of the sentence.

Figure 2.2 illustrates different AIML samples obtained from the standard AIML generated in ALICE, and others from the versions developed in this thesis to represent some of the previous usage of the recursive categories.

Sample (a) shows the use of <srail> to simplify the language, obtained from the Afrikaans version developed in this thesis.

```
<category>
  <pattern>WAT IS JOU NAAM</pattern>
  <template>My naam is Karike. Wat is jou naam?</template>
</category>
<category>
  <pattern>HET JY N NAAM</pattern>
  <template><srail>WAT IS JOU NAAM?</srail></template>
</category>
<category>
  <pattern>HOE HEET JY</pattern>
  <template><srail>WAT IS JOU NAAM?</srail></template>
</category>
```

Sample (b) illustrates the partitioning task, obtained from the standard AIML files.

```
<category>
  <pattern>YES *</pattern>
  <template><srail>YES</srail><sr/></template>
</category>
```

Whatever input matched this pattern, the portion bound to the wildcard * may be inserted into the reply with the mark-up <sr/>, which is equivalent to <srail></star></srail>

Figure 2. 2 Samples of AIML categories, illustrate the usage of the recursive categories.

2.8.4 Preparation for pattern-matching in ALICE

Before starting the pattern-matching procedures, each input to the AIML interpreter must pass through two processes:

1. The normalisation process.
2. Producing input path from each sentence.

The normalisation process

The normalisation process involves three steps: substitution, sentence splitting, and pattern fitting normalisation as shown in table 2.2.

1. Substitution normalisation is a heuristic applied to an input that attempts to retain information in the input that would otherwise be lost during the sentence splitting or pattern fitting normalisation. It can distinguish the dot notation if it is used as an abbreviation, end of sentences or just a prefix of extension name and replace it by its appropriate meaning.
2. Sentence splitting normalisation is done before the matching, where each input is partitioned into sentences using simple rules like breaking sentences at periods, question marks, and exclamation marks.
3. Pattern fitting normalisation involves two tasks:
 - Removing punctuation from input to make it compatible with speech conversation.
 - Converting the input letters to upper case, to facilitate the matching with the AIML patterns.

Depending on the input, substitution and splitting may or may not apply, whereas pattern fitting always used.

Input	Substitution normalised form	sentence-splitting normalised form	pattern-fitting normalised form
"What time is it?"	"What time is it?"	"What time is it"	"WHAT TIME IS IT"
"Quickly, go to http://alicebot.org!"	"Quickly, go to http://alicebot dot org!"	"Quickly, go to http://alicebot dot org"	"QUICKLY GO TO HTTP ALICEBOT DOT ORG"
":-) That's funny."	"That is funny."	"That is funny"	"THAT IS FUNNY"
"I don't know. Do you, or will you, have a robots.txt file?"	"I do not know. Do you, or will you, have a robots dot txt file?"	"I do not know" "Do you, or will you, have a robots dot txt file"	"I DO NOT KNOW" "DO YOU OR WILL YOU HAVE A ROBOTS DOT TXT FILE"

Table 2. 2 The normalisation process in ALICE pattern-matching processing

Producing input path for each sentence

Table 2.3 views the process of producing an input path of a sentence. The input has the following form: Input <that> Tvalue<topic> Pvalue. The input path is composed of three values:

1. The normalised input.
2. <that> tag which holds the previous chatbot answer, normalised in the same way as input.

Tvalue = previous bot answer if exists

Else

Tvalue = *

3. <Topic> tag which allows the interpreter to prefer responses dealing with that topic.

Pvalue: = Topic name if exists

Else

Pvalue = *

Normalised input	Previous chatbot output (normalised)	Value of topic predicate	Input path
"YES"	"DO YOU LIKE CHEESE"	""	"YES <that> DO YOU LIKE CHEESE <topic> *"
"MY NAME IS NOEL"	"I GUESS SO"	"MUSHROOMS"	"MY NAME IS NOEL <that> I GUESS SO <topic> MUSHROOMS"

Table 2. 3 Producing the input path by the AIML interpreter.

2.8.5 The ALICE pattern-matching algorithm

The AIML interpreter tries to match word by word to obtain the largest pattern match, which is the best one. This behaviour can be described in terms of the class *Graphmaster* which consists of a tree of nodes called *Nodemappers* (that map branches from each node) and branches represent the first word of a reply or wildcards. The root of the Graphmaster is a Nodemapper with “about 2000 branches, one for each of the first words of all the patterns (50,000 in the case of the current open-source A.L.I.C.E. brain). The number of leaf nodes in the graph is equal to the number of categories, and each leaf node contains the <template> tag.” (Wallace 2003).

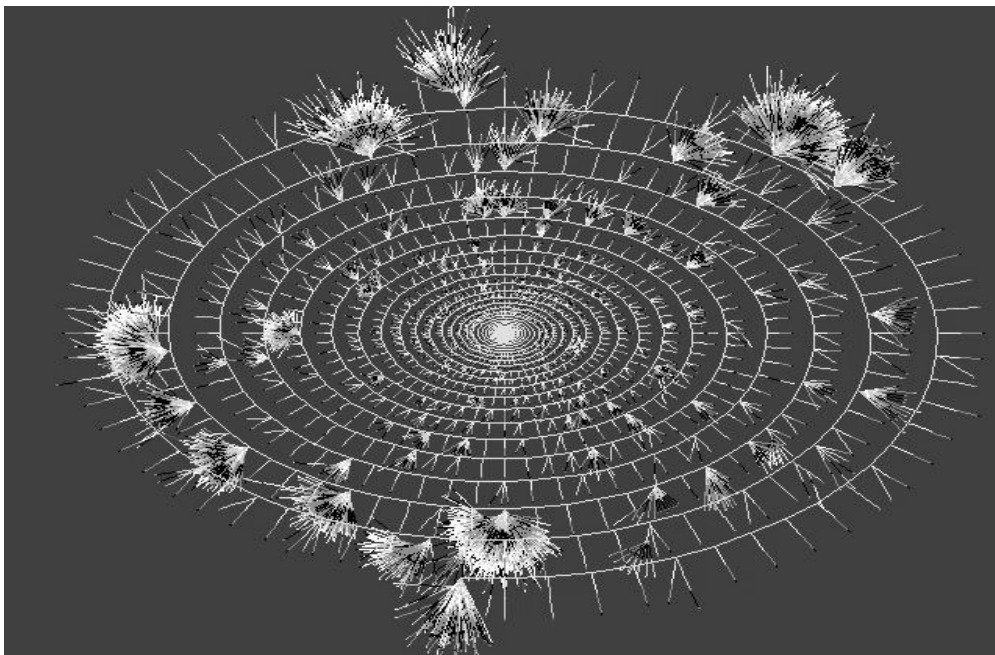


Figure 2. 3 Log-plot visualization of 24,000-state Graphmaster

Figure 2.3 is a visualisation of a 24,000 state graphmaster (from Wallace 2003). The root of the Graphmaster has about 2000 daughters, hard to represent graphically; so instead, this is mapped into a binary tree, each daughter is a binary split. This produces a binary tree with about 2000 levels of right-branching. This tree is then “warped” into a spiral, to fit in a standard PC screen.

The pattern-matching process could be described in terms of the file system, while all files and folders are organised hierarchically, in a tree. The file system is composed of a root, such as “c:\”, which has branches represent files and folders, and each folder in turns has its own files and sub-folders. Every file has a “path name” that determines its name and position. On the same manner the ALICE interpreter works. For example, the question “What is your name?” could be written as “g:/WHAT/IS/YOUR/NAME”, where g: is the Graphmaster root.

Assume the users’ input starts with a word “X”, the pattern-matching algorithm applied using depth-first search and back tracking techniques as follows:

1. “If the folder has a subfolder start with underscore then turn to ,“_/” , scan through it to match all words suffixed X, if no match then:
2. Go back to folder, try to find a subfolder start with word X, if so turn to “X/”, scan for matching the tail of X. Patterns are matched. If no match then:
3. Go back to the folder, try to find a subfolder start with star notation, if so, turn to “*/”, try all remaining suffixes of input following “X” to see if one match. If no match was found, change directory back to the parent of this folder, and put “X” back on the head of the input.” (Wallace 2003)

When a match is found, the process stops, and the template that belongs to that category is processed by interpreter to construct the output. If no match found, a random answer will be generated such as “Sorry, I have no idea”. Figure 2.4 illustrates the matching process.

Note that at every node, the "_" has first priority, an atomic word match has second priority, and a "*" has lowest priority. The patterns need not be ordered alphabetically, only partially ordered so that "_" comes before any word and "*" after any word. An example of ALICE code and the tracing of the matching process are shown in figure 2.4 and figure 2.5 respectively.

```

<category>
  <pattern>_ WHAT IS 2 AND 2</pattern>
  <template><sr/> <srai>WHAT IS 2 AND 2</srai></template>
</category>

<category>
  <pattern>WHAT IS 2 *</pattern>
  <template><random>
    <li>Two.</li>
    <li>Four.</li>
    <li>Six.</li>
  </random></template>
</category>

<category>
  <pattern>HIYA</pattern>
  <template><srai>HELLO</srai></template>
</category>

<category>
  <pattern>HELLO</pattern>
  <template><random>
    <li>Well hello there!</li>
    <li>Hi there!</li>
    <li>Hi there. I was just wanting to talk.</li>
  </random></template>
</category>

```

Figure 2. 4 An example of an AIML code

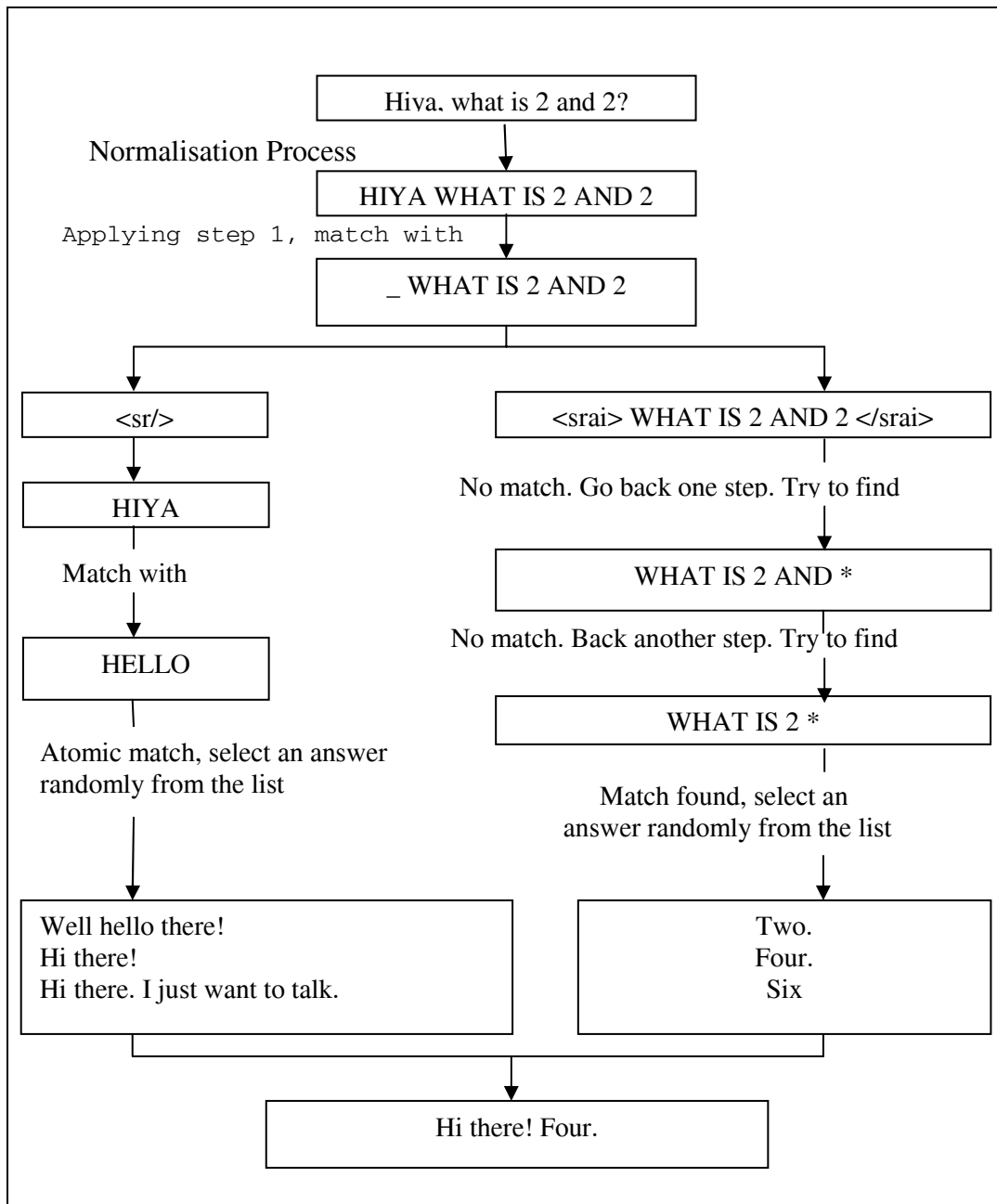


Figure 2. 5 Tracing of ALICE matching process

2.9 Elizabeth system

Elizabeth (Millican 2002) is an adaptation of the ELIZA program, in which the various selection, substitution, and phrase storage mechanisms have been enhanced and generalised to increase both flexibility and (potential) adaptability. Knowledge is stored as a script in a text file, where each line is started with a script command notation. These notations are single characters, one for each rule-type as shown in figure 2.6. Each script command has an index code that is generated automatically. It can also be indexed using a user special code.

<i>W</i> : Welcome message	<i>Q</i> : Quitting message
<i>N</i> : No match	<i>V</i> : Void input
<i>I</i> : Input transformation	<i>/</i> : Comment
<i>K</i> : Keyword pattern	<i>R</i> : Keyword response
<i>M</i> : Memorise phrase	<i>N</i> : No match
<i>O</i> : Output transformation	<i>&</i> : Action to be perform

Figure 2. 6 Elizabeth script command notations

2.9.1 Elizabeth's script file format

A script file may contain at most four parts:

1. Script command lines holding welcome, void and no keyword messages.
2. Input transformation rules, which start with “I” and maps input to another form to be compatible with the defined keywords.
3. Output transformation rules, which start with “O” and changes personal pronouns to be appropriate as a response.
4. Keyword patterns, which start with “K” to denote keyword pattern to be matched, followed by robot response denoted by “R”. There are two types of keyword patterns: simple patterns, which match only a single word, and composite patterns, which match sentences, phrases, words, strings, letters, anything or nothing.

2.9.2 Dynamic processing in Elizabeth

The dynamic process implies performing a set of actions which modifies the script while the conversation is under progress. These actions have the following format “& {script command}”. The dynamic process actions involve adding, memorisation and deleting commands. These processes can be performed directly using “!” or after the matching process.

1. Adding new script commands are used to add new script if it does not exist or replace it by new one if it is already found. For example,

```
I my sister => my sister
& { K MOTHER
& { N DOES ANYTHING ELSE ABOUT YOUR MOTHER? }
      R HOW WELL YOUR MOTHER AND SISTER GET ON? }
```

There are two actions to be performed. Firstly, when matching occurs with the input rule, the system will search for the keyword “MOTHER”, if finds then the response denoted by “R” will be added to the response list of that keyword. If not found then the outer curly brackets will be added to the script. Secondly, when matching with “MOTHER” occurs later on, and then a no key word message will be added to the list or replace the old one if exists.

2. Memorisation commands are used to hold the user or robot response for future matching. The command is denoted by M followed by a memory label then the value. For example,

```
K [ ] MY NAME [phrase]
& { M label [phrase] }
R YOUR NAME [phrase]?
```

If the user input is “my name is John” then the robot answer will be “YOUR NAME IS JOHN”. At the same time a memory label holding “John” is generated.

3. Deleting commands are used to delete any script command by using the back slash notation. For example,

V: deletes all current void messages.

M\one: deletes the memorization command with label “one”.

2.9.3 Elizabeth's pattern-matching algorithm

Before starting the matching process, an active text is generated for each input by converting user input to lower case, inserting spaces between words and punctuation and removing some characters from the input, except: ! " ' () , - . 0..9 : ; ? a..z. Figure 2.7 shows an example of Elizabeth code and table 2.4 illustrates the tracing output of the matching process. The matching process involves the following five stages:

1. Matching input transformation rules and record or apply any dynamic processes found. If the same rule is applicable to the active text more than 10 times in succession, then it is applied just once.
2. Matching keyword patterns where all keywords are applied in turns until one match or no match is found. The response is given according to the first match, and if there is a list of responses, one of them is selected randomly and the index code of this response will be recorded in order not to use it if the same match occurs in the next session. If there is a dynamic process, it will be applied or recorded to be performed in stage five. If no match at all found then go to step four.
3. Output transformation rules start by applying the same algorithm used in stage one and then change the active text to upper case.
4. If no match occurs with the keyword patterns then either the active text is empty, and one of the void messages will be selected randomly, or no keyword found, and one of the no keyword messages will be displayed.
5. Performing any dynamic processes and performing the recorded actions if any exist.

```

/ The Script begins with Welcome, Void, No-Keyword and Quit
responses:

`001' W HELLO, I'M Elizabeth. WHAT WOULD YOU LIKE TO TALK ABOUT?
`001' V CAN'T YOU THINK OF ANYTHING TO SAY?
`002' V ARE YOU ALWAYS CHIE ?
`001' N TELL ME WHAT YOU LIKE DOING.
`001' Q GOODBYE! DO COME BACK SOON.

/ Next come the Input transformations:

`001' I mum => mother
`002' I dad => father

/ Then the Output transformations:

`001' O i am => YOU ARE
`002' O you are => I AM
`003' O my => YOUR
`004' O your => MY
`005' O me => YOU
`006' O I IS => I AM
`007' O YOU IS => YOU ARE

/ And four groups of Keyword transformations:

`001' K I THINK [phrase]
`001' R WHY DO YOU THINK [phrase]?
`002' K MOTHER
`003' K FATHER
`001' R TELL ME MORE ABOUT YOUR FAMILY.
`002' R WHAT DO YOU REMEMBER MOST ABOUT YOUR CHILDHOOD?
`003' R ARE YOU THE YOUNGEST IN YOUR FAMILY?
`004' K [phrase1] IS YOUNGER THAN [phrase2]
`001' R SO [phrase2] IS OLDER THAN [phrase1].
`005' K I LIKE [string]ING
`001' R HAVE YOU [string]ED AT ALL RECENTLY?
`006' K [ ] my [phrase]
      & {M [phrase]}
`001' R YOUR [phrase]?

```

Figure 2. 7 An example of Elizabeth script: illustrates different keyword types

User Input	Input Transforming	Keyword Patterns	Output Transforming	Respond	Actions
Dad loves Mum	'001' mum=>mother '002' Dad => father	'001' K MOTHER '001' R	-	TELL ME MORE ABOUT YOUR FAMILY.	-
My sister is a teacher.	-	'006' K my [phrase] [phrase] => sister is a teacher '001' R	-	YOUR SISTER IS A TEACHER?	'fam' M sister is a teacher
My brother is younger than me.	-	'004' K [phrs1] IS YOUNGER THAN [phrs2] [phrs1] => my brother [phrs2] => me '001' R	[phrs1] '003' my => YOUR [phrs2] '005' me => YOU	SO YOU IS OLDER THAN YOUR BROTHER.	-
I like reading	-	'005' K I LIKE [string]ING [string] => read '001' R	-	HAVE YOU READED AT ALL RECENTLY?	-

Table 2. 4 Tracing an Elizabeth output

2.9.4 Implementing grammatical rules in Elizabeth

Elizabeth has the ability to produce a grammar structure analysis of a sentence using a set of input transformation rules to represent grammar rules. This provides an introduction to some of the major concepts and techniques of natural language processing. Considering the following grammar:

S => NP VP
 NP => D N
 VP => V NP

This can be represented by the input rules to deal with specific nouns, verbs and determiners as shown in figure 2.8.

```

I  a => (A d)
I  the => (THE d)
I  cat => (CAT n)
I  dog => (DOG n)
I  likes => (LIKES v)
I  ([brak1] d) ([brak2] n) => (([brak1] D) ([brak2] N) np)
I  ([brak1] v) ([brak2] np) => (([brak1] V) ([brak2] NP) vp)
I  ([brak1] np) ([brak2] vp) => (([brak1] NP) ([brak2] VP) s)
K  [any?]
R  [any?]
W  TYPE A SENTENCE USING:  A, THE, CAT, DOG, RABBIT, BITES,
    CHASES, LIKES

```

Figure 2. 8 Grammar representation rules in Elizabeth

This script starts from the Welcome message,

Elizabeth: TYPE A SENTENCE USING: A, THE, CAT, DOG, RABBIT, BITES, CHASES, LIKES

User: the cat likes the dog

Elizabeth: (((THE D) (CAT N) NP) ((LIKES V) ((THE D) (DOG N) NP) VP) s)

2.10 A comparison between ALICE and Elizabeth

From the previous two sections, the following conclusions were made:

- ALICE used a simple pattern template to represent input and output, and also using simple pattern-matching algorithm. Whereas Elizabeth uses input rules, keyword patterns and output rules to generate a response.
- The recursive technique used in ALICE is considered as a key feature of the system. It is used for simplifying the input by calling match categories recursively. However, the nature of some rules in Elizabeth may cause cycling or iteration, which is solved by applying the rule only once if it is applicable for the active text.
- In ALICE there is the ability to combine two answers in the case of splitting during the normalisation process, or the partitioning caused by the recursive process. This provides a way to partition the sentence to two sentences then combine their results, but this is not available in Elizabeth.
- The most significant feature in ALICE is the pattern-matching algorithm, which is easy and depends on depth-first search. This algorithm tries to

find the longest pattern match, while Elizabeth gives the response according to the first keyword pattern matched.

- Both systems can change personal pronouns; however, this is more complicated in Elizabeth due to case sensitivity. This may cause errors and give unsuitable answers.
- Both systems support memorisation. Elizabeth's dynamic process, which allows other actions to occur while the conversation is under progress, is considered a good feature which does not exist in ALICE.
- If the same input repeats during the conversation, Elizabeth tries to give different answers by using different random selection responses from the list. ALICE is based on random selection which may give duplicate responses.
- Elizabeth has the ability to give the derivation structure for a sentence using the grammatical analysis, which is not provided by ALICE.
- Elizabeth incorporates analysis tables for all steps in matching, helping the user to understand how answers are generated.

Overall ALICE was simpler and better suited for automatic retraining process proposed in this research, which is discussed in chapter 4.

2.11 Evaluation methodologies

Practical applications and evaluation are key issues in language engineering. Cunningham (1999) characterises language engineering in terms of "...its focus on large-scale practical tasks and on quantitative evaluation of progress, and its willingness to embrace a diverse range of techniques".

Some NLP systems can be evaluated by comparing output against a "gold standard" perfect target output (e.g. Hughes and Atwell 1994, Van Zaanen et al. 2004, Elliot et al. 2004) but this approach does not work for interactive dialogue systems, where output depends on user input and interaction.

The Loebner Prize (Loebner 2003) competition has been used to evaluate machine conversation chatbots. The Loebner Prize is a Turing test, which

evaluates the ability of the machine to fool people that they are talking to human. In essence, judges are allowed a short chat (10 to 15 minutes) with each chatbot, and asked to rank them in terms of “naturalness”.

Since machine conversational systems could range from question answer systems to dialogue systems, a review for dialogue evaluation methodologies is introduced in this section in addition to the Loebner Prize contest.

2.11.1 Evaluating spoken language dialogue systems

Hirschman and Thompson (1997) realise that “evaluation plays an important role for system developers (to tell if their system is improving), for system integrators (to determine which approaches should be used where) and for consumers (to identify which system will best meet a specific set of needs). Beyond this, evaluation plays a critical role in guiding and focusing research.” They defined three kinds of evaluations for spoken language dialogue systems (SLDs):

1. Adequacy evaluation, which is the determination of the fitness of a system for a purpose.
2. Diagnostic evaluation, which is the production of a system performance profile, mostly done with test suits of exemplary input.
3. Performance evaluation, which is the measurement of system performance in one or more specific areas.

Given these competing objectives for spoken dialogue system evaluation, they identify two ways to evaluate spoken language dialogue systems: “A distinction is often drawn between so-called *glass box* and *black box* evaluation, which sometimes appears to differentiate between component-wise versus whole-system evaluation, and sometimes to a less clear-cut difference between a qualitative/descriptive approach (How does it do what it does) and a quantitative/analytic approach (How well does it do what it does).” (Hirschman and Thompson 1997).

McTear (2002) illustrated that the glass box analysis of SLDs is based on evaluating individual components, with measures such as word accuracy which determines the desired output of the component to be compared with its actual output; sentence accuracy which measure the percentage of utterances in a

corpus that have been correctly recognised; sentence understanding, on the other hand, measures the rate of understood sentences in comparison with a reference meaning representation. For example, glass box evaluation was applied on the ARPA Spoken Language system (Hirschman 1995), and it shows that the error rate for sentence understanding was much lower than that for sentence recognition. On the other hand black box evaluation evaluates the system as a whole based on user satisfaction and acceptance. The black box approach evaluates the performance of the system in terms of achieving its task, the cost of achieving the task in terms of time taken and number of turns, and measures the quality of the interaction, normally summarised by the term ‘user satisfaction’, which indicates whether the user “gets the information s/he wants, is s/he comfortable with the system, and gets the information within acceptable elapsed time, etc.” (Maier et al. 1996).

Hasida and Den (1999) agreed that conversational systems must be evaluated in terms of the degree of fulfilment of the task achieved by the system, which reflects the efficiency of communication. They developed a framework named DiaLeague (Hasida and Den 1999) to evaluate natural language dialogue systems on a black box, synthetic, objective, and quantitative basis. The evaluation mechanism is based on comparing between machine-machine dialogues and human-machine dialogues.

Black box evaluation is used to evaluate many spoken systems that provide transactional services to customers and have specific tasks to achieve. For example: the Nuance automatic banking system that enables callers to conduct transactions over the phone (McTear 2002); and the Philips Automatic Train Timetable Information System that provides information over the telephone about train connections between 1200 German cities (Aust et al. 1995).

2.11.2 The Loebner Prize

The story began with the “imitation game” which was presented in Alan Turing’s paper “Can Machine think?” (1950). The imitation game has a human observer who tries to guess the sex of two players, one of which is a man and the other is a woman, but while screened from being able to tell which is which by voice, or appearance. Turing suggested putting a machine in the place of one of the humans and essentially playing the same game. If the observer can not tell which

is the machine and which is the human, this can be taken as strong evidence that the machine can think.

Turing's proposal provided the inspiration for the Loebner Prize competition, which was an attempt to implement the Turing test. The first contest organized by Dr. Robert Epstein was held on 1991, in Boston's Computer Museum. In this incarnation the test was known as the Loebner contest, as Dr. Hugh Loebner pledged a \$100,000 grand prize for the first computer program to pass the test. "for the first computer whose responses were indistinguishable from a human's. Each year an annual prize of \$2000 and a bronze medal is awarded to the most human computer. The winner of the annual contest is the best entry relative to other entries that year, irrespective of how good it is in an absolute sense" (Loebner 2003).

At the beginning it was decided to limit the topic, in order to limit the amount of language the contestant programs must be able to cope with, and to limit the tenor. Ten agents were used of which 6 were computer programs. Ten judges would converse with the agents for ten minutes and rank the terminals in order from the apparently least human to most human. The judges also rank the terminals on scales of "humanness" and "responsiveness". "Responsiveness refers to your impression of whether the conversation proceeded smoothly and normally with questions getting answered and a logical progression. If your conversational partner kept avoiding answering questions or changing the subject, that should be regarded as being unresponsive. Humanness is your subjective reaction - you could think of it as how certain you are that your partner is human." (Zdenek 2001).

However, there are sceptics who doubt the effectiveness of the Turing Test and/or the Loebner Competition. Searle (1980) objected to Turing's idea that the machine is thinking if it can fool users into believing that they are speaking to a human. In Searle's hypothetical "Chinese Room" scenario, he is sitting in a closed room with a slot in the door through which come slips of paper with questions written in Chinese characters. Searle does not understand Chinese but he does have a codebook of instructions in English, which tell him how to develop answers. He prepares the answers on more slips of paper, and pushes them through the slot. The answers make sense to the Chinese speakers outside.

To the outside observer the “room” appears to understand Chinese; but Searle argues that the room and/or its occupant do not understand Chinese; they merely follow a set of instructions to process the messages.

Hutchens (1996), Bastin and Cordier (1998) objected to the limited time to converse, which may not allow judges to make a confident decision. Other sceptics include Block (1981), who thought that “the Turing test is a sorely inadequate test of intelligence because it relies solely on the ability of fool people”; and Shieber (1994), who argued that intelligence is not determinable simply by surface behaviour. Shieber claimed the reason that Turing chose natural language as the behavioural definition of human intelligence is its “open-ended and free-wheeling nature”, which was lost when the topic was restricted during the Loebner Prize. Epstein (1992) admitted that they have trouble with the topic restriction, and they agreed “every fifth year or so ... we would hold an open-ended test - one with no topic restriction.” They decided that the winner of a restricted test would receive a small cash prize while the one who wins the unrestricted test would receive the full \$100,000. The first open-ended implementation of the Turing Test was applied in the 1995 contest.

Loebner in his responses to these arguments believed that unrestricted test is simpler, less expensive and the best way to conduct the Turing Test. Loebner presented three goals when constructing the Loebner Prize (Loebner 1994):

1. “No one was doing anything about the Turing Test, not AI.” The initial Loebner Prize contest was the first time that the Turing Test had ever been formally tried.
2. Increasing the public understanding of AI is a laudable goal of Loebner Prize. “I believe that this contest will advance AI and serve as a tool to measure the state of the art.”
3. Performing a social experiment.

In order for a machine to pass the Turing Test, Rapaport (2004) listed two conditions: the ability to understand and generate natural language. Any natural language understander must be able to do the following:

1. take coherent discourse as input,
2. understand ungrammatical input,
3. make inferences and revise beliefs,
4. make plans and understand plans,
5. construct user models,
6. learn about the world and about the language,
7. have a background knowledge,
8. and remember what it heard, learned, inferred, and revised.

Whether or not the Loebner Prize advances the field of Artificial Intelligence, it does make us aware of how little our understanding of conversation lies in what is said. The annual Loebner Prize contest encourages researchers to develop chatbots that can pass the competition as shown in table 2.5. However, this has led developers to focus on ways to meet the 10-minute challenge, rather than on how to build practical, useful information systems.

In this study some evaluation methods used within SLDs such as the black box approach were adopted to evaluate the chatbot versions generated for this research. Open-ended trials were performed by real users rather than 10-minute trials, which will be discussed in detail in chapter 5.

Year	Loebner Prize winner
1991, 1992, 1993, 1995	Joseph Weintraub won the first, second and third prize for his chatbots, PC Therapist, PC Professor and PC Politician.
1994	Thomas Whalen won the prize for his TIPS program, which provides information on a particular topic.
1996	Jason Hutchens won the prize for his program MegaHAL.
1997	David Levy won \$2,000 and a bronze medal for his program CONVERSE, which had the highest rank at that year.
1998, 1999	Robby Garner won the prize for his FRED program, which can be used as a natural language interface to many different types of software.
2000, 2001, 2004	Richard Wallace won the prize three times for his program ALICE.
2002	Kevin Copple won the prize for his program Ella.
2003	Juergen Pirner won the prize for his Jabberwok program, which represents a dragon-like talking creature.

Table 2. 5 A list of the Loebner prize winner from 1991-2004 from (Loebner 2003)

2.12 Summary

Human machine conversations and chatbots have been surveyed in terms of history, pattern-matching, and chatbot language. The chatbot language is an intermediate between spoken and written, it looks like spoken because it is informal, and it looks like written because the communication is achieved via typing.

The first chatbot was ELIZA which simulates a psychiatrist, and based on key word matching. From first impressions ELIZA is convincing, but after a while it becomes clear that she simply rearranges user's input. After that PARRY was developed to imitate a paranoid patient. Nowadays a lot of chatbots are found for different purposes.

Two chatbot systems called ALICE and Elizabeth, which were inspired from ELIZA, were discussed in detail. The analysis of ALICE and Elizabeth focused in the knowledge representation and pattern-matching algorithms for each of them. Then the main differences between them were illustrated. The comparison revealed that it was easier to retrain ALICE rather than Elizabeth. The simple pattern-matching techniques, and the ease of using AIML language, made ALICE preferable to Elizabeth for automatic retraining.

Details of evaluation methodologies used within spoken dialogue systems and the Loebner Prize contest were reviewed. However, the methods described were not suitable to evaluate our chatbot versions, but some ideas were adopted from these methods, as discussed in chapter 5.

Corpus linguistics and computational linguistics have been surveyed in this chapter. Some natural language engineering tools, information extraction and retrieval systems, man-machine interfaces were explained. Some of the corpora presented in this chapter will be used to retrain the chatbot system. Corpora problems and software implementations are presented in chapters 3 and 4 correspondingly.

3. Corpora to Train Chatbots

“A man said to Rasulallah pbuh : ‘Give me some advice’, and he said : ‘Judge each matter by its disposition. If you see good in its outcome, carry on with it;...’ ”

Prophet Mohammed (pbuh)

3.1 Introduction

Many areas of linguistics benefit from empirical corpus evidence, including syntax, semantics, pragmatics, and discourse. There are no constraints on the size, or contents of a corpus. A corpus could contain the entire work of an author, or all questions and answers related to specific domain, or all historical information about a piece of art. A corpus could be extracted from written text as well as transcribed from recorded speech, for example, interviews with people, conversations of students discussing specific modules, and conversations recorded during a business meeting.

In terms of chatbot systems, the aim is to build a system which simulates human conversations. Therefore most of the work in this thesis is based on dialogue corpus data transcribed from real human conversations. However, there is a big difference between the way we speak and the way we write. A chatbot is a compromise between the two; it looks like writing because it uses the text mode in communications, and it looks like spoken text because it simulates human conversations. So dialogue corpora usually contain a lot of features which are not found in written text, and which must be removed such as overlapping conversations, etc.

In the field of artificial intelligence (AI) or knowledge based systems, knowledge acquisition is the capturing and encoding of human expert knowledge in a computer-usable form, to be used in “intelligent” software. Natural language processing (NLP) systems require linguistic knowledge and formal models of the natural language to be processed. This linguistic knowledge generally ought to be formalised and encoded by someone who is both an expert in the language in question, and in appropriate AI formalisms. However, this combination of expertise is scarce outside the main European Union languages. A possible

alternative is to try to separate the roles of linguist and knowledge engineer, and replace the latter with a machine learning approach.

Ideally, to automatically learn a language model for a specific NLP application, a training corpus is required, which includes sufficient representative examples of the target linguistic behaviour. In practice, a compromise is often needed, and the readily available corpus is used rather than the ideal corpus for our needs. For example, many large-vocabulary speech recognition systems for European languages use language models trained with a corpus of newspaper texts; this is not because these systems are built specifically to recognise newspaper text read out loud, but because newspaper text is readily available in large quantities, and is judged to be reasonably close to the language style or genre of dictation.

To train a chatbot to converse in different languages and serve in multiple domains, ideally a variety of corpora with special characteristics are essential. In this chapter various corpora are presented, which are either available via the Internet or offered by linguists. Samples of these corpora and the problems related to each are discussed in detail in the following sections.

3.1 The Dialog Diversity Corpus

The Dialogue Diversity Corpus (DDC) (Mann 2002) is a collection of dialogue corpora in different fields where each corpus has its individual annotation format. These annotated texts are transcribed from recorded dialogues between two or more speakers. The DDC is composed of the corpora below:

1. MICASE Corpus.
2. CIRCLE Corpus.
3. CSPA Corpus.
4. The Trains Dialogue Corpus.
5. ICE-Singapore.
6. Mishler Book.

Unfortunately each corpus has its own format and annotations. Problems of each corpus are discussed in the following sections.

3.1.1 MICASE corpus problems

The Michigan Corpus of Academic Spoken English¹ (MICASE), is a collection of transcripts of academic speech events recorded at the University of Michigan. A sample of this corpus is shown in figure 3.1.

S1: circumpolar stars. So if I keep my pointer there, [**S2: oh**] **<ROTATES CEILING>** everything else moves and we all get sick. **<SS LAUGH>** and we go backwards in time. And that's even more fun.

S2: make it go really fast.

<SS LAUGH>

S1: Okay so that's how the sky is going to move, a couple of other things that we can do in here, um, this is a presentation of, the, grid, that we use to divide the sky, so these lines that run, north south what do we call those?

S3: declination

Figure 3. 1 Astronomy transcripts sample/MICASE corpus

The Astronomy transcript was made using a digital audio tape recorder with two microphones. Problems examined are:

- Long monologs.
- Overlapping denoted by square brackets [].
- Using more than two speakers.
- Using extra annotations to record actions denoted by angle brackets such as **<SS LAUGH>**.

3.1.2 CIRCLE corpus problems

The CIRCLE² Corpus, collected by the Center for Interdisciplinary Research on Constructive Learning Environments, is a collection of transcripts holding

¹ <http://www.hti.umich.edu/m/micase/>

different tutorial sessions, on topics such as physics, algebra and geometry. An Algebra transcript sample is shown in figure 3.2 while a Physics transcript sample is shown in figure 3.3.

TUTOR [Opening remarks and asks student to read out aloud and begin]

STUD [Reads problem] Mike starts a job at McDonald's that will pay him 5 dollars and hour, Mike gets dropped off by his parents at the start of is shift. Mike works a "h" hour shift. Write an expression for how much he makes in one night?

[Writes "h*5 = how much he makes"]

TUTOR That's right number.

Figure 3. 2 Algebra transcript sample/CIRCLE corpus

T: [student name], I'd like you to read the problem carefully, and then tell me your strategy for solving this.

S: ok

[Pause 17 sec]

hmm.

[Pause 6 sec]

T: thinking out loud as much as possible is good

Figure 3. 3 Physics transcript sample/CIRCLE Corpus

The Algebra transcript was made from a videotape of an algebra session where the tutor and the student are sitting in the same room. The Physics transcript was done from audiotapes of a phone conversation, where the tutor was seated in a room and students were seated in a different one. The tutor and students communicated with a conference phone that let only one person talk at a time.

The main problem here is using different format within the same corpora such as:

- Different formats were used to distinguish speakers.

² <http://www.pitt.edu/~circle/Archive.htm>

- Different linguistic annotations were used to indicate pauses. For example, colons are used in the Algebra corpus whereas in the Physics corpus it was directly written inside brackets.

3.1.3 CSPA corpus problems

The Corpus of Spoken Professional American-English³ (CSPA) includes transcript conversations of various types occurring between 1994 and 1998 as shown in figure 3.4.

LANGER: Hello, I'm delighted to be here. I have carefully read and heard about the University of Albany, the State University of New York. And I'm also the director of the National Research Center on English Learning and Achievement.

STRICKLAND: Her mother wrote the stances.
(Laughter)

KAPINUS: Dorothy, I might add also that Judith probably has more history with NAEP than just about that I know of, you know, NAEP and reading.

STRICKLAND: Yes, yes. We will really turn to you as a very important resource, Judith. And we have a new member, Gloria Lopez Gutierrez. And, Gloria, tell us a little bit about yourself.

Figure 3. 4 CSPA corpus sample

In this transcript, the problems noticed are:

- Long turns/monologues.
- The transcripts were not “anonymised”: speakers’ names were used.

³ <http://www.athel.com/corpdes.html>

3.1.4 The TRAINS dialog corpus problems

The TRAINS⁴ corpus is a corpus of task-oriented spoken dialogue that has been used in several studies of human-human dialogue. For example, it is used to study the prosody of discourse structure. Figure 3.5 shows a sample of the TRAINS corpus. The main problem encountered, is the extra-linguistic annotation like +, <sl>, <brth>, <click> tags, and using capital letters to denote proper names, i.e., E two denotes an engine name E2. The silence tag <sl> and the breath tag <brth> are used to denote word duration within the utterance. The <click> tag denotes obvious tongue clicking.

```

utt11 : u:  okay <sil> um <sil> engine <sil> two
utt12 : s:  + okay +
utt13 : u:  + from + Elmira
utt14 : s:  + mm-hm +
utt15 : u:  + to + Corning
utt16 : s:  okay
utt17 : u:  and then <sil> do I need anything else or can I
           go straight to Bath
utt18 : s:  um you'll need a <sil> boxcar to move the
           oranges from Corning to Bath
utt19 : u:  okay <sil> uh <sil> take the boxcar from
           Dansville <click> <sil> to Corning
utt20 : s:  okay
utt21 :    so we'll have to send uh the engine <sil> what
           E two <brth> <sil> uh from Elmira <sil> to
           Dansville to get the boxcar

```

Figure 3. 5 TRAINS corpus sample

⁴ <http://www.cs.rochester.edu/research/cisd/resources/trains.html>

3.1.5 ICE-Singapore problems

ICE-Singapore is the International Corpus of English, Singapore English (Nelson 2002).

<\$B>
<ICE-SIN: S1A-099#33:1:B>
 How how are things otherwise
<ICE-SIN:S1A-099#34:1:B>
 Are you okay

<\$A>
<ICE-SIN:S1A-099#35:1:A>
 Uhm okay lah
<ICE-SIN:S1A-099#36:1:A>
 Bearing up lah

<\$B>
<ICE-SIN:S1A-099#37:1:B>
 Ah hah

<\$A>
<ICE-SIN:S1A-099#38:1:A>
 Ya I mean I don't really feel comfortable talking about it
 over the phone so when I see you I'll tell you about it lah

Figure 3. 6 ICE-Singapore corpus sample

In figure 3.6 <ICE-SIN:S1A-099#35:1:A> refers to text unit 35, in subtext 1, uttered by speaker A. ICE-SIN refers to corpus name ICE Singapore and S1A refers to academic spoken. This is a spoken dialog recorded through the phone. The problems investigated here are:

- Unconstrained conversations.
- A lot of corpus mark-up annotation.
- Great variation in turn length is noticed.

3.1.6 Mishler Book

Medical Interviews (Mishler 1985) is an example of a scanned text image, including a dialogue between a patient and his physician. In addition to using extra-linguistic annotations, the main problem is that scanned image was not converted to text format.

A lot of problems arise when dealing with the DDC, as illustrated, which can be summarised as follows:

- No standard formats to distinguish between speakers.
- Extra-linguistic annotations were used.
- No standard format in using linguistic annotations.
- Long turns and monologues.
- Irregular turn taking (overlapping).
- More than two speakers.
- Scanned text-image not converted to text format.

3.2 The Korpus Gesprooke Afrikaans corpus

The Korpus Gesprooke Afrikaans (KGA) (Van Rooy 2003) is a set of transcripts of human group conversations. As the language was not known, it was not easy to state whether its content was suitable. Some of the problems discussed in this section can be explained in terms of inappropriateness of the training corpus. A sample of the Afrikaans corpus is shown in figure 3.7.

```

<spreker1> nee hy't Donderdag twee gedruk <oorvleuel>
<spreker1> en Din~ </spreker1>

<spreker2> teen wie't hulle </spreker2> </oorvleuel>
</spreker1>
<spreker2> gespeel </spreker2>

<spreker1> teen <fil> uh uuhm </fil> Proteapark </spreker1>

<spreker2> *a+ gewen </spreker2>

<spreker1> vyf-en-vyftig nul en Dinsdag het hulle agt-en-
tagtig nul gewen </spreker1>

<spreker2> jissie toe wie't hulle *f+ wie't hulle gespeel
<oorvleuel>
<spreker2> altwee ke~ o Elandskraal </spreker2>

<spreker1> Elandskraal Elandskraal </spreker1>
</oorvleuel> Elandskraal o </spreker2>
<spreker1> en <fil> ih </fil> <oorvleuel>
<spreker1> Christoff-hulle </spreker1>

<spreker2> en Christoff </spreker2> </oorvleuel> hulle
eerste span het Dinsdag <fil> uuh </fil> vyf-en-dertig vyf
gewen en gister het hulle drie-en-vyftig nul gewen
</spreker1>
<spreker2> (lag) </spreker2>

```

Figure 3.7 A sample of the Korpus Gesprooke Afrikaans corpus

3.2.1 Afrikaans corpus problems

The KGA was transcribed from recorded conversations between more than two speakers. A number of problems were encountered, analogous to those found in the analysis of English spoken corpus texts in the DDC as described in section 3.1. The sources of the problems could be classified into two categories: the first is from human conversation characteristics; the second is from human transcription problems.

1. Problems resulting from human conversation characteristics:
 - Irregular turn taking and overlapping.
 - Long turns, amounting to monologues.
 - More than two speakers.

2. Syntax problems resulting from human transcription:
 - a. Syntax errors in using mark-up to annotate the dialogue:
 - Spelling errors in opening and closing tags, which leads to different structure format in dealing with them like <oorveuel>, and <oorlveuel> to denote the overlapping.
 - Unequal number of open and close tags especially in nested overlapping, and speaker recognition turns.
 - Syntax errors leading to difficulties in recognising when speakers begin and end turns.
 - b. Syntax errors in adding extra-linguistic annotations:
 - Opening a string with “*” and forgetting to close it by “+”.
 - Syntax errors in using filler tags <fl> and <fil>.
 - Using parentheses to denote the unclear words. This should be replaced with another appropriate word by a linguist.

3.2.2 Principles adopted to solve the KGA problems

The following principles were adopted to solve the corpus mark-up analysis problems in three stages:

1. Correcting the spelling mistakes as discussed below:
 - Replacing all “oorveuel” with “oorvleuel”,
 - Replacing all “fl” with “fil”.
 - Removing all white spaces in “spreker” tags.
 - Adding “/” to the closing tags where it is missing.
2. Removing overlaps since they will not occur between two speakers in a chatbot system, and removing the useless linguistics annotations such as:
 - Removing “fillers” speech that is written between <fil> and </fil> tags.
 - Removing any unclear text, marked up between * and +.

- Removing any text between parentheses; sometimes this results in ungrammatical sentences, but finding a suitable replacement requires linguistic knowledge.
 - Removing speaker's tags <speaker1>, etc.
3. Ignoring the whole turn that involved nested overlapping sessions without sufficient numbers of open and closed tags.

Long turns were kept verbatim, which resulted in some unexpectedly long and involved replies to user input! In order to solve the problem of having more than two speakers, each turn was duplicated; each utterance was used as a pattern in one category, and a template in the next one (see chapter 4).

3.3 The British National Corpus

It took several years for the ALICE Botmaster to accumulate the 50,000 categories in the current public-domain set of AIML files (Wallace 2003). The BNC was used to investigate the possibility of using learning techniques to extract a much larger set of AIML files. In theory, the chatbot-learning program can learn millions of categories given an appropriate dialogue corpus.

The British National Corpus (BNC)⁵ (Aston and Burnard 1998) is a collection of text samples amounting to over 100 million words, extracted from 4,124 modern British English texts of all kinds. The corpus is annotated using SGML (XML-like) mark-up, including CLAWS Part-of-Speech category of every word. All annotations are marked between <angle brackets>. The corpus is partitioned into two types: the spoken and the written transcripts.

The spoken dialogue transcripts amount to 10 million words, and can be divided into two parts:

⁵ <http://www.natcorp.ox.ac.uk/corpora.html>

1. A demographic part, involving transcriptions of spontaneous natural conversations between families, friends, and so forth.
2. The context-governed part, containing transcription recorded in educational, informative, business, leisure, institutional, and public events.

Each corpus file starts with a long header section, containing details of source, speakers, etc. In the transcript body, the dialogue consists of a series of utterances or speaker-turns, marked at start and end by `<u>` and `</u>` tags. Each utterance tag also includes a speaker number (anonymous, e.g. F72PS002). Within a text sample, all sentences are tagged `<s>` and numbered; and each word is preceded with a CLAWS Part-of-Speech tag, e.g. ITJ = interjection, PUN = punctuation mark, NP0 = singular proper name. An example of a sequence of two utterances is shown in figure 3.8.

```

<u who=F72PS002>
<s n="32"><w ITJ>Hello<c PUN>.
</u>
<u who=PS000>
<s n="33"><w ITJ>Hello <w NP0>Donald<c PUN>. </u>

```

Figure 3. 8 A sample of a BNC spoken transcript

3.3.1 The BNC problems

The translation process from BNC format to AIML is not as simple as it might seem to be on the surface. A range of problems emerged during the translation process, some of them are common with almost all dialogue corpora including the KGA, but were treated in different way. The BNC problems can be classified as one of:

1. More than two speakers and long monologues

With long monologues, the speaker's utterance is composed of a set of sentences. These sentences are denoted by sequential numbers as shown in figure 3.9.

```

<u who=F72PS000>

<s n="29"><w PNP>You <w VDB>do<c PUN>?

<s n="30"><w AV0>Well <w PNP>you <w VBB>are <w
AV0>very <w AJ0>fortunate <w NN0>people<c PUN>.

<s n="31"><w CJC>But <w PNI>none <w PRF>of <w PNP>you
<w VM0>will <w VVI>know <w DPS>my <w NN1>friend <w
AV0>over here <w DTQ>whose <w NN1>name <w VBZ>is <w
NP0>Donald<c PUN>. </u>

```

Figure 3.9 A sample represents long monologue in the BNC

2. Unclear sections

Since all spoken samples are transcribed from recorded speech, some parts of the utterances were unclear. The BNC uses the <unclear> tag to mark these, as exemplified in figure 3.10. A problem with the unclear turn is that it might be a response to a previous utterance, or it might introduce a new idea, which the next speaker responds to. In the translation to AIML, it was not obvious if the unclear turn will be treated as a pattern or a template.

3. Overlapping utterances

In case of overlapping where more than one speaker was active at the same time, the BNC corpus transcribed the overlapping turns using an alignment map tag <align> to synchronise points within a spoken text, declared at the start of the division or text concerned; and the pointer tag <ptr target...> points to the identifier which was synchronised. Figure 3.11 illustrates this problem.

```

<u who=PS000>
<s n="5"><unclear> <w AT0>a <w NN1>minute<c PUN>.
</u>

<u who=PS100 ><unclear ></u>

<u who=F72PS000>
<s n="6"><w CJC>And <w DTQ>what <w VBB>are <w
PNP>they<c PUN>?
</u>

```

Figure 3. 10 A sample represents unclear utterances in the BNC

```

<u who=w0014>
<s n=00011><w AJ0>Poor <w AJ0>old <w
NP0>Luxembourg'<w VBZ>s
<w AJ0-VVN>beaten<c PUN>.
<s n=00012><w PNP>You <w PNP>you<w VHB>'ve <w
PNP>you<w VHB>'ve
<w AV0>absolutely <w AV0>just<w VVN>gone <w
AV0>straight
<ptr target=P1> <w PRP>over <w PNP>it <ptr target=P2>
</u>

<u who=w0001>
<s n=00013><ptr target=P1> <w PNP>I <w VHB>haven<w
XX0>'t<c PUN>.
<ptr target=P2/> </u>

<u who=w0014>
<s n=00014><w CJC>and <w VVN>forgotten <w AT0>the <w
AJ0>poor
<w AJ0>little<w NN1>country<c PUN>. </u>

```

This is actually equivalent to:

W0001: Poor old Luxembourg's beaten. You, you've,
you've absolutely just gone straight over it.

W0014: (interrupting) I haven't.

W0001: (at the same time) and forgotten the poor
little country.

Figure 3. 11 A sample showing the overlapping problem in the BNC

4. Other problems

a. Extra-linguistic entities

The spoken transcripts include a set of paralinguistic phenomena such as: voice quality (whispering, laughing, etc.), non-verbal but vocalised sounds (coughs, humming noises), non-verbal and non-vocal events (animal noises, passing lorries), significant pauses (silence) and speech management phenomena (truncation, false starts) as shown in figure 3.12.

```
<u who=PS21K>
<s n="37"><w CRD>forty <w NN0>percent <w PRF>of <w
DPS>her <w NN1>time <w CJS>because <w PNP>she <w
VDZ>does <w PNP>it <w AV0>so
<w AV0>quickly <b vocal desc=laugh> <w CJC>but <w
UNC>er <w ITJ>oh </u>
```

Figure 3. 12 Extra-linguistic entities cause problems in the BNC

b. Using abbreviations

The encoders used some abbreviations in writing the recorded speech as: *I'd*, *don't*, 's, and so on. The problem arises in converting such abbreviations to the AIML patterns. All punctuation should be removed from the pattern to be accepted by the ALICE interpreter. To date the machine-learnt models have not included linguistic analysis of mark-up, such as grammatical, semantic or dialogue-act annotations (Atwell 1996b, Atwell et al. 2000a), as ALICE makes no use of such linguistic knowledge in generating conversation responses. It cannot distinguish if 's is an abbreviation to "is" or a possessive.

c. Using character entity references

Some spoken texts were transcribed using HTML character entity references, such as "à," "ö," and so on. Unfortunately these special accented characters are rejected during compilation of the AIML files.

3.3.2 The principles adopted to solve the BNC spoken text problems

In order to solve the BNC spoken text problems, the following principles were adopted:

1. The problem of having more than two speakers was handled in the same manner as the KGA corpus, by reusing each utterance to be a pattern in one category and a template in the next one.
2. To deal with long monolog utterances, the program merged all of them to form one string starting with `<u who=...>` and ends with `</u>`. The merging process generates a very long turn based on the number of sentences and the length of each, which rarely happens in chatting.
3. The unclear turns were removed. There are two approaches for elimination, either before or after the converter module maps pairs of successive utterances into pattern-template categories. The difference between them is as follows:

Assume that there are four speakers denoted by (spk) and the sequence of turns is: `spk1 → spk2 → unclear → spk3 → spk4`.

The first approach is to omit the unclear turn itself before converting the transcript. In this case we will have the following sequence of utterances: `spk1 → spk2 → spk3 → spk4`. The conversion process will generate three categories: `(spk1 → spk2)`, `(spk2 → spk3)`, `(spk3 → spk4)`.

The second approach is to omit the unclear turn after the conversion. After considering each pair as a pattern and a template, the result is: `(spk1 → spk2)`, `(spk2 → unclear)`, `(unclear → spk3)`, `(spk3 → spk4)`. Then any pair containing the unclear is excluded, leaving two categories.

The second approach completely sidesteps the problem by deleting the two cases where the unclear is a pattern or a template, and this means avoiding the category where `spk3` is a response to `spk2`, which did not actually happen during the conversation. However, it is arguably possible to consider `spk3` as a possible response to `spk2`, even if it did not really happen in this sequence, as at least the utterance is still about the same

topic. As the goal was to generate a large set of categories automatically, the first approach was adopted.

4. In contrast to an earlier KGA corpus, where all overlapped turns were simply removed, the BNC overlapped utterances were treated as new turns. This added extra categories in keeping with the main aim of using the BNC to generate a large amount of categories.
5. The linguistic entities might be of interest for other purposes, but they will never occur in chatting with a computer. Therefore, all linguistic annotations were removed including the POS tags.
6. To deal with abbreviations, it was decided to remove all punctuation without extending the original form. For example “I’d...” will be mapped into “I d...”, this is still compatible with the learning approach, as whether “d” denotes “had” or “would”, it will not be a keyword in the sentence. The “n’t” abbreviation was the only one replaced by a whole word “not”, so “haven’t” becomes “have not” instead of being “haven” which is a different word with different meaning.
7. All entity references were replaced with the corresponding unaccented letters. For example: “à” is mapped to “a”, and “ö” to “o”.

3.4 The Qur’an corpus

Up to now only dialogue corpora have been described. The aim was to generalise a chatbot system using different corpora in different languages and domains. In this section the Qur’an, the holy book of Islam, is used as an example of the Arabic language, and a text which is not conversational in its nature.

The Arabic language differs from English and other European languages in many aspects, which have hindered progress in Arabic natural language processing.

These aspects are (De Roeck 2002, Al-Daimi and Abdel-Amir 1994, Atwell et al. 2004):

- Arabic is a derivative language. Words are typically derived from roots that are just three consonants.
- Interpretation of Arabic is based on the vowelizing. Some Arabic texts are vowelized such as the Qur'an and others are not such as Arabic newspapers. In the latter case readers must infer the intended meaning from the context of the word.
- The written direction is from right-to-left and some of the characters change their shapes based on their location in the word.
- Capitalisation is not used in Arabic, which makes it hard to identify proper nouns, acronyms, and abbreviations.

The Arabic language has 28 consonants and three vowels: a, i, u that can be short or long. Vowels can be omitted in Arabic texts, resulting in either a vowelized text such as the language of the Qur'an or an unvowelized text such as newspapers or most published text. The Qur'an text is available via the Internet; and in principle the Qur'an provides guidance and answers to religious and other questions. In fact, two corpora were used; one in Arabic and the other is in bilingual aligned Arabic-English translations.

The Qur'an version handles the vowelized Arabic text in the same manner as other languages without any attempt to extract the root or understand the meaning. Before retraining ALICE with the generated AIML files, these files must be refashioned to enable the ALICE interpreter to recognise the Arabic characters; and to be accepted by the Pandorobot server. This implies encoding the files using utf-8 code as described in section 3.4.2.

3.4.1 Problems of learning from the Qur'an text

The Qur'an is the holy book of Islam, written in the classical Arabic form. The Arabic language (both style-wise and content-wise) in the Qur'an is considered as one of its miracles. The Qur'an consists of 114 sooras, which could be considered as sections, grouped into 30 parts (chapters). Each soora consists of more than one Ayya (sentence). These ayyas are sorted, and must be shown in the same sequence. Sample of the Qur'an in Arabic and English is shown in figure 3.13.

(113) سورة الفلق

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

قُلْ أَعُوذُ بِرَبِّ الْفَلَقِ {113/1} مِنْ شَرِّ مَا خَلَقَ {113/2} وَمِنْ شَرِّ غَاسِقٍ إِذَا
وَقَبَ {113/3} وَمِنْ شَرِّ النَّفَّاثَاتِ فِي الْعُقَدِ
{113/4} وَمِنْ شَرِّ حَاسِدٍ إِذَا حَسَدَ {113/5}

THE DAYBREAK, DAWN, CHAPTER NO. 113

With the Name of Allah, the Merciful Benefactor, The Merciful Redeemer

113/1 Say: I seek refuge with the Lord of the Dawn

113/2 From the mischief of created things;

113/3 From the mischief of Darkness as it overspreads;

113/4 From the mischief of those who practise secret arts;

113/5 And from the mischief of the envious one as he practises envy.

Figure 3. 13 A sample of the Qur'an in Arabic and English.

Each soora in the Qur'an has its own name and number. And each verse has a unique number representing the soora number and its order. The first line in figure 3.13 represents the soora title and number. The second line is the opening statement, which appears in every soora except soora number (9). The following lines hold the verses, where each verse is treated as a section, starting with a number which represents its identity. The numerator holds the soora number, where the denominator holds the verse number in that soora.

As this text is not a dialogue, problems which result from human conversation will not appear here. However, other problems arose such as:

1. How to divide a non-conversational text into utterance-like chunks.
2. How to enable the ALICE interpreter to recognise Arabic characters.
3. How to deal with long verses, which run over more than two lines.

3.4.2 The adopted principles to solve the Qur'an problems

In order to find answers for the previous questions (problems), the following assumptions were applied:

1. The duplication process used with the KGA and the BNC, to solve the problem of having more than two speakers, were applied to solve the problem of the non conversational nature of the Qur'an. However, a different hypothesis was proposed: if an input is an ayyaa then the answer would be the next ayyaa in the same soora. For this purpose, each ayyaa must be a pattern to guide us to the succeeding ayyaa, and a template for the previous one.
2. The problem of recognising the Arabic letters arose during chatting (run time) with the Arabic-English version. This problem simply solved by encoding the AIML files using utf-8 code. And this necessitates replacing the default AIML header `<?xml version="1.0" encoding="ISO-8859-1"?>` with `<?xml version="1.0" encoding="utf-8"?>`. It was also necessary to use Arabic-enabled keyboard and software to input Arabic text to the chatbot.
3. In the same manner of merging long monologue utterances in the BNC, the long verses of the Qur'an which run over more than two lines are concatenated into one line.

3.5 The SoC Frequently Asked Questions (FAQ) corpus

Fraser (1997) defines spoken dialogue systems as “computer systems with which humans interact on a turn-by–turn basis and in which spoken natural language plays an important part in the communication”. McTear (2002) encountered many systems that are covered by this definition “ranging from question-answer systems that answer one question at a time to “conversational” systems that engage in an extended conversation with the user”.

The chatbot plays the same role as a dialogue system except that the input is in text mode instead of spoken mode, and agrees with the general goal of the automation process; so it was appropriate to try question-answer databases as training corpora for our program. The Frequently Asked Questions (FAQ) website of the School of Computing (SoC) at University of Leeds was selected to:

1. Explore the problems with FAQ databases when converted to AIML format.
2. Explore how to manipulate the system to cope with FAQ databases.

3.5.1 The FAQ problems

The FAQ of the School of Computing is a structured database; most of the “data-cleaning” problems raised with previous corpora are not found here. Moreover almost all HTML tags are recognised by the ALICE interpreter because the AIML definition allows HTML tags to be embedded within templates. The questions and answers were extracted from the HTML files of the FAQ. The following cases need to be taken into consideration when adapting the Java program to deal with the file format used in the School of Computing FAQ website:

1. Each file has a title, and an interface to scroll up and down in the page during navigation, as shown in figure 3.14. The interface is not necessary and can be treated as extra annotations.
2. Some questions are marked by: `<DIV CLASS=“sect1”>` tag and the answers by: `<p>` as shown in figure 3.15.
3. Some questions are marked by: `<DIV CLASS=“question”>` tag and the answer by: `<DIV CLASS=“answer”>` as shown in sample 3.16.
4. Another problem illustrated in figure 3.15 is the reference problem, such as in the question “What is it?”: what does “it” refer to?
5. Using special character entities denoted by the “&” sign; these are not allowed in AIML.

```

<TR><TD WIDTH="10%" ALIGN="left" VALIGN="bottom">
<A HREF="appa02.html" ACCESSKEY="P">Prev</A></TD>
<TD WIDTH="80%" ALIGN="center" VALIGN="bottom">Maintenance
of the FAQ</TD>
<TD WIDTH="10%" ALIGN="right" VALIGN="bottom">
<A HREF="appa04.html" ACCESSKEY="N">Next</A></TD></TR>

```

Figure 3. 14 The extra notations used within SoC FAQ files for navigation

```

<DIV CLASS="sect1">
<H1 CLASS="sect1">
<A NAME="x0101">
</A>What is it?</H1>
<P>The <SPAN CLASS="acronym">FAQ
</SPAN> contains information, mainly in question and
answer form, for students and staff on many aspects of
the School of Computing, concentrating on the School's
computing facilities.
</P> <P>It contains advice on: </P><P></P>
<UL><LI>
<P><A HREF="x02.html">basic computer usage
</A> (logging in, changing passwords)
</P></LI> <LI>
<P>location and use of <A HREF="x09.html">computer
laboratories</A></P></LI>

```

Figure 3. 15 Questions-answers denoted by <DIV CLASS="sect1"> and "<p>" tags

```
<DIV CLASS="question">
<P> <A NAME="AEN598"></A>
<B>1. </B>
<SPAN CLASS="bold"><B CLASS="emphasis">What rules are there
for the use of          computing facilities?
</B> </SPAN> </P></DIV>
<DIV CLASS="answer">
<P>
<B> </B>The School has a <A HREF="aup.html">Policy on the
acceptable          use of computing facilities</A>. All
persons wishing to use the          School's computing
facilities will be required to read, agree to,          and
sign the acceptable use policy.</P> </DIV>
```

Figure 3. 16 FAQ Questions-answers denoted by “question” and “answer” tags

Basically, the above are not linguistic problems; they are points to be taken into consideration during the programming. To solve the problem of the reference resolution, it was noticed that most of the references are the titles of the file which was used in the replacement process. The entity characters were replaced by normal alphabet characters as done within the BNC corpus.

3.6 Summary

All corpora used to retrain the ALICE chatbot are discussed in this chapter. The investigated training corpora are classified into three types according to the text nature:

1. Dialogue corpora such as DDC, KGA, and BNC.
2. Monologue corpora such: Arabic/English Qur'an.
3. Structured database such as the SoC FAQ corpus.

The aim of studying this variety is to build a program which is generic that can be easily adopted to accept any text type in any language and domain.

A lot of problems resulted from using these corpora which are summarised below:

- No standard formats to distinguish between turn-taking speakers
- Extra-linguistic annotations were used, which need to be recognised and filtered.
- No standard format in using linguistic annotations.
- Long turns and monologues.
- Irregular turn taking (overlapping).
- More than two speakers.
- Scanned text-image not converted to text format.
- Transcription errors such as spelling errors, and incompatible numbers of opening and closing tags.
- Non-standard characters, and non-Roman alphabets.
- How to divide non-conversational text into utterance-like chunks.

It is clear that to solve these problems to obtain a good dialog model, a filtering process must be applied first to these transcripts to remove all unnecessary tags and other mark-up which will be discussed in details in chapter 4. Because of the variation in formats and annotations for these transcripts even in the same

corpus, each transcript has to be filtered and processed differently, which contradicts the generalisation objective.

When re-engineering ALICE to a new domain or conversation style, the patterns and templates learnt from a training corpus are only a raw prototype: the chatbot and AIML files must be tested and revised in user trials. One of the main design considerations is how to plan the dialogue. A good dialogue design would mean less time testing and modifying AIML files.

In order to train a chatbot system with minimal need to post-edit the learnt AIML, dialogue corpora should have the following characteristics:

- Two speakers.
- Structured format.
- Short, obvious turns without overlapping, and without any unnecessary notes, expressions or other symbols that are not used when writing a text.

Even such “idealised” transcripts may still lead to a chatbot which does not seem entirely “natural”: although the aim is to mimic the natural conversation between humans, the chatbot is constrained to chatting via typing, and the way we write is different from the way we speak.

4. Retraining ALICE

"Make things easy and convenient and don't make them harsh and difficult..."

Prophet Mohammed (pbuh)

4.1 Introduction

"More and more people in every branch of information science are coming to realise that a corpus as the sample of the living language, accessed by sophisticated computers, opens new horizons." (Sinclair 1991).

The early concerns of corpus linguistics were how to handle a large corpus. McEnery and Wilson (1996) ask "Can you imagine searching through an 11-million-word corpus...with nothing more than your eyes? The whole undertaking becomes prohibitively time-consuming. It also becomes very expensive and error prone". This was accompanied with the great evolution in the computer industry, which led to software tools that "model (and, sometimes, verify or falsify) fragments of linguistic theories deemed of particular interest." (Boguraev et al. 1995). Different software tools were built to store, retrieve, and analyse data using a machine readable corpus as discussed in section 2.1.1.

Instead of building tools to study the language, recent trends of researchers and developers move toward building tools that communicate with users using natural language. "Human-computer interaction is a discipline concerned with the design, evaluation and implementation of interactive computing systems for human use and with the study of major phenomena surrounding them." (Hewett 1992).

In this study the aim is to use the corpus-based approach to generate different versions of a chatbot simulating real human conversations in different languages. Two chatbot engines, ALICE and Elizabeth, were discussed in detail in chapter 2. Different training corpora were discussed in chapter 3. In this chapter the chatbot and the corpus based approach will be merged together. The different

corpora will be used to retrain the ALICE chatbot system using a Java program which is built for this purpose.

The previous analysis of ALICE and Elizabeth, described in chapter 2, focused on the linguistic knowledge representation formalisms used by each system, rather than the algorithm implementation details. The aim of the comparison was to select a system to be retrained using a corpus-based approach. The decision was to use ALICE for this purpose for the following reasons:

1. AIML is closer to the XML mark-up format used in many annotated corpora.
2. AIML uses simpler categories in terms of patterns and templates.
3. ALICE is available in Java via a Web interface, whereas Elizabeth is implemented in Delphi and must be installed on individual user PCs, making web-based evaluation of the retrained versions of ALICE more straightforward.
4. There is a large, flourishing ALICE/AIML developers/users community worldwide, providing open-source software infrastructure, and opportunities for discussion, advice and evaluation.

4.2 Software developments

To retrain ALICE using a corpus-based approach, a Java program was developed to convert a corpus involving real human dialogues to the ALICE chatbot language model format (AIML). The methodology to achieve this followed the general phases used in developing software:

1. Software requirements specification.
2. Software design and implementation.
3. Software testing and evolution, where users played a key role in guiding systems development (Atwell et al. 2000b).

4.2.1 Software requirements specification

The corpora described in chapter 3 are used to retrain ALICE, as summarised in table 4.1.

Corpus	Language	Content
DDC	English	A collections of links to different spoken dialogue corpora
KGA	Afrikaans	Transcripts of General Afrikaans conversations
Spoken BNC transcripts	English	Spoken transcripts in different domains
Qur'an in Arabic	Arabic	Arabic monologue text
Qur'an in English	English input, and Arabic-English output	English monologue text
FAQ of the School of Computing at University of Leeds	English	Frequent Asked Questions relating to the School of Computing

Table 4. 1 The trainable corpora

However, using a corpus is not that simple; a lot of problems arose while using monologue and dialogue corpora. The most common problems between almost of dialogue corpora that have been used are related to the characteristics of the human conversation as described in chapter 3, which are:

- More than two speakers.
- Long-turn monologues.
- Overlapping and irregular turn taking.
- Unclear utterances.

In this chapter the ideas adopted to solve these problems are described alongside the program elaboration.

4.2.2 Software design and implementation

An object-oriented model has been implemented using Java to develop the system. The system is formed from several modules that are separated in more than one class, which interact with each other to achieve the specifications. The basic architecture of the system is composed of four phases as follows:

- Phase one: Reading module, which reads the dialogue text from the basic corpus and inserts it into a list.
- Phase two: Text reprocessing module, where all corpus and linguistic annotations such as overlapping, fillers and others are filtered.
- Phase three: Converter module, where the pre-processed text is passed to the converter to consider the first turn as a pattern and the second as a template. All punctuation is removed from the patterns, and the patterns are transformed to upper case.
- Phase four: Producing the AIML files by copying the generated categories from the list to the AIML file.

Those phases were elaborated according to the corpus annotations and the test results as appropriate to satisfy the users.

4.2.3 Software testing and evolution

After creating AIML files for the previous corpora, the Pandorabot web-hosting service¹ was used to publish different versions of ALICE to make them available for use over the World Wide Web. Users were asked to chat with these versions and provide their feedback.

Based on user feedback and the retrained corpus, five system prototypes were generated to satisfy users' expectations. The key issue in building these prototypes was how to expand the knowledge learned from the corpus to extend the chances of finding a match. The ALICE pattern matching technique was described in chapter 2. The idea is based on finding the best match which is the

¹ <http://www.pandorabots.com/pandora>

longest one. Since the input will not necessarily match exactly a whole sentence extracted from the corpus, other learning techniques were adopted.

In each prototype, different learning techniques were used and a new chatbot was tested. The learning techniques ranged from a primitive simple technique to more complicated ones. Building atomic categories and comparing the input with all atomic patterns to find a match is an instance based learning technique (see section 2.6). However, the learning approach does not stop at this level, but it improved the matching process by using the most significant words. This increases the ability of finding a nearest match by extending the knowledge base which is used during the matching process.

Listed below are the prototypes ordered according to the complexity of the learning approach:

- The DDC prototype: where the first turn of an utterance is treated as a pattern and the second as a template.
- The KGA prototype: where two learning approaches were adopted. The first word and the most significant word approaches.
- The BNC prototype: where in addition to the first word approach, two significant word approaches were used. The same learning technique was used in the following prototypes.
- The Qur'an prototype: where in addition to using the same learning approaches as the BNC, the system was adapted to deal with the Arabic language and the non-conversational nature of Qur'an.
- The FAQ prototype (FAQchat): where the BNC system was adapted to deal with a structured database in terms of questions (patterns) and answers (templates).

4.3 The DDC prototype

The aim of using the DDC was to explore the problems of using the corpus approach to retrain the chatbot. The first prototype is composed of the four main phases described in 4.2.2: reading, text reprocessing, converting and copying the resulting categories to AIML files. Simple pattern-template categories were learnt, where each utterance or turn in the dialogue was taken as a pattern to match the user input, and the subsequent utterance became the template for the chatbot answer. At this stage, the corpus problems were explored without an attempt at solving them, except removing the linguistic annotations. For example, if the program reads the dialogue transcript from the astronomy transcript in the MICASE corpus, every pair of speakers will generate a new AIML category where the first speaker represents the pattern, and the second speaker represents the template, after applying the filtering process. This is illustrated in figure 4.1.

Sample of MICASE transcript

S1: circumpolar stars. So if I keep my pointer there, [S2: oh] <ROTATES CEILING> everything else moves and we all get sick. <SS LAUGH> and we go backwards in time. And that's even more fun.

S2: make it go really fast.

The AIML category is:

```
<category>
  <pattern>CIRCUMPOLAR STARS SO IF I KEEP MY POINTER
          THERE EVERYTHING ELSE MOVES AND WE ALL GET
          SICK AND WE GO BACKWARDS IN TIME AND THAT'S
          EVEN MORE FUN
  </pattern>
  <template>make it go really really fast.</template>
</category>
```

Figure 4. 1 A sample of MICASE transcript with its corresponding AIML category

4.4 The KGA prototype

The KGA corpus was tested within this prototype. This prototype was the basic step towards solving KGA (Afrikaans) and the subsequent corpora problems. One advantage gained here, is the ability to build a chatbot in a language unknown to the author.

4.4.1 Learning techniques

In this prototype the program has a more general approach to find the best match against user input from the training dialogue. Two learning techniques were adapted, the “first word” approach, and the “most significant word” approach.

In the first word approach, the first word of an utterance was assumed to be a good clue to an appropriate response: if the input can not match against a complete corpus utterance, then at least matching just the first word of a corpus utterance could work. This relates to the Functional Linguistics analysis of sentence/utterance into Topic and Comment, or Subject and Predicate: the Topic or Subject is at the start of the sentence, at least in English. For each atomic pattern, a default version was generated that holds the first word followed by wildcard to match any text, and then associated it with the same atomic template.

Unfortunately this approach still failed to satisfy the trial users; this relates to the fact that the first word is often a function word, not the complete topic/subject. Therefore, the word in the utterance with the highest “information content”, the word that is most specific to this utterance compared to other utterances in the corpus, was used during the matching process. This should be the word that has the lowest frequency in the rest of the corpus. The most significant approach was selected to generate the default categories, because usually in human dialogues the intent of the speaker is hiding in the least-frequent, highest-information word. The least frequent word is used to build four default categories holding the least frequent word in different positions as shown in figure 4.2. Within the AIML format, these four categories are the only way to allow the word to appear in a wildcard position in a sentence.

Assuming the atomic category is:

```
<category>
<pattern>SARFU HET OMTRENT</pattern>
<template>tien kleintjies gekry</template>
</category>
```

The least frequent word for the previous pattern is SARFU, so the default categories are:

First: holding the least frequent word only

```
<category>
<pattern>SARFU </pattern>
<template>tien kleintjies gekry</template>
</category>
```

Second: holding the least frequent word at the start of a pattern

```
<category>
<pattern>SARFU * </pattern>
<template>tien kleintjies gekry</template>
</category>
```

Third: holding the least frequent word at the middle of the pattern

```
<category>
<pattern>* SARFU * </pattern>
<template>tien kleintjies gekry</template>
</category>
```

Fourth: holding the least frequent word at the end of the pattern

```
<category>
<pattern>* SARFU </pattern>
<template>tien kleintjies gekry</template>
</category>
```

Figure 4. 2 An atomic category with its most significant word in 4 positions

4.4.2 Elaborating the system architecture

To cope with the problems in section 3.2 and the learning techniques, the program was composed of the five phases as outlined in figure 4.3.

1. The frequency list phase

The main task of this program is to extract the frequency list from the corpus, and produce a list involving all words with their frequencies.

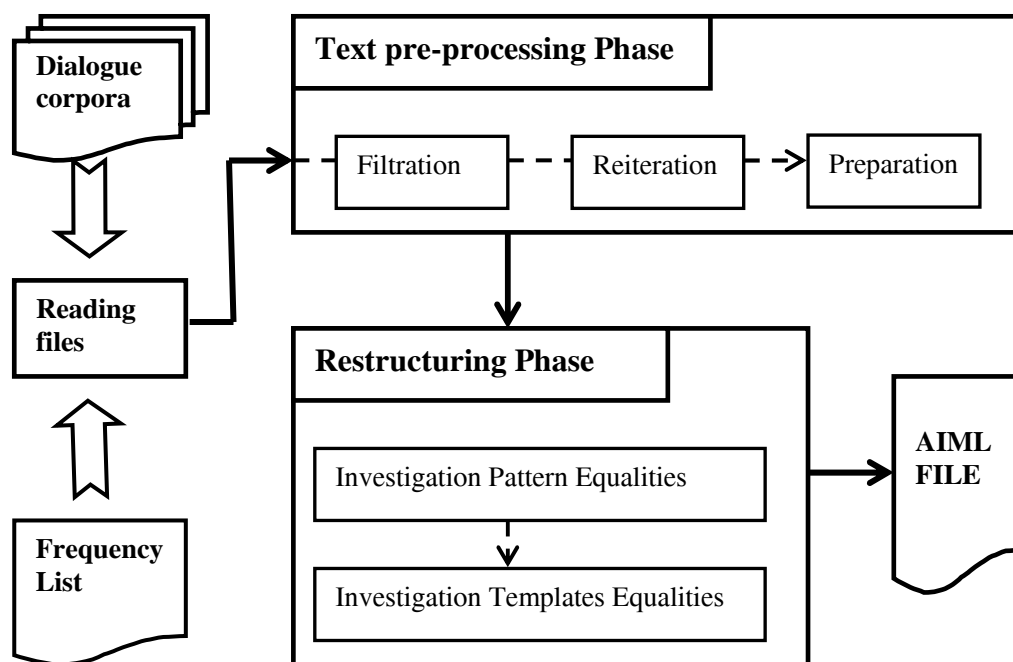


Figure 4. 3 The program algorithm of the KGA prototype

2. Reading phase

The program starts by reading the dialogue corpus and inserts it into a list. Then it completes the frequency list and inserts it in two lists: one for words and the other for frequencies.

3. Text pre-processing phase

Each list element passes through a sequence of modules. First it is filtered where all overlapping and linguistic annotations are removed.

Next the reiteration module duplicates each element to be treated as a pattern in one turn and as a template in the next. After that the preparation module originates pattern and template sequentially, by removing punctuation from the patterns and converting pattern to upper-case. Within the same module a tokenisation process is applied to extract the first word and the least frequent one. The first word is used to build a default category starting with the first word. And after comparing each token with the corpus word-frequency list, the first most significant word (least frequent) is extracted. Four default categories are added which holding the first-most significant word in different position of the pattern: start, middle, last or just the least frequent word alone as described in 4.4.1.

4. The restructuring phase

Given a large training corpus, the same utterance may be said more than once, with different replies; or one reply may be the answer to two or more different utterances. In the case of adding extra default categories to allow the chatbot to answer partial matches, there are many more repeated templates and patterns in our AIML file. The file must be restructured to collate these repetitions. Atomic categories that are generated from the previous process are investigated in two directions. All categories with the same pattern and different templates are grouped to generate one category holding that pattern and new template with a random list of different choices of the original templates. On the other hand, all different patterns with the same templates are used to generate one atomic category of the pattern and template and other categories that use symbolic reduction templates to map each pattern to the atomic one.

5. Copying phase

The AIML list is copied to an AIML file, to retrain ALICE.

Two versions of ALICE were generated using the Afrikaans corpus. The first is called Afrikaana and speaks Afrikaans, and the other speaks bilingual English and Afrikaans and called AVRA. More details about those versions will be discussed in chapter 5.

4.5 The BNC prototype

A new prototype was created to cope with the BNC samples and problems, discussed in 3.3. The key differences for this prototype include:

1. Using the lemmatised frequency list, to extract the least frequent word.
2. Using the BNC spoken transcripts, and modifying the program to handle the BNC annotations and problems.

4.5.1 The BNC frequency list

“A central fact about a word is how common it is. The more common it is, the more important it is to know it.” (Kilgarriff 1997). Kilgarriff believes that foreign learners should be taught the common words first, so they understand them and know how to use them. This idea led him to implement a program to extract words from the BNC corpus and count the frequency of each. Two frequency lists were generated, the lemmatised and unlemmatised list.

The lemmatised frequency list includes 6,318 words with more than 800 occurrences in the whole 100M-word BNC. The frequency of verbal words and its nominal are generated separately, where the count of the verb is the sum of counts of all instances for each verbal, so the frequency of verbal ‘aim’ will count ‘aims’, ‘aiming’, and ‘aimed’.

The unlemmatised list format comprises four fields separated by spaces, starting with the frequency, then the word, the POS, and finally the number of files the word occurs in, as shown in figure 4.4.

6187267	the	at0	4120
2941444	of	prf	4108
2682863	and	cjc	4120
2126369	a	at0	4113
1812609	in	prp	4109

Figure 4. 4 A sample of Kilgarriff unlemmatised frequency list

4.5.2 The alteration method

In this prototype, there is no need to run the frequency list phase since it already exists. The new prototype is composed of the other phases used in the KGA prototype with the following alterations:

1. Reading phase

The program starts by reading the frequency list and copying it into a list named “bnc_freq”. The reading process involves two aspects:

- Extracting the word and its frequency, ignoring the POS and the number of files in which the word occurs.
- Ignoring numbers and any non-orthographic words such as “in-spite-of”.

After that, the BNC files are read and inserted into a list. Because of the huge size of the BNC files, the files are read in sequence. Each file is passed to other phases pre-processing, and restructuring. At the end the atomic and default category of that file is generated and the process is repeated again with another file.

A new problem was encountered in relation to the fact that some BNC spoken tokens were not found in the unlemmatised list. The problem was revealed with utterances which have only one token such as “huhuhuhu”. In such cases the token itself is considered as the least frequent word.

2. Text normalisation phase

The varied annotation formats used to annotate dialogue corpora requires changes in the filtering process, to remove any unnecessary linguistic annotations.

- a. The normalisation module was modified as suggested in section 3.3.2 as follows:
 - Removing the unclear turns.
 - Deeming the overlapping turns as separate turns rather than eliminating it as in the KGA prototype.
 - Replacing the abbreviation of “n’t” with “not”.
 - Replacing the character entity references with appropriate Roman alphabetic characters recognised in AIML, e.g. “é” to “e”.
- b. The phase is continued by reiterating each turn.
- c. Then the preparation process began by considering the first element in the list as a pattern and the second as a template.
- d. All punctuation is removed from the pattern, and then the first word of each pattern is used to create a new default category holding the first word followed by star, which represents the first word approach.
- e. The pattern is tokenised, and the “bnc_freq” list is scanned to extract the frequency for each token in the pattern. The generated list is sorted by frequency in ascending order, and the first token is considered the first-most significant word, and the next token as the second-most significant word.
- f. The process is continued by generating four categories, atomic category holding the first-most significant word only, and another three default categories holding the first-most significant word in the first, middle, and last of the sentence.
- g. Another twelve categories were created, two atomic ones holding the two significant words in different position. The other ten are default categories which hold the two significant words in different orders.

3. The restructuring phase

Then the restructuring phase is executed where the same patterns with different templates are mapped into one pattern with a random list of responses. Different patterns with the same template are categorised as <srai> categories. Finally all categories are written to the AIML default and atomic files.

During the program run, the execution was terminated many times due to an “out of memory storage” problem. This related to the large size of some files; the program tries to store the text in a list prior to conversion. It has been recorded that the size of some files was around 2MB. To solve this problem, the file was split and distributed into several smaller files. Several BNC versions were generated and more than one million categories were created as will be discussed in detail in chapter 5.

4.6 The Qur’an prototype

This prototype was designed to cope with the Arabic language, and the non-conversational nature of the Qur’an. The original idea was to have two versions, one is Arabic only and the other accepts English and gives Arabic and English answers. Because of the large size of the English Qur’an text (956 KB), it was portioned into sub-texts. The Arabic text of the Qur’an was partitioned in the same way to be compatible with the English ones. This prototype is composed of the following:

Creating frequency list

This involves the same phase used to produce the previous frequency lists in the previous prototypes. This program was used twice to generate an Arabic frequency list extracted from the Arabic Qur'an and an English frequency list extracted from the English translation of the Qur'an.

Generating patterns-templates

Four main processes are involved in this phase: reading, concatenation, reiteration and creating patterns-templates.

1. Reading process

The program starts by applying three reading processes:

- a. Reading and inserting an English text in a list named "English_String".
- b. Reading and inserting the corresponding Arabic text in a list named "Arabic_String". While reading the Arabic text, each element is repeated, because the doubled list will be used later with the corresponding doubled English one.
- c. Reading the English frequency list of the Qur'an[, and inserting the words and its frequencies in two lists "least" and "count".]

2. Concatenating process

To facilitate building patterns and templates, each list element was treated as a separate ayya². Lines related to same monologue are merged together as suggested in 3.4.2. During this process the statement "What God said is True" is added to be the last element in order to recognise the end of a soora, as Muslims always say it at the end.

² To recap, the Qur'an is composed of 114 sooras (chapters), where each one consists of several ayyas (verses).

3. Reiteration process

Since the Qur'an is not a series of "turns", during this process each element of the English list except the opening line is repeated to be a pattern in one turn and a template in the next.

4. Finding the most significant word

The list is organised as follows, the first element is the pattern, and the next is its template. Even indices hold the patterns while the odd ones hold the templates. A tokenisation process is activated on each pattern. Then the frequency of each token is extracted from the frequency list. Finally the first-most significant word as well as the second-most significant one are extracted.

5. Creating patterns and templates

For each English pattern and template pair (after the reiteration) do the following:

- a. Extract the pattern from the even index and the template from the next odd one.
- b. Extract the corresponding pattern and template from the Arabic list, which has the same position and format, so element (i) in the English list has its corresponding translation in the Arabic list at index (i).
- c. Find the two most significant words for the English pattern.
- d. Build the atomic category, which has the English pattern after normalisation, and the English/Arabic template.
- e. Build the default categories using the most significant words as patterns and the verses that have these words as a template as done before within the BNC prototype.

In the atomic category the pattern is the ayya, and the template is the next ayya. While in the default categories the pattern is the most significant word(s) connected with the "*" in different position to match any string, and the template is the ayya holding this word(s). During this phase, the

English and Arabic soora numbers are replaced by the corresponding soora names. The Arabic template is appended to the English one. At the end, the generated patterns and templates are copied to a file.

The restructuring

All generated AIML files were merged and copied in a list together. Then as done in the previous prototypes:

1. All categories with the same patterns are mapped to one category with a template composed of a random list.
2. The difference from the previous prototypes is that recursive categories are not used because each pattern has a different template.
3. The list is split into two lists, the first holding the atomic categories, and the second holding the default ones. At the end the atomic and default lists are copied in two files.

The Arabic version which accepts Arabic input and produces Arabic output is generated in the same manner. As a result of partitioning the source text into two sub-texts, four versions of ALICE were generated: two which accept input in Arabic and produced output in Arabic only, the other two accepts English input and reply in both English and Arabic languages.

4.7 The SoC FAQ prototype (FAQchat)

The fifth prototype used the Frequently Asked Questions (FAQ) of the School of Computing (SoC) at University of Leeds described in section 3.5. The previous prototypes processed real human dialogues or monologues. In this prototype, the aim is to explore how to adapt the system to work with a structured database and to evaluate the Java software in terms of precision and recall metrics based on number of hits found, which can not be counted using the previous dialogue or monologue corpora. The program elaboration is described in the following sections.

The Qur'an prototype is used but with a few modifications. The FAQchat prototype is divided as follows:

Phase 1: Creating links file

The FAQ is read to extract all links and put it in a file after eliminating any repeated links.

Phase 2: Generating atomic file

The second phase is generating the atomic file:

1. Extracting questions
 - a. Reading the questions which are denoted by specific tags illustrated in figure 3.13 such as `<DIV CLASS="question">` and `<H1 CLASS="sect1">`.
 - b. Concatenating the question lines until `</DIV>` is encountered.
 - c. Normalising the question by removing punctuation and unnecessary tags.
 - d. Adding the question as a pattern.

2. Extracting the answer
 - a. Reading the answer which is denoted by: `<DIV CLASS="answer">`.
 - b. Checking that the number of the `<DIV CLASS=...>` tag is equal to the number of the `</DIV>` that denote the end of tags. If the number is not equal, the extra tags will be removed. This module was necessary to avoid the incompatible number of begin and end tags.
 - c. Replacing special character entities starting with “&” with normal alphabetic character.
 - d. Extracting the link for each question from the links file and adding it at the end of the template preceded by “For more information look at:”

Phase 3: Generating the frequency list

The frequency list is built from the questions only, since the most significant words will be used within the questions. All questions denoted by <pattern> are read from the atomic file. The frequency is calculated using the same module as in previous prototypes.

Phase 4: Generating the default files

1. Reading the questions and extracting the two most significant words (content words only) by using the same module as in the previous prototypes.
2. Extracting the links that involve the most significant words.
3. Different categories are added to extend the chance of finding answers, where the answer is either a set of links or a direct answer:
 - Build four categories using the first-most significant word in four positions as patterns and the set of links it has as templates.
 - Repeat the same using the second-most significant word.
 - Build four categories using the first word and the first-most significant words where it is handled in four positions.
 - Build two categories using the first-most significant and the second-most significant words, keeping the order of position as in the original question. The answer for this is the links having both words. In the case of having one link, the answer is mapped to the atomic pattern to get a direct answer.
 - Build a category using the first word, the first-most significant word, and the second-most significant word where the template is a direct answer.

Building a new category holding the first word in addition to the most significant ones were added to this prototype to make the pattern-matching process more accurate and generated a more reasonable answer which either correct or close enough to the correct one. At the end a version of ALICE called FAQchat was generated to give answers to questions relating to the School of Computing at University of Leeds.

4.8 Summary

In this chapter, a Java program, which is developed to map the corpus to the AIML format, is described. The basic phases of the program are:

1. Reading phase: store the corpus turns into a list.
2. Text processing phase: to remove all unnecessary annotations, fillers, and overlapping.
3. Pattern-template generating phase: the first turn is considered as a pattern and the following one as a template.
4. Copying phase: copy the last list which holds the patterns and templates to AIML files.

These phases were elaborated to cope with each corpus discussed in chapter 3 to handle the corpus annotations, and to solve the corpus problems. As a result five versions of the program were developed, and at the same time the learning technique in each version was evolved based on users' feedback as follows:

- Generating the atomic categories: using the DDC prototype and generated atomic files only where the first turn in the corpus is a pattern and the second is a template.
- Using first word and the first-most significant word approaches; a frequency list was created from the corpus, and in addition to generate atomic categories, default categories were generated by adding new categories holding the first word of each pattern, and others holding the most significant words. These approaches were used within the KGA prototype.
- Using the first-most significant word, and the second-most word within the BNC and the Qur'an prototypes.
- Using the first-most significant word and the second-most significant one in addition to the first word within the SoC FAQ prototype.

The learning techniques were able to extend the likelihood of finding a match. Samples of dialogues generated with these versions and users' feedbacks are presented in the next chapter.

5. Evaluation

“Say not, ‘I have found the truth,’ but rather, ‘I have found a truth.’ ”

Gibran Khalil Gibran, “The Prophet”

5.1 Introduction

In this chapter, results and evaluation of the chatbot versions developed will be described. All the implemented versions are in fact chatbots which could be evaluated according to naturalness using the same approach used in the Loebner Prize contest as discussed in section 2.8. However, the evaluation is based on meeting the objectives of each version of the chatbot. If the goal is achieved then the implementation was successful. While evolving the different versions, the aim was to improve the matching process in addition to varying the domain and the language as shown in section 1.4.

Four main methodologies were used for evaluation:

1. Evaluate the naturalness of ALICE by comparing human-to-human dialogues versus human-to-chatbot ones.
2. Evaluate the success of the learning techniques in giving answers, based on dialogue efficiency, quality and users’ satisfaction.
3. Evaluate the ability to generate a large number of categories and use of the chatbot as a tool to visualise a corpus.
4. Evaluate the ability of using the chatbot as an information retrieval system by comparing it with a search engine.

The evaluation could not have been done without the help of many experts and users. I would like to thank all of them for their help:

- Paul Rayson, for providing access to the W-matrix tool.
- Gerhard Huysteen, Bertus Van Rooy and their colleagues and students at Potchefstroom University for providing the KGA transcripts, and evaluating the Afrikaana version.
- Derick Burger and his wife who graded the Afrikaana replies.
- Adam Kilgarriff, Sebastian Hoffman and David Lee who helped providing the BNC data.

- Muslims and non-Muslims who evaluated the Qur'an versions.
- The staff and students in the School of Computing at University of Leeds for their help in evaluating the FAQchat.
- Special thanks to Roger Boyle, Katja Markert, and Vania Dimitrova for giving critical feedback.

5.2 Human-to-human versus human-to-chatbot dialogues

In section 2.8 the Loebner Prize was discussed as a method to evaluate chatbots in terms of fooling people that they are chatting with a real human. One conclusion was that 10 minutes chatting may not be sufficient to judge naturalness. Dialogue transcripts generated via chatting with ALICE, the three times Loebner prize winner, are compared with real conversations extracted from different dialogue corpora. The comparison will illustrate the strength or weakness of ALICE as a human simulation, according to linguistic features: lexical, part-of-speech, and semantic differences.

The Wmatrix tool (Rayson 2003) was used for this comparison. Wmatrix computes part-of-speech class and semantic class for each word in the texts, and then highlights specific words, part-of-speech categories, and semantic word-classes which appear more often in one text than the other. The comparison results are viewed as feature frequency lists ordered by log-likelihood ratio: highest LL values indicate the most important differences between corpora. Wmatrix was used to compare between human-to-human dialogues, extracted from CSPA corpus (which is part of the DDC), and human-to-computer dialogues extracted from conversations with ALICE on the AI movie website¹. The ALICE dialogues are general conversations and not restricted to a particular domain such as movies.

The Corpus of Spoken Professional American English (CSPA) includes transcripts of conversation of different types, covering professional activities broadly tied to academia and politics. The transcripts were recorded during professional meetings. The figures 5.1-5.3 below illustrate the most important differences in semantic, POS and lexical levels between ALICE chatbot dialogue and the spoken professional transcript (denoted by 01 and 02 respectively).

¹ <http://aimovie.warnerbros.com>

5.2.1 Semantic comparison

Sorted by log-likelihood value						
Item	O1	%1	O2	%2	LL	
E2+	16	1.42	62	0.15 +	40.90	Liking
Z4	38	3.37	400	0.95 +	40.05	Discourse Bin
Q2.2	37	3.28	449	1.06 +	32.05	Speech acts
Z1	34	3.01	406	0.96 +	30.16	Personal names
P1	1	0.09	671	1.59 -	27.63	Education in general
Z8	214	18.97	5485	12.98 +	26.46	Pronouns etc.
H4	8	0.71	23	0.05 +	24.20	Residence
X2.2+	19	1.68	173	0.41 +	23.85	Knowledge
Z5	268	23.76	13314	31.51 -	22.95	Grammatical bin
O4.2+	6	0.53	14	0.03 +	20.10	Judgement of appearance (pretty etc.)

Figure 5. 1 Screenshot of the semantic comparison between ALICE (01) and CSPA (02)

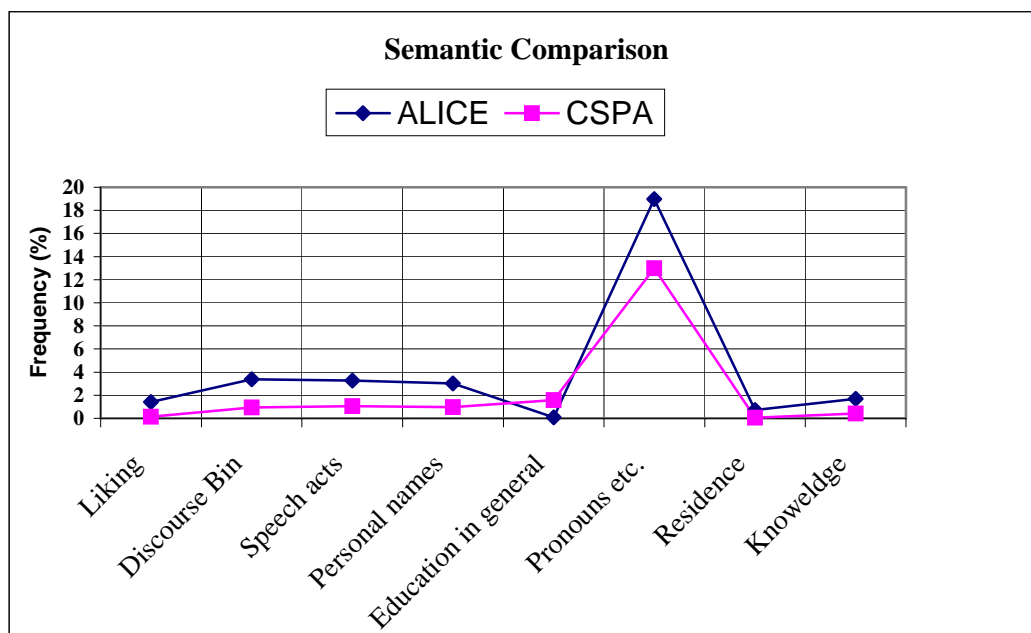


Figure 5. 2 Semantic comparison between ALICE and CSPA

Semantic comparisons in figure 5.1 and figure 5.2 show that the following semantic categories are used more in ALICE transcripts: explicit speech act expressions are highly used within ALICE, an attempt to reinforce the

impression that there is a real dialogue; pronouns (e.g. he, she, it, they) are used more in ALICE, to pretend personal knowledge and contact; discourse verbs (e.g. I think, you know, I agree) occurred more frequently in ALICE, to simulate human trust and opinions during the chat; liking expressions (e.g. love, like, enjoy) appeared more often in ALICE, to give an impression of human feelings. The only categories used noticeably more in CSPA are: education terms, hardly surprising given the academic discourse source; and grammatical function words, corresponding to more complex grammar.

5.2.2 Part-of-Speech comparison

Figure 5.3 and figure 5.4 show the part-of-speech frequency differences between the two sources. Singular first-person pronoun (e.g. I), second-person pronoun (e.g. you) and proper names (e.g. Alice) are used more in ALICE, to mark participant roles more explicitly and hence reinforce the illusion that the conversation really has two participants. Plural personal pronouns (e.g. we) were used more in CSPA, because all samples were extracted from meetings between cooperating professionals, using inclusive language. Coordinating conjunctions (e.g. and, or) and subordinating conjunctions (e.g. if, because, unless) are used more within the CSPA, these indicate more complex clause and phrase structure which ALICE avoids because it applies simple pattern matching techniques, and cannot handle dependencies between clauses. It also makes less use of interjections, preferring more formal clause structure; another interpretation of this imbalance could be that ALICE makes more use of interjections as fillers when no good match is found in the pattern database.

Sorted by log-likelihood value					
Item	O1	%1	O2	%2	LL
PPY	80	7.09	503	1.19 +	144.18
VDO	43	3.81	258	0.61 +	80.57
PPIS2	1	0.09	799	1.89 -	34.03
PPIO1	10	0.89	38	0.09 +	25.87
CC	10	0.89	1343	3.18 -	25.68
PPIS1	55	4.88	984	2.33 +	23.02
NP1	44	3.90	744	1.76 +	20.97
NNB	5	0.44	8	0.02 +	19.59
DD1	9	0.80	1030	2.44 -	16.56
CST	6	0.53	813	1.92 -	15.68
UH	14	1.24	181	0.43 +	11.01

Figure 5. 3 Screenshot of POS comparison between ALICE (01) and CSPA (02)

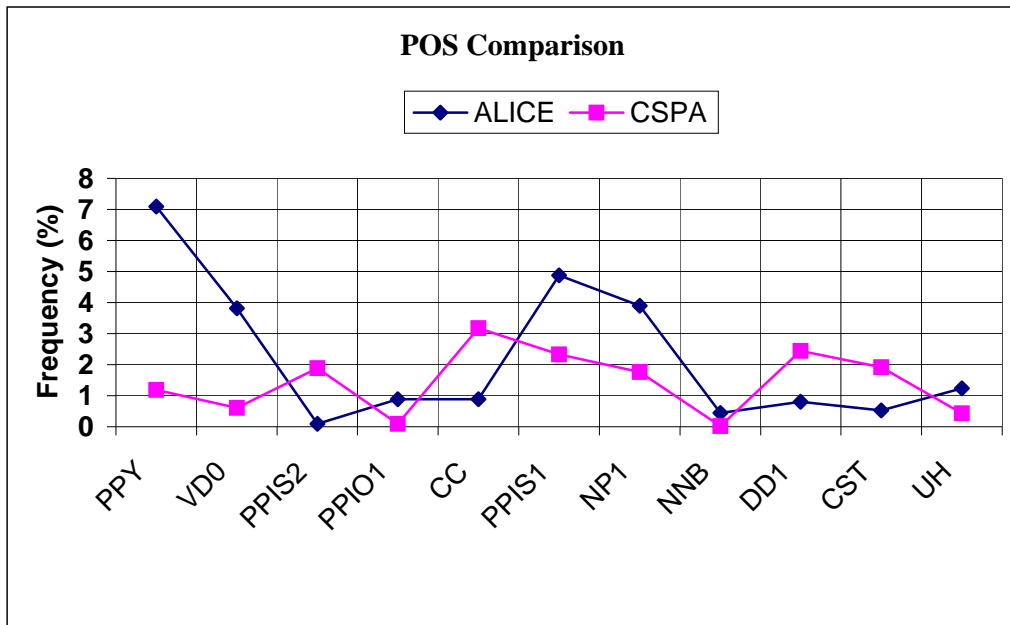


Figure 5. 4 POS comparison between ALICE and CSPA

5.2.3 Lexical comparison

Sorted by log-likelihood value					
Item	O1	%1	O2	%2	LL
you	72	6.38	496	1.17 +	119.80
Emily	9	0.80	0	0.00 +	65.69
do	44	3.90	370	0.88 +	60.25
you_know	8	0.71	7	0.02 +	38.04
Alice	5	0.44	0	0.00 +	36.50
created	5	0.44	0	0.00 +	36.50
internet	5	0.44	0	0.00 +	36.50
name	6	0.53	2	0.00 +	34.90
we	1	0.09	799	1.89 -	34.03
Dr	4	0.35	0	0.00 +	29.20
chocolate	4	0.35	0	0.00 +	29.20
dance	4	0.35	0	0.00 +	29.20
french	4	0.35	0	0.00 +	29.20
ok	4	0.35	0	0.00 +	29.20
am	6	0.53	5	0.01 +	28.90

Figure 5. 5 Screenshot of word comparison between ALICE (01) and CSPA (02)

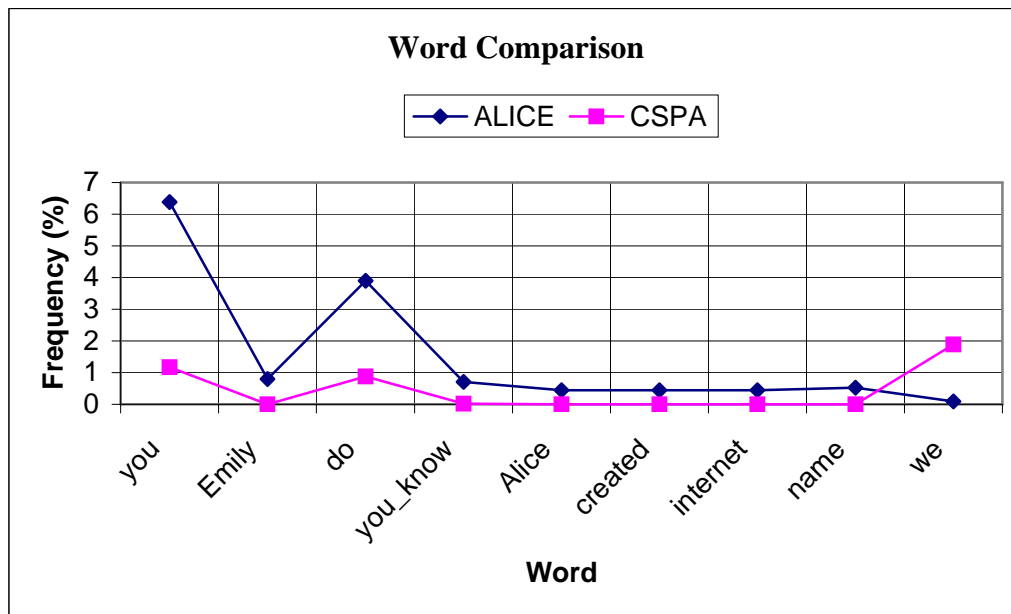


Figure 5. 6 Word comparison between ALICE and CSPA

Word-level analysis results shown in figure 5.5 and figure 5.6 confirm and exemplify the more general part-of-speech and semantic class preferences. ALICE transcripts made more use of specific proper names “Alice” (not surprisingly!) and “Emily”; and of “you-know”, where the underscore artificially creates a new single word from two real words. ALICE also made more use of lexical items which correspond to the “marked” POS and semantic categories above; for example, ALICE transcripts included more use of “I” and “you”.

The above comparison shows that when ALICE tries to simulate real dialogue, ALICE over exaggerates use of key lexical, grammatical and semantic features of dialogue. Other human dialogue corpus texts were compared against ALICE transcripts; there are genre- or topic-specific differences for each corpus, but ALICE’s over-exaggerated use of speech act verbs, first-person pronouns, and similar explicit dialogue cues are a recurring result.

5.3 Evaluating the Afrikaans version

Instead of chatting for just 10 minutes as suggested by the Loebner Prize, alternative evaluation methods more attuned to and appropriate to practical information systems applications are advocated. Methods to train and adapt a chatbot to a specific user’s language use or application are investigated via a

user-supplied training corpus. The evaluation takes account of open-ended trials by real users, rather than artificial 10-minute trials. One example is a chatbot for Afrikaans-speaking researchers and students in South Africa.

In section 4.2 we described the adaptation of ALICE/AIML chatbot architecture to be retrained from a dialogue corpus to generate a new version of ALICE in a different language style (e.g. Professional American English), or even a completely different language. A training corpus of Afrikaans dialogue transcripts, the Korpus Gesproke Afrikaans (KGA) as discussed in section 3.2, was used to develop Afrikaans-speaking and bilingual Afrikaans-English chatbots. Literal matching, the first word approach, and the most significant word approach were applied as clarified in section 4.4.2. Two versions of ALICE for the Afrikaans language were developed, “Afrikaana” that speaks only Afrikaans and “AVRA” that speaks English and Afrikaans. AVRA combines the categories from the KGA with the original English ALICE categories. This was inspired by our observation that the KGA actually includes some English, as Afrikaans speakers are generally bilingual and “code-switch” comfortably. Prototypes of the chatbots on websites were mounted, using Pandorabot² service, and open-ended testing was encouraged and feedback was provided from five remote users at Potchefstroom University in South Africa; this helped in refining the system more effectively.

To evaluate this prototype, three evaluation metrics were adopted:

1. Dialogue efficiency in terms of matching type.
2. Dialogue quality metrics based on response type.
3. Users’ satisfaction assessment based on an open-ended request for feedback.

5.3.1 Dialogue efficiency metrics

The efficiency of four sample dialogues was measured in terms of atomic match, first word match, most significant match, and no match. The aim was to measure the efficiency of the adopted learning mechanisms to see if they increase the ability to find answers to general user input. Table 5.1 shows the frequency of each type in each dialogue generated between the user and the Afrikaans chatbot; in figure 5.7, these absolute frequencies are normalised to relative probabilities.

² <http://www.pandorabots.com/botmaster/en/home>

Matching Type	Dialogue 1	Dialogue 2	Dialogue 3	Dialogue 4
Atomic	1	3	6	3
First word	9	15	23	4
Most significant	13	2	19	9
No match	0	1	3	1
Number of turns	23	21	51	17

Table 5. 1 Response type frequencies for “Afrikaana”

This approach of evaluation via dialogue efficiency metrics, illustrates that the first word and the most significant approach increase the ability to generate answers to users and let the conversation continue.

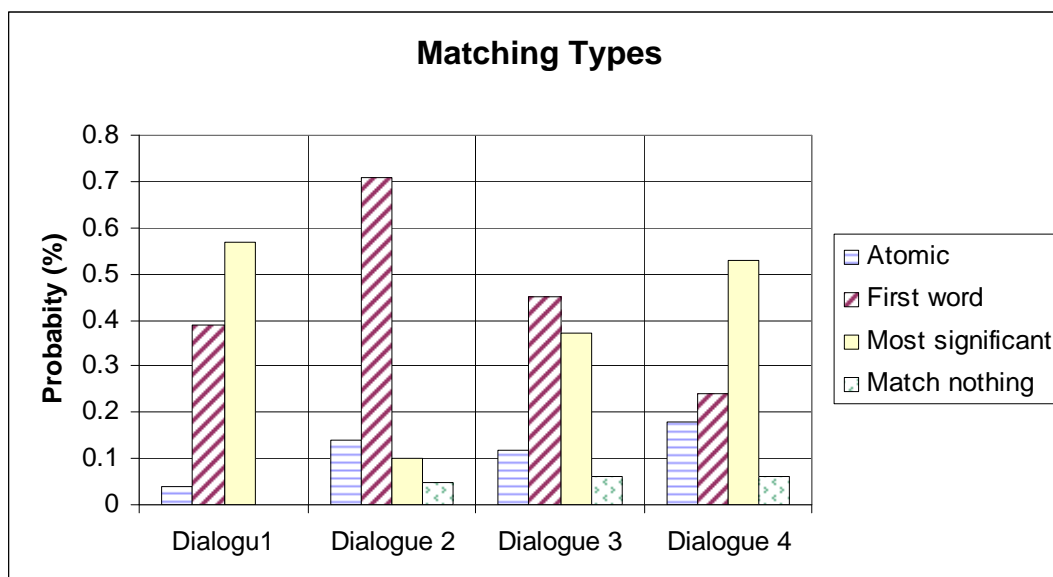


Figure 5. 7 Dialogue efficiency: “Afrikaana” matching types relative probabilities

5.3.2 Dialogue quality metrics

In order to measure the quality of each response, responses were classified according to an independent human evaluation of “reasonableness”: reasonable reply, weird but understandable, or nonsensical reply. The chatting transcript was given to an Afrikaans-speaking teacher to mark each response according to these classes. Table 5.2 shows the number of turns in each dialogue and the frequencies of each response type. Figure 5.8 shows the frequencies normalised to relative probabilities of each of the three categories for each sample dialogue.

For this evaluator, it seems that “nonsensical” responses are more likely than reasonable or understandable but weird answers.

Response Type	Dialogue 1	Dialogue 2	Dialogue 3	Dialogue 4
Number of turns	23	21	51	17
Reasonable	2	4	5	5
Weird	19	3	7	1
Nonsensical	2	14	39	11

Table 5. 2 Response type frequencies: subjective analysis of “Afrikaana” responses

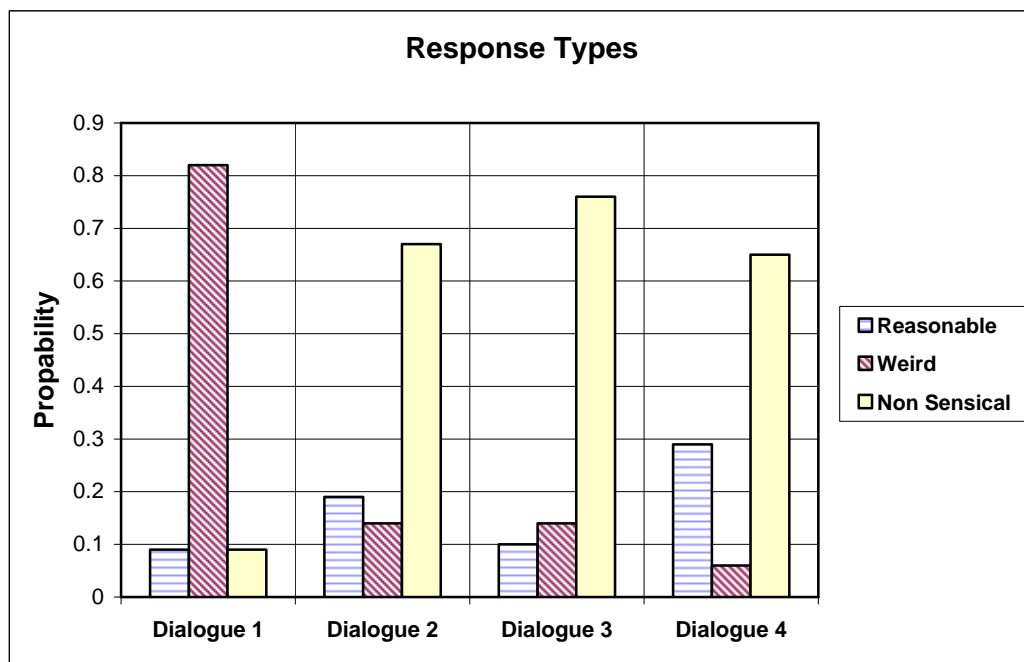


Figure 5. 8 The quality of the dialogue: “Afrikaana” response type relative probabilities

5.3.3 User satisfaction

Users' satisfaction was gathered from users' comments after chatting with the systems. The first prototypes were based only on literal pattern matching against corpus utterances: the first word, and the most significant word were not implemented, so "wildcard" default categories were not added. The Afrikaans-speaking evaluators found these first prototypes disappointing and frustrating: few of their attempts at conversation found exact matches in the training corpus, therefore Afrikaana replied with a default "ja" most of the time. However, expanding the AIML pattern matching using the first-word and the most-significant-word approaches yielded more favourable feedback; the informants found the conversations less repetitive and more interesting. In this respect, user satisfaction had been achieved based on this kind of informal user feedback.

5.3.4 The methodology drawbacks

Unfortunately, this evaluation is restricted to the available resources which are limited. Firstly, the chatting process was done by a few people in South Africa, colleagues and students of the Afrikaans linguists, who sent the corpus. Secondly to label the generated dialogues in terms of the quality metrics, one Afrikaans teacher was found to do that.

However, the reasons behind why the majority of responses were nonsensical can be related to three issues:

1. The dialogue corpus context does not cover a wide range of domains; Afrikaana can only "talk about" the domain of the training corpus.
2. The repeated approach used to solve the problem of determining the pattern and the template in case of more than two speakers may lead to incoherent transcripts.
3. The adopted machine-learned models have not included linguistic analysis mark-up, such as grammatical, semantic or dialogue-act annotations (Atwell 1996b, Atwell et al. 2000a), as ALICE/AIML makes no use of such linguistic knowledge in generating conversation responses.

5.4 Evaluating the BNC versions

Evaluation of the BNC version is based on achieving the following objectives:

1. Measuring the ability of the program to generate large number of categories was the main goal for using the BNC corpus.
2. Measuring the ability of using the chatbot as a tool to visualise such a large corpus.

5.4.1 Results and discussion.

After nearly ten days of running the program, 1,153,129 categories were generated and separated in 801 files and 47 folders, which took up 320 MB of disk space. This number is 20 times bigger than any existing chatbot: the large public-domain AIML category-file developed over several years contains about 50,000 categories (Wallace 2003).

A new version of ALICE using the Pandorabot web-hosting service was created to make our chatbot available for use over the World Wide Web. Instead of launching 801 files, a Java code was written to map the files of each folder into one file. The mapping process squeezed the files to 47 files, which were used to launch the Pandorabot. Unfortunately, the Pandorabot server could not handle such large files, so the previous 801 files were tried with the Pandorabot. After the successful compilation, and a little chatting, a message was received from the Pandorabot that the big size of our dialogue model was causing major problems for the server and to other users' chatbots; as a result, the BNC version was taken offline again.

Instead, the BNC files were copied into our PC, and a version of the ALICE program was run locally. Three users tried chatting with this version. Different chatting examples were generated, figure 5.9 is a sample of dialogue generated between a user and the chatbot, which is used to analyse the reasons behind users' disappointment.

User: Hi
Chatbot: good morning. Now young lady what can I do for you today?

User: Let's talk.
Chatbot: Yeah

User: Do you like dancing?
Chatbot: No. I ve. I ve just. All I ve done is because Erm because that is a function of that.

User: Tell me about dancing:
Chatbot: The dances. Oh they were just great fun the dances all met in tch school. It was the nearest to us and the next one was the Glen Ayloch school. But oh we d we d a lot of fiddlers in those days and a girl at the Piano. And then when things got a wee bitty better we used to have the Camerons up from Kilray from Kerry to dance to the. Play at the dances. And of course the night of the. Gathering Ball it was just packed tight with people hardly, the school would hardly hold them all. And then they put up a big tent in the. Beside the school where you went out for supper er it was that is s.

User: You had a dancing party at your school?
Chatbot: That is good.

User: I like swimming
Chatbot: Mhm.

User: Do you like swimming?
Chatbot: No. I ve. I ve just. All I ve done is because Erm because that is a function of that.

User: Talk to you later.
Chatbot: Oh. It is still fine.

User: Bye bye
Chatbot: Right cheerio now.

Figure 5.9 A sample of chatting with the BNC chatbot

A careful look at this example dialogue provides us with an insight into some remaining problems with the pattern matching process. Some responses seem meaningless, because the response is taken from a specific BNC text and this chat is not in the same context (e.g. “you had a dancing party at your school?”). Other responses show that the pattern-extraction process needs to be improved. For example, when the user input starts with “do you like...”, the same response will be generated regardless of the object. This is because it matches the first word approach before the most significant one.

To solve these problems:

1. The POS tags in the BNC were used to benefit from the corpus linguistic annotation: the first word method was improved by considering the first word scanned in the pattern which is not a question, a pronoun, or a preposition. The first “meaningful” word in the utterance will be considered; also the most significant “meaningful” word will be considered as a least frequent word. However, this experiment contradicts the generalisation approach, but it was necessary to check that for future work, the annotated corpus could be used to obtain the POS-tags to have more reasonable answers.
2. To concentrate on a certain context, specific domains from the BNC were extracted and used to develop different versions of the chatbot. Seven versions were generated which emulate London teenagers and “Loudmouths” as shown in table 5.3; each contains about half the number of categories in the public-domain ALICE “brain”. These versions were tested by 8 users.

Another problem encountered in the previous sample of chatting, is the incomplete utterances. Actually this resulted from the unclear or overlapping parts and the approach in removing the overlapping.

Teenager version (BNC files)	Atomic		Default		Total
	Categories	File size (KB)	Categories	File size (KB)	Categories
Michael	7,021	1,044	14,914	1,905	21,935
Jessica	4,261	787	10,069	1,554	14,330
Peter	5,912	939	12,878	1,771	18,790
Robin	4,726	673	14,580	1,774	19,306
Samantha	6,058	788	18,283	1,695	24,341
Loudmouth1	8,114	1,301	17,827	2,368	25,941
Loudmouth2	8,383	1,217	16,971	2,130	25,354

Table 5. 3 The BNC London teenager and loudmouth chatbots

The Bergen Corpus of London Teenager English language (COLT) is an English corpus involving speech of 13-17 years old teenage students (Andersen and Stenström 1996). Some COLT files were added to the “K” folder of the BNC. A list of the first line of COLT files were provided by Sebastian Hoffmann from Switzerland. The first line of each was used to search the BNC “K” folder to extract the BNC file names. The BNC files KPL, KPS, KPT, KPW, KPX, and KPY were used to generate Michael; KPG was used for Jessica; KP4 and KP6

were used to generate Peter; KPX and KPY for Robin; and finally KPE to generate Samantha. The files were selected taking into consideration the size to avoid causing a problem with the Pandorabot server. The loudmouth versions were extracted from files having swear words, which were provided by Hoffman. KDN was used to build Loudmouth1; KD9, KE1, and KR2 for Loudmouth2. Figure 5.10 illustrates the dialogue between a user and Peter.

Human: I never give cigarettes
Peter: Did you give that one back to Dan and Honey?

Human: You would not give me a cigarette anyway
Peter: In any case, Who else would not give you a cigarette?

Human: It depends
Peter: Yeah

Human: Did your mum know that you smoke cigarettes
Peter: About a month ago

Human: Alright look
Peter: Stop whispering about me

Human: Yeah well I am so used to you talking about me
Peter: I, I Well that's okay. Everything is changing so quickly these days.

Figure 5. 10 A sample of chatting with Peter-BNC chatbot

Another five versions were generated in different domains as shown in table 5.4 and 3 users tried it. David Lee's Excel spreadsheet³ was used to extract the BNC file names for this purpose. Figure 5.11 shows chatting with the sport version to visualise what topics are there.

Chatbot	Atomic		Default		Total
	Categories	File size (KB)	Categories	File size (KB)	Categories
Sport	3,283	974	7,913	1,820	11,196
World affairs	3,120	983	8,756	1,886	11,876
Travel	640	314	1,636	575	22,76
Media	1,061	491	3,126	1,210	4,187
Food	503	93	1,125	168	1,628

Table 5. 4 Different BNC versions

³ <http://clix.to/davidlee00>

Human: Hello there!

Sport: OH!

Human: do you intend to watch a football match?

Sport: Days gone by?Well, you had to send miles for them, did not you er er blacksmith Blacksmiths were the thing on the side of the road, did not he do all the wheeling, mend your wheels, and all horse, all iron wheels, and things like that, wo not years ago, and you had to send for him to come and do 'em, w well, he had men that he'd, that he'd taught like boys, apprentices He, they used to go out and do all the outside work for him

Human: I'm talkin about football!

Sport: Until he'd before he took me to hospital!

Human: I'm talkin about football!!

sport: I think football seems to, to bring out the worst in a lot of people and that's the part of it that I do not like

Human: do you like football?

Sport: Yep

Human: I watch it regularly.

Sport: Because if football's on it has to be watched!It's like a a religion! Fi

Human: what about you?

Sport: OH!

Human: do you always watch it?

Sport: No, but when you switch on the telly on a Saturday afternoon there's football, football, football!Especially Sunday nights too !

Human: Admit, how many hours would you like to watch it if available?

Sport: If played properly, it's poetry in motion!And as the lady down here said, if the ladies do not like it, switch it off!If I had the chance I'd watch it twenty four hours a day!

Human: is there any type of sport you practice?

Sport: So, I played this and I taught myself and I had, I played at England and all over for oth , a ladies team, played a on a man's team and er, the we , the west and that, But, the men did not play you rough or anything but we had some good games!

Human: what about basketball?

Sport: Erm

Figure 5. 11 A sample of chatting with the sport-BNC chatbot

The classical way of viewing a data set is using the visualisation process, which maps the data from numerical or textual form to a visual representation that our mind can easily interpret such as: using graphical diagrams, charts, and geometric representation. However, it is difficult to visualise a corpus using these techniques. Instead of this, frequency lists and concordances could be the typical methods which help to visualise corpus content.

Another method is using the chatbot. The visualisation could be achieved in terms of:

1. Giving an overview of a whole data set, and this is depending on the chatting time.
2. Focusing on a particular topic, to see what a corpus has in this topic by asking a direct question as illustrated in user's question in figure 5.11: "What about basketball?"

Unfortunately, this approach is not complete in this respect; one may argue that, sometimes the topic of text is about cars, for example, even though the word "car" itself is not using within the text, but denoted by a "motor". Humans can infer that motor refers to car, while the current chatbot can not. Another criticism is that the "find" tool used in any word processor could check if a certain word like "basketball" is in the corpus or not.

This is totally true, no attempt is made here to compare the chatbot with other visualisation alternatives, but on the other hand, users could use the generated responses to go on a long conversation. By this they could have an overview not particularly if this domain is covered or not, but also could have an insight about the language used. The aim of this experiment is to show that the chatbot could be used as a tool to visualise corpora if the method improved to include topics, synonyms, and other linguistic and semantic features.

5.5 Evaluating the Qur'an versions

The evaluation of the Arabic/English Qur'an version is based on achieving the following goals:

1. Exploring problems in using Arabic language.
2. Exploring the problems of adapting the program to cope with a text which is not conversational in its nature.
3. Exploring the ability to use the chatbot as a tool to access an information source, and a useful application for this.

As illustrated in section 4.6, four versions of the Qur'an were developed:

1. Two Arabic versions where the input and output is in Arabic language.
2. Two bilingual English/Arabic versions where the input is in English and the output is in Arabic and English languages.

5.5.1 Results and discussion of the Arabic versions

The Arabic version program was tested on the sooras from 58-114, where these sooras are short verses, and are recited by Muslims. 76,961 categories were created and tested by Arabic Speaking Muslims.

One potentially interesting achievement is: if an ayya is repeated in more than one soora, the following ayya is presented from each soora in addition to the soora name that has this ayya; this may be useful for statistical analysis by Qur'an scholars and others. Feedback from Arabic evaluators presented the following difficulties of the Arabic version:

1. Difficulty in entering the voweled Arabic words using Arabic keyboards. For example, in order to enter a word like (ALzaytoon), after every letter we type we have to press SHIFT key and the key of the vowel, which causes a lot of mistakes. Sometimes the consonant is doubled so users can miss the sign of the doubled consonants (Shadda).
2. Another problem is that there is only one verse returned to the user, when they hoped to see all relevant verses.
3. Not all words have an answer; this is related to the most significant word technique we used, so just two words from each ayyas generated the categories.

To improve the system, the restructuring process was modified. Instead of viewing one verse randomly from the matched list, the whole list was returned. This modification shows an improvement in users' satisfaction. To solve the main problem represented by voweled letters, the future work must focus on removing the vowels from the generated patterns effectively creating an unvoweled Qur'an. Figure 5.12 shows a chatting dialogue with the Qur'an, where the human enters voweled Arabic, and the chatbot replies by finding the ayya(s) matching the human input.

Human:	أَحَدٌ
Qur'an28-30:	1 قُلْ هُوَ اللَّهُ أَحَدٌ (ة روسصالخ إ ل ا) وَلَا يُوْتِئُ وَثَاقَهُ أَحَدٌ (ة روس رج فل ا) 26
Human:	قُلْ هُوَ اللَّهُ أَحَدٌ
Qur'an28-30:	2 اللَّهُ الصَّمَدُ (ة روسصالخ إ ل ا)
Human:	أَنْتُمْ غَائِبُونَ
Qur'an28-30:	3 وَلَا أَنْتُمْ غَائِبُونَ مَا أُعْبِدُ (ة روس نورفاك ل ا) وَلَا أَنْتُمْ غَائِبُونَ مَا أُعْبِدُ (ة روس نورفاك ل ا) 5

Figure 5. 12 A sample of chatting with the Arabic Qur'an chatbot

5.5.2 Results and discussion of the English/Arabic version

Two versions of ALICE were published using the Pandorabot service, the first handles sooras from 1 to 30, and the second handles sooras from 31-114 as shown in table 5.5. This splitting is according to the size, sooras from 1-30 have long verses, and the size of the file is big. Unfortunately, the full Qur'an version general could not be installed to a local PC, because the Arabic language was not recognised.

AIML Files	Sooras 1-30			Sooras 31-114		
	No. of files	Files size (MB)	Categories	No. of files	Files size (MB)	Categories
Atomic	5	7.95	12,893	3	3.56	10,057
Default	19	24.3	41,805	10	19.9	32,544
Total	24	32.3	54,698	13	23.26	67,771

Table 5. 5 Characteristics of the English/Arabic Qur'an chatbots

Human: Is it right that our protector is Allah?
Qur'an14-114:
 {Muhammad (Muhammad)} 11 That is because Allah is the Protector of those who believe, but those who reject Allah have no protector.

1. ذَلِكَ بِأَنَّ اللَّهَ مَوْلَى الَّذِينَ آمَنُوا وَأَنَّ الْكَافِرِينَ لَا مَوْلَى لَهُمْ (ة روس دمحم) 11

{The Morning Star (At-Tariq)} 4 There is no soul but has a protector over it.
 4 إن كُلُّ نَفْسٍ لَّمَّا عَلَيْهَا حَافِظٌ (ة روس قراطلا) 4

Human: What is the resurrection day?
Qur'an14-114:
 {The Letter Qaf (Qaf)} 11 As sustenance for (Allah's) Servants;- and We give (new) life therewith to land that is dead: Thus will be the Resurrection.
 11 رِزْقًا لِلْعِبَادِ وَأُوْحِيْنَ بِهِ بَلْدَةً مَّيْتًا كَذَلِكَ الْخُرُوجُ (ة روس ق) 11

{Nooh (Nooh)} 18 "And in the End He will return you into the (earth), and raise you forth(again at the Resurrection)?
 18 ثُمَّ يُعِيدُكُمْ فِيهَا وَيُخْرِجُكُمْ إِخْرَاجًا (ة روس حون) 18

{The Resurrection (Al-Qiyama)} 1 I do call to witness the Resurrection Day;
 1 لَّا أَقْسِمُ بِبَيْتِ الْمَقَامَةِ (ة روس عم اي قلا) 1

Figure 5. 13 A sample of chatting with the English/Arabic Qur'an chatbot

The user inputs a turn in the conversation, such as a question or a statement; the system responds with one or more verses from the Qur'an, which seems appropriate as shown in figure 5.13. As this is a chatbot rather than a search engine, the ayyaas found are not simply the result of keyword-lookups; the response-generation mechanism is in fact hidden from the user, who will sometimes get the response "I have no answer for that". On the other hand figure 5.14 shows the case of matching a whole ayya, where the robot answer will be the next one.

For English users, there is a problem of transliteration ambiguity. This is when a word in Arabic could be transliterated in multiple ways, for example, Koran-Qur'an, Mohammad-Mohammed. This could be solved in future by adding a manual transliterated list, where alternatives of the same word are bundled together.

Human: Say: He is Allah, the One and Only
Qur'an14-114: {Sincerity (Al-Iklas)} 2 Allah, the Eternal, Absolute;
 اللَّهُ الصَّمَدُ (صالح إلالة روس) 2

Human: Allah, the Eternal, Absolute
Qur'an14-114: {Sincerity (Al-Iklas)} 3 He begetteth not, nor is He begotten;
 لَمْ يَلِدْ وَلَمْ يُولَدْ (ةروس صالح إلالة) 3

Human: He begetteth not, nor is He begotten
Qur'an14-114: {Sincerity (Al-Iklas)} 4 And there is none like unto Him.
 وَلَمْ يَكُنْ لَهُ كُفُوًا أَحَدٌ (ةروس صالح إلالة) 4

Human: And there is none like unto Him.
Qur'an14-114: What the Greatest God Said is True
 قدص اللة ميظعلالة

Figure 5. 14 A sample of matching an entire ayya via chatting with English/Arabic chatbot

In fact evaluation of this kind of general information access is not easy. As the information accessed is not in terms of specific questions, we cannot count numbers of “hits” in order to compute precision and recall scores. The best we could hope for is some kind of qualitative satisfaction assessment. When you come away from a conference, you may well think it was useful, not because you found answers to a list of specific questions, but because you have a better feel for current thinking and ideas in your “sources”, the people you chatted to.

Muslim users were asked to chat with the Qur’an version, and to answer the following questions:

1. Do you feel you have learnt anything about the teachings of the Qur’an?
2. Do you think this might be a useful information access mechanism? If so, who for, what kinds of users?

Some users found the tool unsatisfactory since it does not provide answers to the questions. However, using chatting to access an information system can give the user an overview of the Qur’an contents. It is not necessary that the user will have a correct answer for his request, but at least it motivates him to engage in a long conversation based on using some of the outputs to know more about the Qur’an.

Other users found it interesting and useful in the case a verse being read and the user wanting to find out from which soora it came from. This would also benefit those who want to know more about the religion to learn what the Qur'an says in regards to certain circumstances, etc.

All Muslims are taught to recite some of the Qur'an during school because reading the Qur'an is one of the important basic elements of praying. However, students usually get bored of the traditional ways teachers followed in teaching the Qur'an such as repeating after the teacher or reading from the holy book. Since most students like playing with computers, and chatting with friends, this tool may encourage them to recite a certain soora by entering an ayya each time as illustrated in figure 5.12 and figure 5.14. Since it is text communication, students must enter the ayya to get the next one, and this will improve their written Arabic.

5.5.3 Evaluation methodology and results

To measure the quality of the answers of the Qur'an chatbot version, the following approach was applied:

1. Random sentences from Islamic sites were selected and used as inputs of the English/Arabic version of the Qur'an.
2. The resulting transcripts which have 67 turns were given to 5 Muslims and 6 non-Muslims students, who were asked to label each turn in terms of:
 - Related (R), in case the answer was correct and in the same topic as the input.
 - Partially related (PR), in case the answer was not correct, but in the same topic.
 - Not related (NR), in case the answer was not correct and in a different topic.

Proportions of each label and each class of users (Muslims and non-Muslims) were calculated as the total number over number of users times number of turns. 4 out of the 67 turns returned no answers, therefore actually 63 turns were used. Table 5.6 and figure 5.15 represent the results.

Users	Mean of users classified answers as			Proportion of classifying answers as		
	R	PR	NR	R	PR	NR
Muslims	9.4	17.00	36.6	15%	27%	58%
Non-Muslims	11.34	20.00	31.67	18%	32%	50%
Overall	10.45	18.64	33.91	17%	30%	54%

Table 5. 6 Proportions of answers generated from the Qur’an

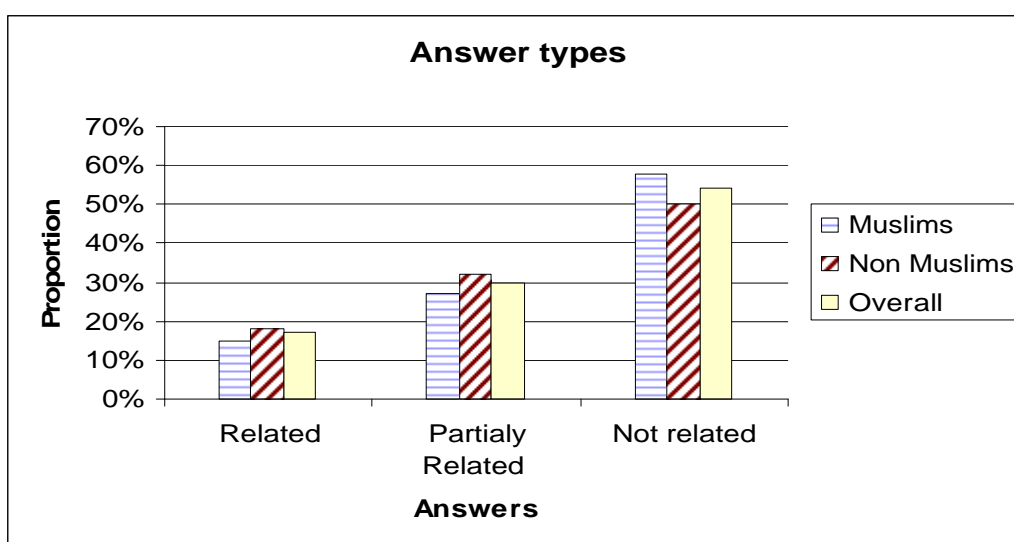


Figure 5. 15 The Qur’an proportion of each answer type denoted by users

In the transcripts used, more than half of the results were not related to their inputs. A small difference can be noticed between Muslims and non-Muslims proportions. Approximately one half of answers in the sample were not related from non-Muslims’ point of view, whereas this figure is 58% from the Muslims’ perspective. Explanation for this includes:

- The different interpretation of the answers. The Qur’an uses traditional Arabic language, which is sometimes difficult to understand without knowing the meaning of some words, and the historical story behind each verse.
- The English translation of the Qur’an is not enough to judge if the verse is related or not, especially given that non-Muslims do not have the background knowledge of the Qur’an.

The main goal of chatting with the Qur'an was not to measure the quality of answers but to explore the problems of using the Arabic language to retrain ALICE and the difficulties found by users while chatting with the Arabic versions.

In this context, many ideas could be applied to improve the Qur'an versions:

- Apply an algorithm to remove vowels from the source text, so it could be matched with the unvoiced user input.
- Attach each verse to its explanation as reported in a well known commentary of the Qur'an.
- Add the phonetic transliteration of each Arabic verse in addition to the textual one to teach the correct pronunciation.

Using chatting to access the Qur'an looks like the use of a standard Qur'an search tool. In fact it is totally different; a searching tool usually matches words not statements. For example, if the input is: "How shall I pray?" using chatting: the robot will give you all ayyas where the word "pray" is found because it is the most significant word. However, using a search tool⁴ will not give you any match. If the input was just the word "pray", using chatting will give you the same answer as the previous, and the searching tool will provide all ayyas that have "pray" as a string or substring, so words such as: "praying, prayed, etc." will match.

Another important difference is that in the search tool there is a link between any word and the document it is in, but in the chatting system there is a link just for the most significant words, so if it happened that the input statement involves a significant word(s), a match will be found, otherwise the chatbot answer will be: "I have no answer for that". Comparing a chatbot with a search tool is explained in the next section.

⁴ <http://www.islamicity.com/QuranSearch/>

5.6 Evaluating the FAQ chatbot

Section 4.2 holds the description of the system prototype which was adapted to cope with a frequently asked questions (FAQ) database. The FAQ in the School of Computing (SoC) at the University of Leeds⁵ has been used to retrain the ALICE chatbot system, producing FAQchat. The results returned from FAQchat are similar to ones generated by search engines such as Google, where the outcomes are links to exact or nearest match web pages.

A search engine is “a program that searches documents for specific keywords and returns a list of the documents where the keywords were found.” (Internet.com 2004). However, FAQchat could also give direct answers and the algorithm underlying each tool is different. To evaluate FAQchat, a comparison was made between the FAQchat and Google. The main objective of this experiment is to demonstrate that:

1. FAQchat is a viable alternative to Google.
2. The chatbot could be used as a tool to access FAQ databases.

Currently there are two methods to ask a question related to the SoC:

1. Using the search input box found in the main webpage of Leeds University, which is in fact the Google search engine.
2. Using the SoC FAQ website.

The SoC FAQ website can currently be accessed in two ways:

1. Finding a topic of interest, by browsing the table of contents.
2. Looking for a keyword, by browsing the index.

If the FAQchat succeeds in giving correct answers, a third way could be added “3. Asking a chatbot”.

⁵ <http://www.comp.leeds.ac.uk/Faq>

5.6.1 Comparing the FAQchat with Google

Google is “a search engine which is very easy to use. It returns pages based on the number of sites linking to them and how often they are visited, indicating their popularity.” (SeniorNet 2004). Search engines like Google retrieve information in four phases (Boyle 2003, SeniorNet 2004):

1. Obtaining documents to be searched. The method used gives a classification of search engine types:
 - Search engines which use crawlers, or spiders to get URLs such as Google;
 - Search engines based on human submission;
 - Others that are a combination of the two.
2. Preparing the documents to be searched, which involve operations such as: filtering the text, and extracting the meaningful terms.
3. Indexing the terms. One of the mechanisms used by Google is the inverted file structure. Three stages are applied :
 - a. Each document is given a unique ID;
 - b. A dictionary of all stemmed words from all documents is created;
 - c. Each term in the dictionary is associated with a pointer to the inversion list. The inversion list associates each term to all documents containing it.
4. The matching process which is used to give the best answer to a specific user query. One of the most widely used methods is the vector space model, where a two-dimensional array (term by document) is constructed with size $M \times N$; M represents the items in dictionary, and N represents the documents. A weighting scheme may be applied such as column normalisation or tf-idf. The user query is represented as a vector of size M , and it is normalised, stemmed, and weighted in the same manner as the document's items. At the end the best hit will be selected using different methods of ranking. This ranking algorithm is the “hidden ingredient” differentiating rival search engines.

Most search engines break up the user query into keywords, and return results according to keyword matches like Google. AskJeeves⁶ is a search engine that returns a result after analysing the query, using a question-processing engine to help understand the meaning of the words and grammar of the question. FAQchat is a compromise between the two. In retrieving information FAQchat will try to give the results using most significant words as keywords, and try to find the longest pattern to match without using any linguistic tools, or analysing the meaning. FAQchat does not need any linguistic knowledge module, and also in principle is language independent: it can be trained with FAQs in any natural language. The way FAQchat works was described in section 4.2.

The aim of this evaluation is to show that FAQchat works properly; it is not a search engine, but it could be a tool to access web pages, and giving answers from FAQ databases. The aim is not specifically to measure comparative success of Google against FAQchat, merely to demonstrate the FAQchat is a viable alternative.

Moreover, the most significant word approach has already been used to develop earlier versions of the chatbot, which deal with text and dialogues as illustrated in section 4.7. The aim of this experiment is to show that the same approach is applicable with the FAQ database.

5.6.2 The methodology

To evaluate FAQchat, the following approach was taken:

1. An interface was built, which has a box to accept the user input, and a button to send this to the system. The outcomes appear in two columns: one holds the FAQchat answer, and the other holds the Google answer after filtering it to the FAQ database as shown in figure 5.16. Google allows searches to be restricted to a given URL, but this still yields all matches from the whole SoC website⁷ so a Perl script was required to exclude matches not from the FAQ sub-pages.
2. An evaluation sheet was prepared which contains 15 information-seeking tasks or questions on a range of different topics related to the FAQ database, as shown in appendix A. The tasks were suggested by a range

⁶ <http://ask.co.uk/home>

⁷ <http://www.comp.leeds.ac.uk>

of users including SoC staff and research students to cover the three possibilities where the FAQchat could find a direct answer, links to more than one possible answer, and where the FAQchat could not find any answer. In order not to restrict users to these tasks, and not to bias to specific topics, the evaluation sheet included spaces for users to try 5 additional tasks or questions of their own choosing. Users were free to decide exactly what input string to give to FAQchat to find an answer: they were not required to type questions verbatim; users were free to try more than once if no appropriate answer was found; users could reformulate the query.

3. The evaluation sheet was completed by 21 members of the staff and students. Users were asked to try using the system, and state whether they were able to find answers using the FAQchat responses, or using the Google responses; and which of the two they preferred and why.

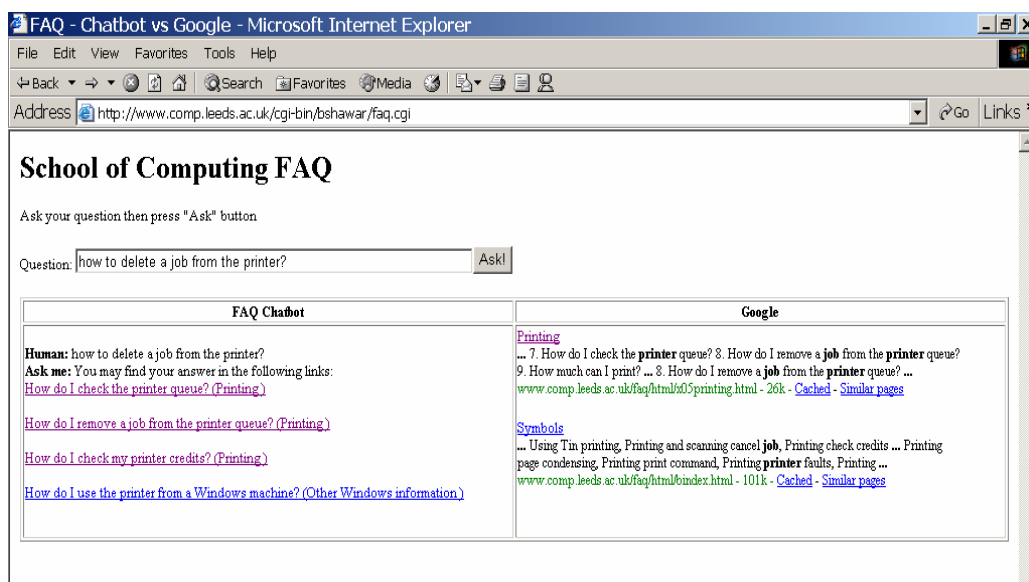


Figure 5. 16 FAQchat interface

5.6.3 Results and discussions

Twenty-one users tried the system; nine members of the staff and the rest mainly were postgraduates. The analysis was tackled in two directions: the preference and the number of matches found per question and per user.

Number of matches per question

Table 5.7 summarises the number of evaluators who managed to find answers by FAQchat (denoted by “C”) and Google (denoted by “G”), for each question.

Questions	Staff		Students		Overall	
	C	G	C	G	C	G
Q1: Can you find out how to convert word doc to PDF?	6	6	7	10	13	16
Q2: Can you find out what scanners are in the school?	6	6	8	9	14	15
Q3: Can you find out the most interesting module(s) for the third year?	2	2	4	3	6	5
Q4: Can you find out the opening and closing time of the labs?	8	4	11	6	19	10
Q5: Can you find out how to set up a personal website?	4	2	10	5	14	7
Q6: Can you find out how to use Linux?	3	3	11	11	14	14
Q7: Can you find out when and where can you submit your coursework?	6	5	10	6	16	11
Q8: Can you find out what to do if you didn't attend an exam?	8	3	12	4	20	7
Q9: Can you find out what to do if you have problems with your supervisor?	8	7	10	8	18	15
Q10: Can you find out how to delete a job from the printer?	8	8	12	9	20	17
Q11: Can you find out what students can do after graduation?	2	1	3	0	5	1
Q12: Can you find out about available tools for knowledge discovery?	2	0	3	1	5	1
Q13: Can you find out how to write a PhD thesis?	1	5	9	7	10	12
Q14: Can you find out the E_mail address of a member staff?	8	1	10	3	18	4
Q15: Can you find out how to print an email message from Pine?	8	4	9	8	17	12

Table 5. 7 Can you find the answer? Analysed per question

The mean for each class of users (staff and student), and for each tool (FAQchat and Google) has been calculated using formula 1:

$$Mean = \frac{\sum_{i=1}^n Xi}{n} \quad \text{Formula 1}$$

Where: X_i represents the number of users in group i (staff, students, overall), who found answer.

n is the total number of questions which is 15.

The proportion was calculated using formula 2:

$$Proportion = \frac{\sum_{i=1}^n X_i}{(n \times G)}$$

Formula 2

Where: G is the total number of users in each group (staff, students), which are 9 and 12 correspondingly.

Users / Tool	Mean number of users finding answers		Proportion of users finding answers	
	FAQchat	Google	FAQchat	Google
Staff	5.33	3.8	59%	42%
Students	8.6	6.0	72%	50%
Overall	13.93	9.8	66%	47%

Table 5. 8 Proportion of users finding answers by FAQchat and Google

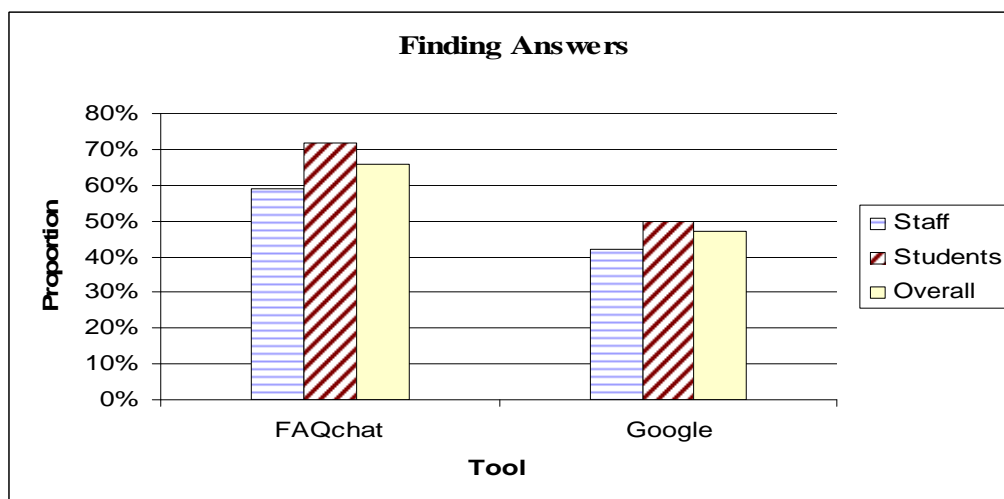


Figure 5. 17 The proportion of finding answers by the FAQchat and Google.

Results in table 5.8 and figure 5.17 show that 66% overall of our sample of users managed to find answers using the FAQchat while 47% found it by Google. Since there is no specific format to ask the question, there are cases where some users could find answers while others could not. The success in finding answers is based on the way the questions were presented to FAQchat.

Of the overall sample, the staff outcome shows that 59% were able to find answers by FAQchat compared to 72% of students who managed to do so; students were more successful than staff.

The preferred tool per question

For each question, users were asked to state which tool they preferred to use to find the answer. Results are presented in table 5.9. Some users had no preference.

Questions	Staff		Students		Overall	
	C	G	C	G	C	G
Q1: How to convert word doc to PDF?	3	4	4	5	7	9
Q2: What scanners are in the school?	5	1	7	0	12	1
Q3: The most interesting module(s) for the third year?	3	2	7	2	10	4
Q4: The opening and closing time of the labs?	5	2	9	1	14	3
Q5: How to set up a personal website?	3	1	8	1	11	2
Q6: How to use Linux?	3	0	4	2	7	2
Q7: when and where can you submit your coursework?	2	0	4	0	6	0
Q8: What to do if you didn't attend an exam?	7	0	11	0	18	0
Q9: What to do if you have problems with your supervisor?	2	0	6	0	8	0
Q10: How to delete a job from the printer?	3	0	8	2	11	2
Q11: What students can do after graduation?	1	2	3	0	4	2
Q12: Can you find out about available tools for knowledge discovery?	3	1	3	1	6	2
Q13: How to write a PhD thesis?	1	5	5	2	6	7
Q14: Can you find out the E_mail address of a member staff?	9	0	8	1	17	1
Q15: How to print an email message from Pine?	6	1	5	0	11	1

Table 5. 9 Which tool do you prefer? Analysed per question

The proportion of users who preferred each tool was calculated using formula 2. Table 5.10 and figure 5.18 show that 51% of the staff, 41% of the students, and 47% overall preferred using FAQchat against 11% who preferred Google.

Users / Tool	Mean of users who preferred		Proportion of users who preferred	
	FAQchat	Google	FAQchat	Google
Staff member	3.733	1.27	41%	14%
Students	6.13	1.13	51%	9%
Overall	9.87	2.4	47%	11%

Table 5. 10 Proportion of users' preference tool, analysed per question

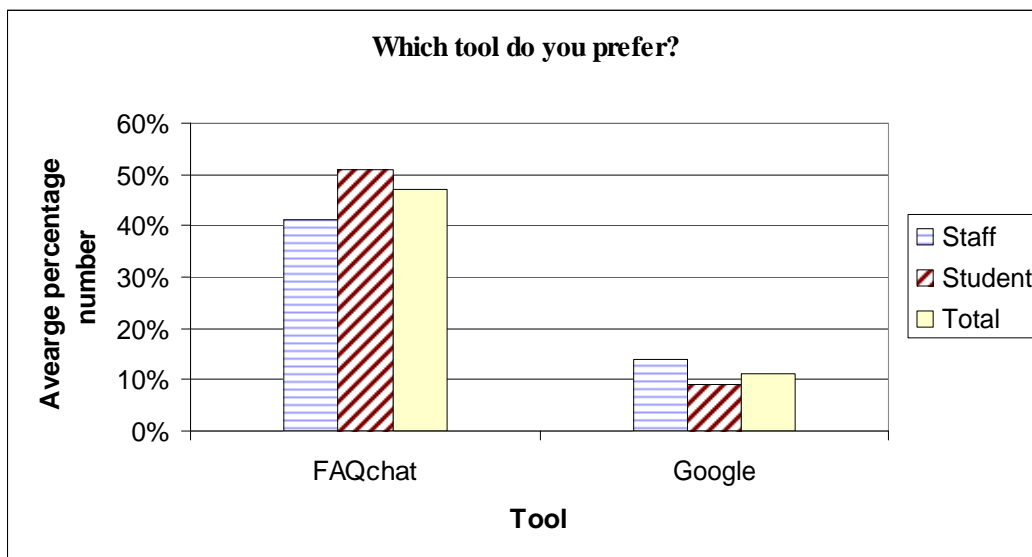


Figure 5. 18 Which tool do you prefer? Summary

Number of matches and preference found per user

The number of answers each user had found was recorded in table 5.11. The proportions found were the same as the ones in table 5.8 and table 5.10. However, the mean presented in table 5.12 was different because there are 9 staff, 12 students and 21 overall.

Staff	FAQchat (Y)	Google (Y)	Prefer C	Prefer G	Students	FAQchat (Y)	Google (Y)	Prefer C	Prefer G
S1	8	7	2	2	T1	9	5	9	5
S2	12	6	9	1	T2	10	9	10	9
S3	10	7	7	2	T3	11	6	11	6
S4	10	7	5	2	T4	9	7	9	7
S5	11	8	10	1	T5	10	8	10	8
S6	12	7	7	1	T6	11	6	11	6
S7	6	5	5	4	T7	14	5	14	5
S8	6	5	8	2	T8	11	6	11	6
S9	8	6	3	4	T9	12	9	12	9
					T10	9	5	9	5
					T11	12	10	12	10
					T12	14	8	14	8

Table 5. 11 Number of matches found per user

Users / Tool	Mean no. of questions for which answers were found		Mean of preference expressed	
	FAQchat	Google	FAQchat	Google
Staff member	9.22	6.44	6.22	2.11
Students	11	7	7.67	1.42
Overall	10.24	6.95	7.05	1.71

Table 5. 12 The mean for giving answers and preference each user found

The evaluation sheet ended with an open section inviting general feedback. The following feedback was obtained:

1. Both staff and students preferred using the FAQchat for two main reasons:
 - The ability to give direct answers sometimes while Google only gives links.
 - The number of links returned by the FAQchat is less than those returned by Google for some questions, which saves time browsing/searching.

2. Users who preferred Google justified their preference for two reasons:
 - Prior familiarity with using Google.
 - FAQchat seemed harder to steer with carefully chosen keywords, but more often did well on the first try. This happens because FAQchat gives answers if the keyword matches a significant word. The same will occur if you reformulate the question and the FAQchat matches the same word. However, Google will give different answers in this case.

An interesting additional result is how often one system found an answer but the other did not. In 9% of cases Google found an answer but FAQchat did not. However, in 28% of cases FAQchat found an answer but Google did not. This resulted from the different matching technique used by each tool.

Unfortunately just 4 users tried to add extra questions. Formula 1 was used to find the mean, where $n = 5$ and $G = 4$. Overall, 35% of the sample preferred using FAQchat and 25% preferred Google.

5.6.4 Testing reliability

The previous results were true for the sample; however, to be sure that it is reliable and not produced by chance, the t-Test was used.

The main goal of the t-Test is “to determine whether the means of two groups of scores differ to a statistically significant degree” (Kranzler and Moursund 1999). The paired t-Test is selected to compare between FAQchat and Google for each class of users (staff and students). Formula 3 has been used (Greer and Mulhern 2002) to calculate T:

$$T = \frac{md}{sd} \times \sqrt{k} \quad \text{Formula 3}$$

Where:

- md is the difference of mean between the two groups,
- sd is the standard deviation for the difference,
- k is the total number in the group.

To extract the T_{crit} from a table, the degree of freedom must be calculated using formula 4.

$$df = k - 1 \quad \text{Formula 4}$$

Analysis comparing hits per question and per user

First, the data presented in table 5.7 and table 5.11 was analysed in table 5.13 which clarifies the results of finding answers by the two tools. In table 5.15 each row denoted by question represents matches found per question (table 5.9) where $k=15$. Each row denoted by user represents matches found per user (table 5.13) where $k=9$ for the staff, $k=12$ for the students, and 21 overall. Again C denotes FAQchat and G denotes Google. P denotes the probability of assuming the null hypothesis. The null hypothesis is “there is no difference in the mean in finding answers by FAQchat and Google”.

Finding answers		Mean		md	sd	T	df	T _{crit}	P
		C	G						
Staff	question	5.54	3.87	1.67	2.61	2.47	14	2.145	0.027
	user	9.22	6.44	2.78	1.79	4.66	8	2.306	0.002
Students	question	8.8	5.87	6.25	2.76	4.11	14	2.145	0.001
	user	11.0	7.00	4.00	2.22	6.25	11	2.201	0.000
Overall	question	14.3	9.73	4.6	4.84	3.68	14	2.145	0.002
	user	10.2	6.95	3.29	2.08	7.25	20	2.086	0.000

Table 5. 13 Paired t-Test for finding answers analysed per question and per user

As shown in table 5.13, $T > T_{crit}$, which means the assumption that there is no difference was incorrect. The result implies that either staff, students, and overall manage to find more answers using FAQchat or both tools were equivalent in giving answers, these results are summarised in figure 5.19.

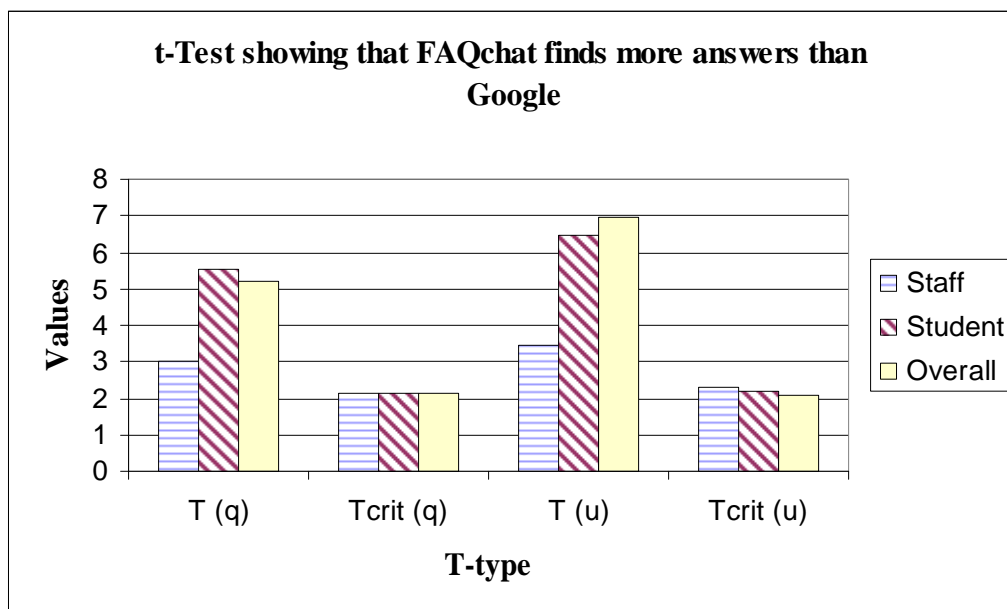


Figure 5. 19 Comparing t-Test values for finding answers

Analysis: Which tool do you prefer?

The process was repeated to test the significance for the preferences. Table 5.14 summarises the results obtained from the data in tables 5.9 and 5.11. The null hypothesis is: “there is no difference in the mean between the two tools according to users’ preference”.

Which tool do you prefer?		Mean		md	sd	T	df	T _{crit}	P
		C	G						
Staff	question	3.73	1.27	2.47	3.18	3.00	14	2.145	0.010
	user	6.22	2.11	4.11	3.55	3.47	8	2.306	0.008
Students	question	6.49	1.14	6.49	2.95	6.56	14	2.145	0.000
	user	7.67	1.42	6.25	3.33	6.49	11	2.201	0.000
Overall	question	9.87	2.40	7.47	5.57	5.20	14	2.145	0.000
	user	7.05	1.71	5.33	3.51	6.96	20	2.086	0.000

Table 5. 14 Paired t-Test for users preferred tool analysed per question and per user

Again clear results in all cases were gained, T was greater than T_{crit} in both analyses “per question” and “per user” for staff, students and overall. Figure 5.20 is a summary analogous to figure 5.19. Consequently the null hypothesis was rejected. Overall, we concluded that users prefer using the FAQchat tool.

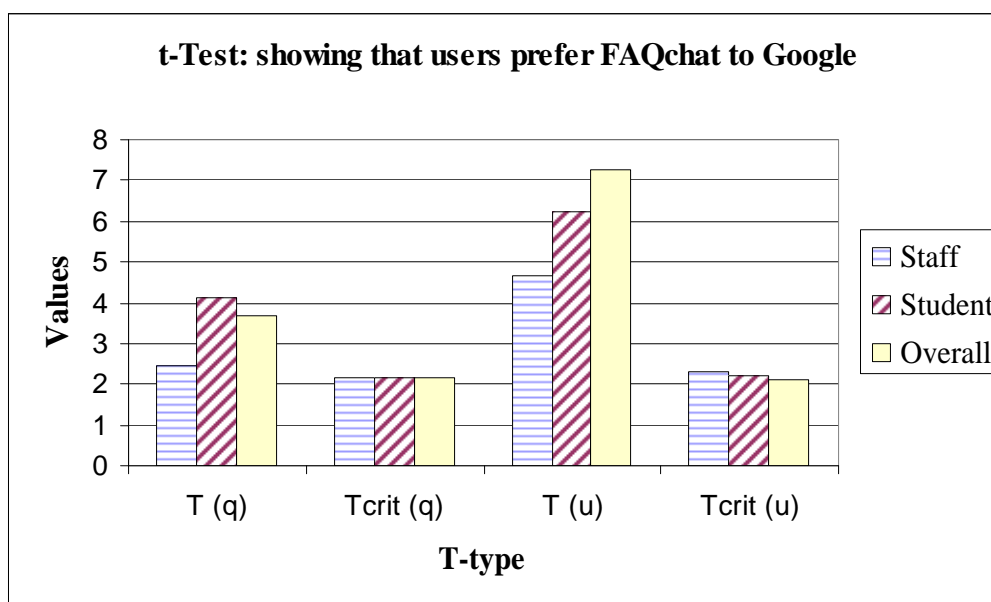


Figure 5. 20 Comparing t-Test values for the preferred tool.

5.6.5 Samples of chatting

In this section samples of asking questions to FAQchat and Google are presented which show the different answer possibilities: direct answer, or a set of related links as shown in figure 5.21 and figure 5.22.

a. Direct answer generated by FAQchat using the FAQ of the school of computing at University of Leeds

Human: How can I remove a job from the printer?

FAQchat:

If you want to remove a job from the printer queue then find out the job number using the [lpq command](#), and in a UNIX shell type:

```
% lprm -P printername job_no
```

You can only remove jobs that you have sent to the printer yourself - apart from Support that is.

For more information look at:

[How do I remove a job from the printer queue? \(Printing \)](#)

b. Related links generated by FAQchat

Human: How can I delete a job from a printer?

FAQchat: You may find your answer in the following links:

[How do I check the printer queue? \(Printing \)](#)

[How do I remove a job from the printer queue? \(Printing \)](#)

[How do I check my printer credits? \(Printing \)](#)

[How do I use the printer from a Windows machine? \(Other Windows information \)](#)

Figure 5. 21 Answers generated by FAQchat

FAQchat could generate either a direct answer or related links for rewordings of the same question. This is because the matching process is based on matching the most significant words found in the user input. In figure 5.21, sample a, the significant words are “remove” and “printer” in addition to the first word “how”. Where sample b, the significant word is “printer” only in addition to the first word “how”. According to that, different answers could be generated based on the way the question is formulated.

a. Answers generated by FAQchat

Human: What happen if i can't submit the coursework on time?

FAQchat: You may find your answer in the following links:

[Examinations/Coursework \(General Information \)](#)

[Why coursework? \(Examinations/Coursework \)](#)

[What if I can't hand coursework in on time? \(Examinations/Coursework \)](#)

[Can students be granted extensions or exemptions from coursework? \(Teaching \)](#)

[What are the penalties for late submission of coursework? \(Teaching \)](#)

b. Answers generated by Google for the same question

[Examinations/Coursework](#)

... 5. What happens **if** I **can't** attend my exam ... DOESN'T HAPPEN!! ... **If** a student is considered by the School Examinations Board to have made no serious attempt to pass a ...

www.comp.leeds.ac.uk/faq/public/x0704.html - 14k - Supplemental Result - [Cached](#) - [Similar pages](#)

Figure 5. 22 Answers generated by FAQchat and Google

Figure 5.22 shows an example where Google's answer is the same as the top answer produced by FAQchat. However, FAQchat provides all links related to courseworks, where the third link holds the real answer for the user's input.

5.7 Summary

In contrast to the Loebner Prize contest which is used to measure the ability of a chatbot to fool people that they are chatting with real humans, the evaluation process described in this chapter is tackled in three main directions:

- The naturalness of ALICE.
- The ability of the learning techniques in finding a match.
- Significance tests for user satisfaction with the results.

Even though ALICE won the Loebner prize three times, in this chapter the naturalness of ALICE was measured by comparing human-to-human versus

human-to-chatbot dialogues in three levels: lexical, semantic, and part of speech. The comparison shows that an over exaggeration in using key lexical grammatical and semantic features, such as speech-act verbs and first-person pronouns, are raised in ALICE as an attempt to simulate real human dialogues.

In order to measure the success of the Java program and the adopted learning techniques in the KGA prototype, three dialogue metrics were adopted:

- Dialogue efficiency to measure if the learning techniques increase the ability to find an answer by analysing the matching types.
- Dialogue quality to measure the quality of each response according to reasonable, weird but understandable, or nonsensical reply.
- Users' satisfaction assessment based on open-ended request for feedback.

Results showed that expanding the AIML pattern matching using the first-word and the most significant word approaches yielded more favourable feedback: the informants found the conversations less repetitive and more interesting. Responses were sometimes weird and apparently irrelevant, but at least they were not always "no answer".

The evaluation of the BNC system presented the ability of the program to generate more than one million categories. In contrast to the traditional way of visualisation by giving numerical figures, the chatbot was used as a tool to visualise a corpus, this means giving insight, or an overall picture of the data contents as illustrated by using the BNC chatbot versions.

The Arabic chatbot version was evaluated in terms of exploring the problems of using Qur'an Arabic language which are: the voweled text of the Qur'an, and the different interpretation of verses in the absence of the correct word meaning and the historical knowledge. However, the AIML interpreter can handle the Arabic language; the Java program was modified to cope with the textual nature of Qur'an instead of the dialogue ones; also the Qur'an version succeeded in giving an insight about the Qur'an's contents.

The FAQchat version was retrained using the FAQ of the School of Computing at the University of Leeds. A comparison was made between FAQchat answers and Google. The aim was not to try to evaluate the two systems to come up with relative scores, but to show that it is a viable alternative compared to search engines such as Google and it could be used as a tool to access FAQ databases. The t-Test was used to measure the significance of the results. Feedback favourable to FAQchat was gained from almost all users, even those who preferred Google. They found it a novel and interesting way to access the FAQ using natural language questions. Overall, about two thirds of users managed to find answers by FAQchat, and about two thirds of the users preferred to use it.

6. Conclusions

"Only when you drink from the river of silence shall you indeed sing. And when you have reached the mountain top, then you shall begin to climb."

Gibran Khalil Gibran, "The Prophet"

6.1 Summary of the work

In chapter two, chatbot history was reviewed starting from the first chatbot ELIZA, generated in 1966 to simulate a psychotherapist. After that, different chatbots or "machine conversation" systems were implemented. The chatbots presented were viewed in terms of four important characteristics which affected the progress of development:

1. Finding useful applications for the chatbots.
2. Improving the human-machine communication by studying the chatbot language.
3. Fooling the user that they are chatting with a real human by improving the pattern matching techniques used.
4. The relationship between chatbots and other computational linguistic fields such as information retrieval and information extraction.

In building chatbots, programmers adopted the idea the chatbot must have something to say or to do; based on this each conversational machine has its own application such as a commercial advisor or museum guide. Another trend which encourages scholars to develop chatbots is to study the language of the chatbot. These studies state that human machine communication through the Internet and chatting using text mode is a compromise between spoken and written language. In order to give reasonable answers and to improve the communication medium, different pattern matching techniques were used. Some of these techniques were based on keyword matching. However, others were more sophisticated such as using Markov models; or based on ideas imported from information retrieval systems.

Two chatbot systems ALICE and Elizabeth were discussed in detail in terms of knowledge base representation and the pattern matching techniques. The aim of this study was to select one of these systems to be automatically retrained. The ALICE system was selected for automation because:

1. The AIML language used by ALICE is a derivative form of the XML language.
2. The ALICE interpreter uses a simple pattern matching technique, yet delivers convincing dialogues.
3. The ALICE-derived chatbot could be published online using the Pandorobot service which allows a wide range of people to try it.
4. There is a large community of ALICE/AIML developers and users.

Different evaluation methods were reviewed, which cover dialogue systems and chatbot systems. The Loebner Prize contest is based on measuring the ability of the chatbot to fool the user that they are chatting with a real human. However, this evaluation is based on ten minutes of chatting. The chatting time was not restricted in the evaluation work in this thesis; also some of the black box evaluation metrics used in dialogue systems were used to measure the success of the machine learning techniques used in the automation process.

Chatbot technology resulted from merging computational linguistics and natural language engineering. Some of the computational linguistic, and language engineering tools were reviewed to show how the corpus based approach is used in different systems and domains.

Chapter 3 presented the corpora used to train the ALICE chatbot system. Corpora problems were classified into two main groups: human transcription mistakes, and human spoken language characteristics.

In chapter 4, the Java program which converts a machine readable text to AIML format was described. Different versions were implemented to handle different corpora. Each version has a different normalisation process to cope with the annotation and problems of the corpus. The most significant word approach was adopted to learn more categories from the corpus. This approach increases the

ability of giving an answer to users and finding a match. Chapter 5 describes the evaluation methodologies

6.2 Results

The main target in this thesis is to develop a general program to convert a machine readable text to a chatbot language. There are two main goals behind the automation process:

1. Overcome the problem of restricting the chatbot to a specific domain and language.
2. Generate chatbot versions which communicate with people in a way which is closer to real human spoken language.

To achieve these goals, the ALICE chatbot system was selected to be automated; and the corpus based approach was adopted to make the chatbot speak like a human. A Java program was developed to generate different chatbot versions that speak different languages. English, French, Afrikaans, and Arabic languages were used.

However, the AIML language has two main types of categories: the atomic, and the default categories. Extracting atomic categories from the corpora is done by assuming the first turn as a pattern and the second as a template. Since there is no guarantee that users will enter the same input as the one found in the corpus, the following learning techniques are adopted:

- The first word approach, where the first word was extracted from each pattern and a default category was built.
- The most significant word approach, where the least frequent word of each pattern was extracted and four default categories were generated.
- Another default category was built using the first word in addition to the most significant ones.

Other versions were improved using a stop list, and the first most significant word and the second most significant word were used to extend the availability

of finding a match. Continuing the generalisation trend, different corpora structures were tested:

- Dialogue corpora such as KGA, and BNC.
- Monologue or text corpus such as the Qur'an corpus.
- Frequently Asked Questions such as the FAQ of the School of Computing at University of Leeds.

The evaluation process is tackled as follows:

1. In order to measure the naturalness of ALICE, the Wmatrix tool was used to compare human-to-human dialogue to ALICE chatbot dialogues in three levels: lexical, semantic, and POS level. The comparison shows that ALICE tries to simulate real dialogue by exaggerating use of key lexical, grammatical and semantic features.
2. To evaluate the success of the learning techniques in extending the availability of finding a match, the Afrikaans versions were evaluated using dialogue efficiency, dialogue quality and user satisfaction. The first version was based on the first word approach. Few users managed to obtain sensible answers. After adding the significant word approach users were more satisfied.
3. The BNC version was evaluated in terms of the ability to generate more than one million categories.
4. The Qur'an version was evaluated in terms of the ability to adapt the program to handle the Arabic language, and the non conversational nature of the Qur'an.
5. FAQchat used the SoC Frequently Asked Questions; evaluation is based on comparing FAQchat with Google after filtering it to use the same FAQ as the chatbot. The t-Test was applied to check if the results are reliable. Most users found the chatbot an interesting way to access the Frequently Asked Questions, and succeeded in finding related answers.

6.3 System drawbacks

Drawbacks could be related to ALICE and the AIML interpretation, or from the automation process. ALICE and AIML interpreters' drawbacks are:

1. ALICE and other chatbots are unable to hold the dialogue history, which is an important characteristic in human conversations. The ALICE system can hold the history for one previous turn, which is obviously not enough.
2. The AIML interpreter does not handle any reference resolution.
3. The wildcards used by the AIML format are restricted: the pattern has to have something in a specific location; the empty string is not included. In this case to generate a general pattern to handle the existence of a significant word in different positions in the pattern, at least 4 categories will be added as shown in section 4.4.1. This increased the number of default categories, and the size of the file. This problem necessitates splitting the file to more than one to be accepted by the Pandorabot service.
4. AIML does not store “deep” or linguistic knowledge, only text patterns.

However, the proposal was to retrain ALICE, leaving its interpreter in its original state since it had won the Loebner Prize three times. Even in learning from the corpora, simple techniques were tested and succeeded in finding a match. No sophisticated algorithms or any dialogue models have been tried. Since the program was general, each version could be improved separately to work properly.

The main drawback in the developed program was that some answers generated by the chatbot are nonsensical. This could be caused by:

- The users’ input has nothing to do with topic of the corpus.
- The adopted learning techniques and the other approaches used to overcome dialogue corpora problems such as omitting overlapping and reiterating the patterns were not sufficient.

Non-dialogue corpora in particular domains were used in the hope that these drawbacks no longer existed as they are specific to general dialogue corpora. In this case, the success of the matching algorithm could be evaluated to see if it also causes nonsensical responses. Results showed better feedback and that the learning techniques give better (correct or related) answers in the case of FAQ corpus where the dialogue corpora problems are not found. The FAQ answers

were the most reasonable answers according to other corpora. This may be related to the generalisation approach itself, so it may be better to restrict the chatbot to a particular domain.

6.4 Future work

A new project is proposed to investigate the possibility of using a chatbot system in assisting:

- Teaching courses which are factual in their nature.
- Learning new languages.
- Improving students' understanding for some modules by answering questions, and chatting about the topic.

Recent trends in education focus on using computer mediated communication, which could help e-learning, distance learning, and also full time learning through sharing ideas and developing the students' writing skills. A good agent for this purpose could be a chatbot. Recent approaches focus on using the chatbot to simulate an expert academic for some courses. For example:

- Cooperation between researchers at Huddersfield and Manchester Metropolitan universities succeeded in “using a chatbot conversation to enhance the learning of social theory” (Gibbs et al. 2004). The idea is based on using the ALICE chatbot engine to build knowledge base that answers questions of the type “tell me more about...” The trainable chatbot was tested by students in Huddersfield and Manchester who are studying a natural social theory course.
- The Elizabeth chatbot was implemented by Dr. Millican at University of Leeds in 2002. Dr. Millican used Elizabeth in the Artificial Intelligence module, to teach undergraduate students how to use natural language to build a chatbot, and to analyse the grammar of specific sentences. Elizabeth has the ability to produce a grammar structure analysis of a sentence using set of input transformation rules to represent grammar.

- Speak2Me¹ is another chatbot which is used to assist in learning English language through chatting. The chatbot accepts the input in English text mode, and returns the reply in both textual and voice media. This tool is used to improve the user's writing skills and users' pronunciation.

6.4.1 Project aims

1. Developing knowledge base automatically. This could be applicable for any language in any domain with any type of corpus: structured text such as FAQ, dialogue, and text extracted from books and tutorials.
2. Testing the impact of using the chatbot technology in the teaching process, and making all material available.
3. Improving students' understanding of some courses which is empirical, substantive and factual in nature by creating an environment where students can work together on the World Wide Web.
4. Improving students' skills in writing and learning from each other by sharing ideas.

6.4.2 Project plans

The plan will be tackled as follows:

1. Extracting the knowledge base and improving the matching process.
2. Using the hand-crafted database which is used in previous versions such as the one used in teaching social theory in Huddersfield and Manchester.
3. Using the Java program to extract patterns and templates from this database and retrain ALICE by the automatic generated files. Evaluate the new version comparing it with the manual one by students who are studying this course, using the same interface.
4. Improving the matching process by using some techniques from information retrieval systems and search engines.
5. Adapting the system to use teaching materials such as books, powerPoint slides etc, to extract the knowledge automatically for different modules in terms of patterns (questions) and templates (answers).

¹ www.speak2Me.net

6. Using different languages sources especially Arabic language and extracting AIML categories for it, to build version of ALICE that could be used to teach language.

6.5 Overall conclusions

The thesis provides answers to the research questions and objectives stated in chapter 1. The Java program which was implemented to convert the text to the AIML format and the simple learning techniques succeeded in:

- Extracting more than one million categories (input-output rules).
- Generating chatbot versions speaking different languages, where some of these languages were unknown.
- Using the chatbot as a tool to retrieve answers for questions, and accessing information systems.

Overall, specialised chatbots have a wide range of potential applications; and corpus-based machine learning allows us to automate development of specialised chatbots. Clearly, the corpus-based approach is good in terms of generating human like chatting. However, to have a good quality language, the corpus must be cleaned first in terms of spelling and grammatical errors especially if it will be used for educational purposes. Another direction to improve the matching technique may be by using another chatbot (e.g. MegaHAL) which use a sophisticated pattern-matching, or either building a new chatbot from scratch for this purpose.

Finally, as Colby (1999a) states, “We need not take human-human conversation as the gold standard for conversational exchanges. If one had a perfect simulation of a human conversant, then it would be human-human conversation and not human-computer conversation with its sometimes odd but pertinent properties.”

7. References

- Abu Shawar, B. and Atwell, E.** (2003a). Using dialogue corpora to retrain a chatbot system. In Archer, D., Rayson, P., Wilson, A. and McEnery, T. (eds.), *Proceedings of the Corpus Linguistics 2003 conference (CL2003)*. Lancaster University, UK, pp. 681-690.
- Abu Shawar, B. and Atwell E.** (2003b). Using the Corpus of Spoken Afrikaans to generate an Afrikaans chatbot. *SALALS Journal: Southern African Linguistics and Applied Language Studies*. Vol. 21, pp. 283-294.
- Abu Shawar, B. and Atwell E.** (2004a). A chatbot as a novel corpus visualization tool. In Lino, M.T., Xavier, M.F., Ferreira, F., Costa, R., Silva, R., Pereira, C., Cavalho, F., Lopes, M., Catarino, M. and Barros, S. (eds.), *Proceedings of Fourth International Conference on Language Resources and Evaluation (LREC'04)*. Vol. VI, pp. 2057-2060.
- Abu Shawar B. and Atwell E.** (2004b). *Accessing an information system by chatting*. In Farid Meziane and Elisabeth Metais (eds.), *Proceedings of 9th International conference on the Application of Natural Language to Information Systems, NLDB 2004*. Salford, UK, pp. 396-401, LNCS Lecture Notes in Computer Science, Springer.
- Abu Shawar, B. and Atwell, Eric.** (2005a). A chatbot system as a tool to animate a corpus. *ICAME Journal*. Vol. 29, pp. 5-23.
- Abu Shawar, B., Atwell, E. and Roberts, A.** (2005b). *FAQchat as an information retrieval system*. To appear in *Proceeding of L&T05 conference*. www.ltc.amu.edu.pl
- Aijmer, K. and Altenberg, B.** (1991). Introduction. In Aijmer, K. and Altenberg, B. (eds.), *English Corpus Linguistics: Studies in honour of Jan Svartvik*. Longman, London, pp. 1-6.
- Alabiso, B. and Kronfeld A.** (1999). LEAP: language enabled applications. In Wilks, Y. (ed.), *Machine Conversations*. Kluwer, Boston/Dordrecht/London, pp. 187-204.
- Al-Daimi, K., and Abdel-Amir, M.** (1994). The syntactic analysis of Arabic by machine. *Computers and Humanities*. Vol. 28, No. 1, pp. 29-37.

- Andersen, G. and Stenström, A.** (1996). COTL: a progress report. *ICAME Journal*. Vol. 20, pp. 133-136.
- Aston, G. and Burnard, L.** (1998). *The BNC Handbook: Exploring the British National Corpus with SARA*. Edinburgh University Press, Edinburgh, pp. 1-42.
- Atwell, E.** (1996a). Comparative evaluation of grammatical annotation models. In Sutcliffe, R., Koch, H-D., McElligott, A. (eds), *Industrial Parsing of Technical Manuals*. Rodopi, Amsterdam , pp. 25-46.
- Atwell, E.** (1996b). Machine Learning from corpus resources for speech and handwriting recognition. In Thomas, J. and Short M (eds.), *Using Corpora for Language Research: Studies in the Honour of Geoffrey Leech*. Longman, Harlow, pp. 151-166.
- Atwell, E.** (1999). *The Language Machine*. British Council, London.
- Atwell, E.** (2003). A word-token-based machine learning algorithm for neoposy: coining new parts of speech. In: Archer, D., Rayson, P., Wilson, A. and McEnery, T. (eds.), *Proceedings of International Conference on Corpus Linguistics (CL2003)*. Lancaster, Lancaster University, pp. 43-47.
- Atwell, E.** (2004). Machine Learning Approaches to Analysis of Corpora. In Bel, B. and Marlien, I. (eds.), *Proceedings of TALN04: 11th Conference on Natural Language Processing*. Fes, Maroc, pp. 27-32.
- Atwell, E., Demetriou, G., Hughes, J., Schiffrin, A., Souter, C. and Wilcock, S.** (2000a). A comparative evaluation of modern English corpus grammatical annotation schemes. *ICAME Journal*. Vol. 24, pp. 7-23.
- Atwell, E. and Elliott, J.** (2001). A corpus for interstellar communication. In Rayson, P., Wilson, A., McEnery, T., Hardie, A. and Khoja, S. (eds.) *Proceedings of International Conference on Corpus Linguistics (CL2001)*. Lancaster University, UK., pp. 31-39.
- Atwell, E., Howarth, P., Souter, C.** (2003). The ISLE corpus: Italian and German spoken learners' English. *ICAME Journal*. Vol. 27, pp. 5-18.
- Atwell E, Howarth, P., Souter, C., Baldo, P., Bisiani, R., Pezzotta, D., Bonaventura, P., Menzel, W., Herron, D., Morton, R. and Schmidt, J.** (2000b). User-Guided System Development in Interactive Spoken Language Education. *Natural Language Engineering journal*. Vol.6 No.3-4, Special Issue on Best Practice in Spoken Language Dialogue Systems Engineering, pp. 229-241.

- Aust, H., Oerder, M., Seide, F. and Steinbiss, V.** (1995). The Philips automatic train timetable information system. *Speech Communication*. Vol 17, pp. 249-262.
- Baeza-Yates, R. and Ribeiro-Neto, B.** (1999). *Modern Information Retrieval*. ACM Press, New York, pp. 1-18.
- Ball, G.** (1999). The SpeakEasy dialogue controller. In Wilks, Y. (ed.), *Machine Conversations*. Kluwer, Boston/Dordrecht/London, pp. 47-56.
- Bastin, V. and Cordier, D.** (1998). Methods and tricks used in an attempt to pass the Turing Test. In D.M.W. Powers (ed.), *NeMLap3/CoNLL98 Workshop on Human Computer Conversation (ACL)*, pp. 275-277.
- Batacharia, B., Levy, D., Catizone R., Krotov A. and Wilks, Y.** (1999). CONVERSE: a conversational companion. In Wilks, Y. (ed.), *Machine Conversations*. Kluwer, Boston/Dordrecht/London, pp. 205-215.
- Block, N.** (1981). Psychologism and behaviourism. *Philosophical Review*. Vol. 90, No. 1, pp. 5-43.
- Boguraev, B., Garigliano, R. and Tait, J.** (1995). Editorial. *Natural Language Engineering*. Vol. 1, No. 1, pp. 1-7.
- Bontcheva, K., Cunningham, V., Tablan, V., Maynard, D. and Hamza, O.** (2002). Using GATE as an environment for teaching NLP. In *Proceedings of Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics*. Philadelphia, pp. 54-62.
- Boyle, R .** (2003). Understanding search engines. In: Boyle, R (ed.), *COMP1600: SY11 Introduction to Computer Systems 1. Lecture Notes*. School of Computing, University of Leeds, UK, pp. 65-72.
- Brin, S.** (1998). Extracting patterns and relations from the World Wide Web. In *WebDB Workshop*, pp. 172-183.
- Burnard, L. and Dodd, T.** (2003). Xara: an XML aware tool for corpus searching. In Archer, D., Rayson, P., Wilson, A. and McEnery, T. (eds.), *Proceedings of the Corpus Linguistics 2003 conference (CL2003)*. Lancaster, Lancaster University, UK, pp.142-144.
- Cassell, J., Torres O., and Prevost S.** (1999) Turn taking versus discourse structure. In Wilks, Y. (ed.), *Machine Conversations*. Kluwer, Boston/Dordrecht/London, pp. 143-153.

- Cheepen, C. and Monaghan, J.** (1999) Designing for naturalness in automated dialogues. In Wilks, Y. (ed.), *Machine Conversations*. Kluwer, Boston/Dordrecht/London, pp. 127-142.
- Churcher, G., Atwell, E. and Souter, C.** (1997). The semantic/pragmatic annotation of an Air Traffic Control corpus for use in Speech Recognition. In Ljung, M. (ed.), *Corpus-based Studies in English: Papers from ICAME 17th International Conference on English Language Research on Computerized Corpora*. Rodopi, Amsterdam, pp. 353- 374.
- Colby, K.** (1973). Simulation of belief systems. In Schank, R. and Colby, K. (eds.), *Computer Models of Thought and Language*. Freeman, San Francisco, pp. 251-286.
- Colby, K.** (1999a). Comments on human-computer conversation. In Wilks, Y. (ed.), *Machine Conversations*. Kluwer, Boston/Dordrecht/London, pp. 5-8.
- Colby, K.** (1999b). Human-computer conversation in a cognitive therapy program. In Wilks, Y. (ed.), *Machine Conversations*. Kluwer, Boston/Dordrecht/London, pp. 9-19.
- Cowie, J. and Lenhart, W.** (1996). Information extraction. *Communications of the ACM*. Vol. 39, No. 1, pp. 80-91.
- Cowie, J. and Wilks, Y.** (2000). Information Extraction. In Dale, R., Moisl, H. and Somers, H. (eds.). *Handbook of Natural Language Processing*. Marcel Dekker, New York.
- Costello, F. and Smeaton, A.** (2004). *Question-Answering Systems*. [Online]: <http://www.computing.dcu.ie/~away/PROJ3/00-01/lenahan.html>
- Cunningham, H.** (1999). A definition and short history of language engineering. *Journal of Natural Language Engineering*. Vol 5, No. 1, pp. 1-16.
- Cunningham, H.** (2002). GATE, a General Architecture for Text Engineering. *Computers and the humanities*. Vol. 36, No.2, pp. 223-254.
- Cunningham, H., Gaizauskas, R., Humphreys, K., and Wilks, Y.** (1999). Experience with a language engineering architecture: 3 years of GATE. In *Proceedings of the AISB'99 Workshop on Reference Architectures and Data Standards for NLP*.

- Cunningham, H., Wilks, Y., and Gaizauskas, R.** (1996) GATE- a General Architecture for Text Engineering. In *Proceedings of the 16th conference on computational linguistics (COLING96)*. Vol. 1, Copenhagen, pp. 1057-1060.
- Demetriou, G. and Atwell, E.** (2001). A domain-independent semantic tagger for the study of meaning associations in English text. In Bunt, H., Sluis, I. and Thijsse, E. (eds.), *Proceedings of the Fourth International Workshop on Computational Semantics (IWCS-4)*. Tilburg, Netherlands, pp. 67-80.
- De Roeck, A.** (2002). Arabic for absolute beginner. *The ELRA Newsletter*. Vol. 7, No. 1, pp. 3-4.
- Dumais, S., Banko, M., Brill, E., Lin, J., and Ng, A.** (2002). Web question answering: is more always better?. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. (SIGIR 2002)*. Tampere, Finland, pp. 291-298.
- Elliott, D., Atwell, E., and Hartley, A.** (2004). Compiling and using a shareable parallel corpus for MT evaluation. In *Proceedings of the Workshop on The Amazing Utility of Parallel and Comparable Corpora. Fourth International Conference on Language Resources and Evaluation (LREC)*. Lisbon, Portugal, pp. 18-21.
- Epstein R.** (1992). Can machines think? The quest for the thinking computer. *AI Magazine*. Vol 13, No. 2, pp. 80-95.
- Flycht-Eriksson, A.** (2001). *Domain Knowledge Management in Information-providing Dialogue Systems*. Linköping Studies in Science and Technology, Thesis No. 890. pp. 1-33.
- Fraser, N.** (1997). Assessment of interactive systems. In Gibbon, D., Moore, R. and Winski, R. (eds.), *Handbook of Standards and Resources for Spoken Language Systems*. Mouton de Gruyter, New York, pp. 564-614.
- Gaizauskas, R., Cunningham, H., Wilks, Y., Rodgers, P., and Humphreys, K.** (1996). GATE: an environment to support research and development in natural language engineering. In *Proceedings of the 8th IEEE International Conference on Tools with Artificial Intelligence*. Toulouse, France.
- Garside, R.** (1987). The CLAWS word tagging system. In Garside, R., Leech, G. and Sampson, G. (eds.), *The Computational Analysis of English*. Longman, London, pp. 30-41.

- Gibbs, G.R., Cameron, C., Kemenade, R., Teal, A. and Phillips, D.** (2004). *Using a chatbot conversation to enhance the learning of social theory*. [Online]: <http://www.hud.ac.uk/hhs/dbs/psysoc/research/SSCRG/chatbot.htm>
- Granger, S., Dagneaux, E. and Meunier, F.** (2002). The International Corpus of Learner English. *Handbook and CD-ROM*. Presses Universitaires de Louvain, Louvain-la-Neuve.
- Greenbaum, S.** (ed). (1996). *Comparing English Worldwide: The International Corpus of English*. Clarendon Press, Oxford.
- Greer, B. and Mulhern, G.** (2002). *Making sense of data and statistics in psychology*. Palgrave,UK, pp.100-120.
- Grishman, R.** (1986). *Computational Linguistics: an introduction*. Cambridge University Press, Cambridge, pp. 1-9.
- Grishman, R.** (1997). Information extraction techniques and challenges. In Pazienza, M. (ed.), *Information extraction: a Multidisciplinary Approach to an Emerging Information Technology*. International summer school, Frascati, Italy. Springer, Berlin, pp. 10-26.
- Grishman, R. and Yangarber, R.** (2000). *Issues in Corpus-Trained Information Extraction*. Computer Science Department. New York University, New York. [Online]: <http://nlp.cs.nyu.edu/publication/papers/yangarber-tokyo00.doc>
- Grondelaers, S., Speelman, D. and Geeraerts, D.** (2003). A corpus-based approach to informality: the case of Internet chat . In Archer, D., Rayson, P., Wilson, A. and McEnery, T. (eds.), *Proceedings of the Corpus Linguistics 2003 conference (CL2003)*. Lancaster, Lancaster University, UK, pp. 264.
- Güzeldere, G. and Franchi, S.** (1995). Dialogue with colourful personalities of early ai. *Stanford Electronic Humanities Review*. Vol. 4, No. 2, pp. 1-9. [On line]: <http://www.stanford.edu/group/SHR/4-2/text/toc.html>
- Hasida, K. and Den, Y.** (1999). A synthetic evaluation of dialogue systems. In Wilks, Y. (ed.), *Machine Conversations*. Kluwer, Boston/Dordrecht/London, pp. 113- 126.
- Herring, S.** (1996). Introduction. In Herring, S. (ed.), *Computer-Mediated Communication: Linguistics, Social and Cross-Cultural Perspectives*. John Benjamins, Amsterdam, pp. 1-10.

- Hewett, T.** (1992). ACM SIGCHI Curricula for Human Computer Interaction. Association for Computing Machinery. [Online]: <http://sigchi.org/cdg/index.html>
- Hirschman, L.** (1995). The Roles of language processing in a spoken language interface. In Roe, D. and Wilpon, J. (eds.), *Voice Communication Between Humans and Machines*. National Academy Press, Washington DC, pp. 217-237.
- Hirschman, L. and Thompson, H.** (1997). Overview of evaluation in speech and natural language processing. In Cole, R.A., Mariani, J., Uszkoreit, H., Zaenen, A. and Zue V. (eds.), *State of the Art in Natural Language Processing*. Cambridge University Press, Cambridge, pp. 475-518.
- Hughes, J. and Atwell, E.** (1994). The automated evaluation of inferred word classifications. In Cohn, A. G. (ed.), *Proceedings of ECAI'94: 11th European Conference on Artificial Intelligence*. John Wiley, Chichester, pp. 535-540.
- Hunston, S.** (2002). *Corpora in Applied Linguistics*. Cambridge University Press, Cambridge, pp. 1-24.
- Hutchens, J.** (1996). *How to pass the Turing test by cheating*. [Online], <http://ciips.ee.uwa.edu.au/Papers/>, 1996
- Hutchens, T., Alder, M** (1998). *Introducing MegaHAL*. [Online], <http://cnts.uia.ac.be/conl198/pdf/271274hu.pdf>
- Internet.com** (2004). *Search engine*. [Online]: http://www.webopedia.com/TERM/s/search_engine.html
- Johansson, S., Atwell, E., Garside, R. and Leech, G.** (1986). The tagged LOB Corpus. User's manual. The Norwegian Centre for the Humanities, Bergen.
- Jurafsky, D. and Martin, J.** (2000). *Speech and Language Processing: an Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall, New Jersey, pp. 1-19.
- Katz, B.** (1997). From sentence processing to information access on the World Wide Web. In *Natural Language Processing for the World Wide Web: papers from the 1997 AAAI Spring Symposium*, pp. 77-94.
- Kilgarriff, A.** (1997). Putting frequencies in the dictionary. *International Journal of Lexicography*. Vol. 10, No. 2, pp. 135-155.

- King, G. and Lowe, W.** (2003). An Automated information extraction tool for international conflict data with performance as good as human coders: a rare events evaluation design. *International Organization*. Vol. 57, No. 3, pp. 617-642.
- Klein, D. and Manning, C.D.** (2003). Fast exact inference with a factored model for natural language parsing. In Suzanna, B., Sebastian, T., and Klaus, O. (eds.), *Advances in Neural Information Processing Systems 15* (NIPS 2002). MA:MIT Press, Cambridge, pp. 3-10.
- Kosseim, L., Plamondon, L. and Guillemette L.** (2003) Answer formulation for question-answering. In Xiang, Y. and Chaib-draa, B. (eds.), *Lecture Notes in Artificial Intelligence*. No. 2671, Springer, Verlag, pp. 24-34.
- Kranzler, G., and Moursund, J.** (1999). *Statistics for the Terrified*. Prentice-Hall, New Jersey, pp. 74-100.
- Kučera, H. and Francis, W.H.** (1967). *Computational analysis of present-day American English*. Brown University Press, Providence, Rhode Island.
- Kwok, C., Etzioni, O., and S.Weld, D.** (2001). Scaling question answering to the web. *ACM Transactions on Information Systems*. Vol. 19, No.3, pp. 242-262.
- Kytö, M., Rudanko, J. and Smitterberg, E.** (2000). Building a bridge between the present and the past: A corpus of 19th-century English. *ICAME Journal*. Vol. 24, pp. 85-98. [Online]: <http://helmer.aksis.uib.no/icame/ij24/merja.pdf>
- Lee S., Sung C. and Cho S.** (2001). An effective conversational agent with user modeling based on Bayesian network. *Lecture Notes in Computer Science 2198*. Springer, Berlin, pp. 428-432.
- Leech, G., Garside, R. and Atwell, E.** (1983). The automatic grammatical tagging of the LOB corpus. *ICAME Journal*. Vol.7, pp. 13-33.
- Leech, G. and Weisser, M.** (2003). Generic speech act annotation for task-oriented dialogues. In Archer, D., Rayson, P., Wilson, A. and McEnery, T. (eds.), *Proceedings of the Corpus Linguistics 2003 conference (CL2003)*. Lancaster, Lancaster University, UK, pp. 441-446.
- Leech, G.** (1992). Corpora and theories of linguistic performance. In Svartvik, J. (ed.) *Directions in Corpus Linguistics*. Mouton de Gruyter, Berlin, pp. 105-22.

- Lewis, D., and Jones, K.** (1996). Natural language processing for information retrieval. *Communications of the ACM*. Vol. 39, No. 1, pp. 92-101.
- Loebner, H** (1994). In Response to lessons from a restricted Turing Test. *Communications of the ACM*. Vol. 37, No. 6, pp. 79-82. [Online], <http://www.loebner.net/Prizef/In-response.html>
- Loebner, H** (2003). *Home page of the Loebner prize-the first Turing test*. [Online], <http://www.loebner.net/Prizef/loebner-prize.html>
- Loper, E. and Bird, S.** (2002). NLTK: The Natural Language Toolkit. In *Proceeding of ACL Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics*. Association for computational linguistics, Philadelphia.
- Lozano-Pérez, T. and Kaelbling, L.** (2003). Machine learning lecture slides and accompanying transcripts. [Online]: <http://ocw.mit.edu/OcwWeb/Electrical-Engineering-and-Computer-Science/6-034Artificial-IntelligenceSpring2003/Readings/>
- Makins, M.** (ed.). (1991). *Collins English Dictionary*, 3rd ed. Harper Collins.
- Mórquez, L.** (2000). Machine learning and natural language processing. *Seminar. Industrias de la lengua/la ingeniería Lingüística en la sociedad de la información*. University Politècnica de Catalunya, Barcelona.
- Maier, E., Mast, M. and LuperFoy, S.** (1996). Overview. In Maier, E., Mast, M. and LuperFoy, S. (eds.), *Dialogue Processing in Spoken Language Systems*. Springer, Berlin, pp.1-13.
- Mann, W.** (2002). *Dialog Diversity Corpus*. [Online]: <http://www-rcf.usc.edu/~billmann/diversity/DDivers-site.htm>
- McEnery, T. and Wilson, A.** (1996). *Corpus Linguistics*. Edinburgh University Press, Edinburgh, pp. 1-20.
- McEnery, A.M., Wilson, A., Sanchez-Leon, F. and Nieto-Serano, A.** (1997). Multilingual resources for European languages: contributions of the CRATER project. *Literary and Linguistic Computing*. Vol. 12, No. 4, OUP, Oxford, pp. 219-226.
- Mc Kevitt, P.** (2000). The OSCON Operating System Consultant. *Artificial Intelligence Review*. Vol. 14, No. 1-2, pp. 89-119.
- McTear M.** (2002). Spoken dialogue technology: enabling the conversational user interface. *ACM Computing Surveys*. Vol. 34, No. 1, pp. 90-169.

- MICASE.** (2003). *MICASE Manual: The Michigan Corpus of Academic Spoken English*. The English language institute, university of Michigan. [Online]:
http://www.lsa.umich.edu/eli/micase/MICASE_MANUAL.pdf
- Millican, P.** (2002). *Elizabeth's home page*. [Online]:
<http://www.etext.leeds.ac.uk/elizabeth>
- Mishler, E.** (1985). *The discourse of medicine: dialectics of medical interviews*. Ablex, New Jersey. [Online]:
<http://www-rcf.usc.edu/~billmann/diversity/Tr.5.1a.gif>
- Nelson, G.** (2002). *International Corpus of English: the Singapore Corpus user manual*. [Online]:
http://www-rcf.usc.edu/~billmann/diversity/ICE-SIN_Manual.PDF
- Poibeau, T.** (2000). A corpus-based approach to information extraction. *Journal of Applied Systems Studies (JASS)*. Vol. 1, No. 2, pp. 254-267.
- Rapaport, W.** (2004). The Turing test. [Online]:
<http://www.cse.buffalo.edu/~rapaport/Papers/ell2.pdf>
- Rayson, P.** (2003). Matrix: a statistical method and software tool for linguistic analysis through corpus comparison. *Ph.D. thesis*. Lancaster University.
- Renouf, A.** (1987). Corpus development. In Sinclair, J. M. (ed.), *Looking Up: an Account of the COBUILD Project in Lexical Computing*. Collins, London, pp. 1- 40.
- Rich, E. and Knight, K.** (1991). *Artificial Intelligence*. McGraw-Hill, New York, pp. 3-27.
- Ringenberg, M.** (2002). *CIRCLE's tutorial archive*. [Online]:
<http://www.pitt.edu/~circle/Archive.htm>
- Roberts, A.** (2004). *aConCorde*. [online]:
<http://www.comp.leeds.ac.uk/andyr/software/aConCorde/>
- Robin, B., Hammond, K., Kulyukin, V., Lytinen, S., Tomuro, N. and Schoenberg, S.** (1997). Question answering from frequently-asked question files: experiences with the FAQ finder system. *AI magazine*. Vol. 18, No. 2, pp. 57-66.
- Russell, S., and Norvig, P.** (1995). *Artificial Intelligence: a Modern Approach*. Printice Hall, Inc, pp. 691-696.

- Saygin A., Cicekli I. and Akman V.** (2000). Turing test: 50 years latter. *Minds and Machines*. Vol. 10, No. 4, pp. 463-518.
- Scott, M.** (2004). *Wordsmith tools*. [online]:
<http://www.lexically.net/wordsmith/index.html>
- Searle, J.** (1980). Minds, brains, and programs. *The behavioural and Brain Sciences*. Vol. 3, No. 3, pp. 417-457.
- SeniorNet.** (2004a). *Lesson 4*. [Online]:
<http://www.seniornet.org/php/default.php?ClassOrgID=5337&PageID=5920>
- Shieber, S.** (1994). Lessons from a restricted Turing test. *Communications of the Association for Computing Machinery*. Vol. 37, No. 6, pp. 70-78
- Sinclair, J.** (1991). *Corpus, Concordance, Collocation*. UP, Oxford, pp. 1-180.
- Sinclair, J.** (1995). Corpus typology- a framework for classification. In Melchers, G. and Warren, B. (eds.), *Studies in Anglistics*. Almqvist & Wiksell, Stockholm, pp. 17-3.
- Sinclair, J.** (1996). *EAGLES: preliminary recommendations on corpus typology*. [Online]: <http://www.ilc.cnr.it/EAGLES96/corpusstyp/corpusstyp.html>
- Sneiders, E.** (1999). Automated FAQ answering: continued experience with shallow language understanding. In *Question Answering Systems. Papers from the 1999 AAAI Fall Symposium*. AAAI Press, Massachusetts, pp. 97-107.
- Stephanidis, C. and Sfyarakis, M.** (1995). *Current trends in man-machine interfaces*. [Online]:
<http://www.stakes.fi/include/manmac.html#Knowledgebased,%20Intelligent%20User%20Interfaces>
- Stephens, K. R.** (2004). *Non-Loebner conversant systems*. [On line]:
<http://www.behavior.org/computer-modeling/stephens/stephens1.cfm>
- Supekra, K., Patel, C., Singh, S. and Le e, Y.** (2004). *BEE-SMART: A natural language interface for knowledge retrieval and service execution over the semantic web*. [Online]:
<http://www.sce.umkc.edu/~leeyu/Publications/book8.pdf>

- Takahashi, J.** (2003). Do we talk (or write?) differently over the net? – A lexical enquiry into ‘a’ Net-EN-. In Archer, D., Rayson, P., Wilson, A. and McEnery, T. (eds.), *Proceedings of the Corpus Linguistics 2003 conference (CL2003)*. Lancaster, Lancaster University, UK, pp. 764-772.
- Taylor, S.** (2004). Information extraction tools: deciphering human language. *IEEE Xplore*. Vol. 6, No. 6, pp. 28-34.
- Turing, A.** (1950). Computing machinery and intelligence. *Mind* 49, pp. 433-460.
- Van Rooy, B.** (2002). Transkripsiehandleiding van die Korpus Gesproke Afrikaans. [Transcription Manual of the Corpus Spoken Afrikaans.] Potchefstroom: Potchefstroom University.
- Vrajitoru, D.** (2003). Evolutionary sentence building for chatterbots. In *Proceedings of Genetic and Evolutionary Computation Conference Late Breaking Papers*. Illinois, Chicago, pp. 315-321.
- Wallace, R.** (2003) *The Elements of AIML Style*. ALICE A.I. Foundation.
- Walonick, D.** (2004). *Statistical significance*. [online]: <http://www.statpac.com/surveys/statistical-significance.htm>
- Weizenbaum, J.** (1966). ELIZA-A computer program for the study of natural language communication between man and machine. *Communications of the ACM*. Vol. 10, No. 8, pp. 36-45.
- Weizenbaum J.** (1967). Contextual understanding by computers. *Communications of the ACM*. Vol. 10, No. 8, pp. 474-480.
- Whalen, T.** (2003). *My experience with 1994 Loebner competition*. [Online], <http://hps.elte.hu/~gk/Loebner/story94.htm>
- Whitehead, S.D.** (1995). Auto-FAQ: an experiment in cyberspace leveraging. *Computer Networks and ISDN Systems*. Vol. 28, No., 1-2, pp. 137-146.
- Wilks, Y.** (1997). Information extraction as a core language technology. In Pazienza (ed.), *Lecture notes in artificial intelligence 1299*. Springer (SCIE-97), Berlin, pp. 1-9.
- Wilks, Y.** (1999). Preface. In Wilks, Y. (ed.), *Machine Conversations*. Kluwer, Boston/Dordrecht/London, pp. vii-x.
- Wilks, Y. and Catizone, R.** (2001). Human-Computer Conversation. In Kent, A. (ed.), *Encyclopedia of Library and Information Science*. Vol. 69, Dekker, New York.

- Wilson, A. and Rayson, P.** (1993). Automatic Content Analysis of Spoken Discourse: a report on work in progress. In Souter, C. and Atwell, E. (eds), *Corpus Based Computational Linguistics*. Rodopi, Amsterdam, pp. 215-226.
- Winograd, T.** (1972). *Understanding Natural Language*. Edinburgh University Press, Edinburgh, pp. 1-34.
- Witten, I. and Frank, E.** (1999). *Data Mining: Practical Machine Learning tools and Techniques with Java Implementations*. Kaufmann.
- Zaanen, M., Roberts, A. and Atwell, E.** (2004). A Multilingual Parallel Parsed Corpus as Gold Standard for Grammatical Inference Evaluation. In *Proceedings of the Workshop on The Amazing Utility of Parallel and Comparable Corpora. Fourth International Conference on Language Resources and Evaluation (LREC)*. Lisbon, Portugal, pp. 58-61.
- Zadrozny, W., Budzikowska, M., Chai, J. and Kambhatla, N.** (2000). Natural language dialogue for personalized interaction. *Communications of the ACM*. Vol 43, No. 8, pp. 116-120.
- Zavrel, J., P. Berck, and W. Lavrijssen.** (2000). Information extraction by text classification: corpus mining for features. In *Proceedings of the Workshop Information Extraction Meets Corpus Linguistics*. Athens, Greece.
- Zdenek, S.** (2001). Passing Loebner's Turing test: a case of conflicting discourse functions. *Minds and Machines*. Vol. 11, pp. 53-76.

Appendix A. FAQchat Evaluation Sheet

Evaluation sheet: the School of Computing FAQ chatbot

I am Bayan Abu Shawar, a PhD student in the School of Computing at the University of Leeds. The following evaluation is part of my work. We want to investigate the possibility of using a chatbot to access the FAQ of the School of Computing. The evaluation process is based on user satisfaction in using this tool compared with the Google search engine to answer a question related to the FAQ. User satisfaction is measured by two metrics: the ease of use, and the accuracy in giving the right answers. Below are a number of questions that you can try to answer, which will not take more than 20 minutes.

Input your question in the ‘Question’ box and press the “Ask” button. The answer from both the chatbot and the Google search-engine will appear on the screen. You can use the following URL:

<http://www.comp.leeds.ac.uk/cgi-bin/bshawar/faq.cgi>

Name:

Level of education: Staff
Undergraduate

Postgraduate

Signature:

Your task is to try using our system to find answers to typical questions asked by School of Computing students and staff. In case you cannot find an answer, check the spelling, and try to re-formulate the question.

In the table below, state whether you were able to find an answer using the chatbot responses; whether you found an answer using the Google responses; which of the two you preferred; and why.

Question	Finding answer by chatbot (Y/N)	Finding answer by Google (Y/N)	Which one do you prefer? 1. Chatbot 2. Google 3. Both are adequate	Explain the reason(s) for your preference
1. Can you find out how to convert a word document to PDF format?				
2. Can you find out what scanners are in the school?				
3. Can you find the most interesting module(s) for the third year?				
4. Can you find out the opening and closing time of the labs?				
5. Can you find out how to set up a personal website?				
6. Can you find out how to use Linux?				
7. Can you find when and where can you submit your coursework?				
8. Can you find out what to do if you did not attend an exam?				
9. Can you find out what to do if you have problems with your supervisor?				

10. Can you find out how to delete a job from the printer?				
11. Can you find out what students can do after graduation?				
12. Can you find out about available tools for knowledge discovery?				
13. Can you find out how to write a PhD thesis?				
14. Can you find the E-mail address of a member staff?				
15. Can you find out how to print an email message from Pine?				
Below is some space in case you wish to try out (and evaluate) any other questions.				
16.				
17.				
18.				
19.				
20.				

21. Which tool do you prefer overall?	Chatbot	Google	Both	
22. List below any comments, feedback or suggestions				

Many thanks for your cooperation,

Bayan Abu Shawar
School of Computing
University of Leeds
Tel: (0113) 34 37288
E_mail: bshawar@comp.leed.ac.uk