ANEESAH: A Novel Methodology and Algorithms for Sustained Dialogues and Query Refinement in Natural Language Interfaces to Databases

Khurim Shabaz

A thesis submitted in partial fulfilment of the requirements of the Manchester Metropolitan University for the degree of Doctor of Philosophy

School of Computing, Mathematics and Digital Technology

the Manchester Metropolitan University

June 2017

Abstract

This thesis presents the research undertaken to develop a novel approach towards the development of a text-based Conversational Natural Language Interface to Databases, known as ANEESAH. Natural Language Interfaces to Databases (NLIDBs) are computer applications, which replace the requirement for an end user to commission a skilled programmer to query a database by using natural language. The aim of the proposed research is to investigate the use of a Natural Language Interface to Database (NLIDB) capable of conversing with users to automate the query formulation process for database information retrieval. Historical challenges and limitations have prevented the wider use of NLIDB applications in real-life environments. The challenges relevant to the scope of proposed research include the absence of flexible conversation between NLIDB applications and users, automated database query building from multiple dialogues and flexibility to sustain dialogues for information refinement. The areas of research explored include; NLIDBs, conversational agents (CAs), natural language processing (NLP) techniques, artificial intelligence (AI), knowledge engineering, and relational databases.

Current NLIDBs do not have conversational abilities to sustain dialogues, especially with regards to information required for dynamic query formulation. A novel approach, ANEESAH is introduced to deal with these challenges. ANEESAH was developed to allow users to communicate using natural language to retrieve information from a relational database. ANEESAH can interact with the users conversationally and sustain dialogues to automate the query formulation and information refinement process. The research and development of ANEESAH steered the engineering of several novel NLIDB components such as a CA implemented NLIDB framework, a rule-based CA that combines pattern matching and sentence similarity techniques, algorithms to engage users in conversation and support sustained dialogues for information refinement. Additional components of the proposed framework include a novel SQL query engine for the dynamic formulation of queries to extract database information and perform querying the query operations to support the information refinement.

Furthermore, a generic evaluation methodology combining subjective and objective measures was introduced to evaluate the implemented conversational NLIDB framework. Empirical end user evaluation was also used to validate the components of the implemented framework. The evaluation results demonstrated ANEESAH produced the desired database information for users over a set of test scenarios. The evaluation results also revealed that the proposed framework components can overcome the challenges of sustaining dialogues, information refinement and querying the query operations.

List of Figures

List of Tables

List of Equations

List of Appendices

Dedication

الرَّحِيْمِ الرَّحْمٰنِ اللهِ بِسْمِ

"In the name of God, the Most Gracious, the Most Merciful"

This thesis is dedicated to my mother (deceased) and father. Both of whom are like a candle – It consumes itself to light the way for others.

Acknowledgements

I must express my very profound gratitude to my parents (Mohammed Aslam and Ghulam Safia), my wife (Sadia Noreen), my two daughters (Aneesah and Afifa) and other family members for providing me with unfailing support and continuous encouragement throughout the research process. This accomplishment would not have been possible without them.

I would also like to acknowledge Dr James O'Shea, Dr Keeley Crockett and Dr Annabel Latham of the Manchester Metropolitan University as the supervisory team for this research, and I am gratefully indebted to them for their very valuable support, guidance and help throughout the research.

Thank you.

List of Publications

SHABAZ, K., O'SHEA, J. D., CROCKETT, K. A. & LATHAM, A. ANEESAH: A Conversational Natural Language Interface to Databases. Proceedings of the World Congress on Engineering, 2015.

(Appendix H)

List of Abbreviations

AI	Artificial Intelligence
NLIDB	Natural Language Interface to Database
CA	Conversational Agent
CM	Conversation Manager
DOMAIN	Sample Database/Dataset Used for Evaluation
FAQ	Frequently Asked Question
GC	General Chat
GQM	Goal Question Metric
GUI	Graphical User Interface
KB	Knowledge Base
NLP	Natural Language Processing
PARADISE	Paradigm for Dialogue System Evaluation
PM	Pattern Matching
STS	Short Text Similarity
XML	Extensible Mark-up Language
AIML	Artificial Intelligence Mark-up Language
NLI	Natural Language Interface
System	ANEESAH NLIDB
ERP	Enterprise Resource Planning

Chapter 1 - Introduction

1.1 Introduction

This thesis presents research investigating whether a Natural Language Interface to Database can mimic a human query expert by conversationally interacting with users, and formulate queries to extract and refine desired database information. The research entails a thorough analysis of Natural Language Interfaces to Database (NLIDBs) and Conversational Agents (CAs) as well as inherent challenges involved in implementing conversational NLIDBs such as social adaptability, sustained dialogues, information refinement and querying the query operations. The research has led to the development a novel conversational Natural Language Interface to Database, called ANEESAH. The ANEESAH NLIDB is designed to model a database expert by directing the conversation and translating the user's requirements into database query language (e.g. Structured Query Language). The architecture of ANEESAH comprises several new components, which have been specifically developed to address the unique challenges involved in implementing NLIDBs. This chapter provides the background and motivation of the proposed research, aims and objectives, along with a summary of the research contributions, and brief description of thesis structure.

1.2 Background

Information has its fundamental importance in decision making at any level. The largest sources of information storages are databases in public or commercial environments. The databases servers are constantly evolving with not only information but in terms of complexities of structures and designs (Hamaz and Benchikha, 2017). Retrieving information from a database normally requires querying the database using specialised programming code not accessible to the inexperienced users, known as structured query language (SQL). There exists a need for creating computer applications that permit inexperienced users to extract desired information stored in a database (Yaghmazadeh et al., 2017). Natural Language Interfaces to Databases (NLIDBs) are computer programmes that replace the requirement for an end user to commission a skilled programmer to query a database by using natural language (Pazos R et al., 2013; O'Shea et al., 2011).

The primary focus of NLIDB development has been intended to process a single query response transaction. The conversational capabilities of NLIBDs have received little attention in research, among other aspects such as social adaptability and sustained interaction to elicit what an end user envisages about the domain (Owda et al., 2007; Androutsopoulos et al., 1995). NLIDB development attempts span decades. However there remain unsolved key challenges for their wider acceptance in public and commercial environments (Pazos R et al., 2013). Despite many development approaches, NLIDBs have failed to achieve 100% user satisfaction. There are a number of challenges have been identified by researchers such as linguistic problems, conversational abilities, query translation, information refinement, domain independence and ease of configuration. Also, the lack of generally accepted evaluation benchmarks that can be used to test effectiveness and reliability of NLIDBs adds to the existing challenges and those of building similar applications (Castillo et al., 2014; O'Shea et al., 2011).

The motivation for this research came from the need for a conversational NLIDB that could mimic a human structured query language expert by employing conversation with users to dynamically formulate queries to extract and refine database information. The research aim, questions, hypothesis and objectives are discussed as follows.

1.3 Research Aim

The main aim of this research is to contribute to the understanding of existing NLIDBs by developing a novel conversational NLIDB architecture for sustained dialogues to automate query formulation and information refinement processes and perform an evaluation of the implemented prototype through a sample database. The following research questions are investigated as part of this research.

1.4 Research Questions

The research questions are as following:

1. Can a NLIDB allow users to retrieve desired information from a database conversationally?
2. Can a NLIDB allow users to engage in sustained dialogues to refine query produced information from a database?
3. Can a Pattern Matching approach be used to successfully develop a conversational NLIDB, capable of automating complex query formulation process?
4. Can a conversational NLIDB generate comparable results to those produced conventionally by a database expert?

1.5 Research Hypothesis

The null hypothesis (**H0**) is that a general user cannot interact with a NLIDB to formulate a query to retrieve and refine desired information from a relational database.

The (**H1**) hypothesis is that a general user can interact with a NLIDB to formulate a query to retrieve and refine desired information from a relational database, successfully.

1.6 Research Objectives

The objectives of this research are:

1. Investigate state of the art (SOA) on existing NLIDBs and CAs, based on approaches and architectures adopted for developments.
2. Research and identify historical challenges that have prevented progress and wider adaptability of NLIDBs in real-life environments.
3. Specify a methodology for the development and implementation of a novel NLIDB framework, scripting language and evaluation framework to measure features implemented in the proposed prototype system.
4. Review and investigate knowledge engineering techniques to generate and implement a NLIDB knowledge base.

5. Build a conversational NLIDB architecture with CA components and conversational abilities to allow users to extract and refine information stored in a database.
6. Design an appropriate evaluation methodology for the proposed conversational NLIDB, and determine its ability to handle conversation with the users, usability and usefulness (such as user experience, user acceptance, information accuracy, reliability) through specific experiments.

Figure 1.1 illustrates each objective of the proposed research and how it is addressed in this thesis. An overview of each chapter is detailed below thesis structure section (1.8).

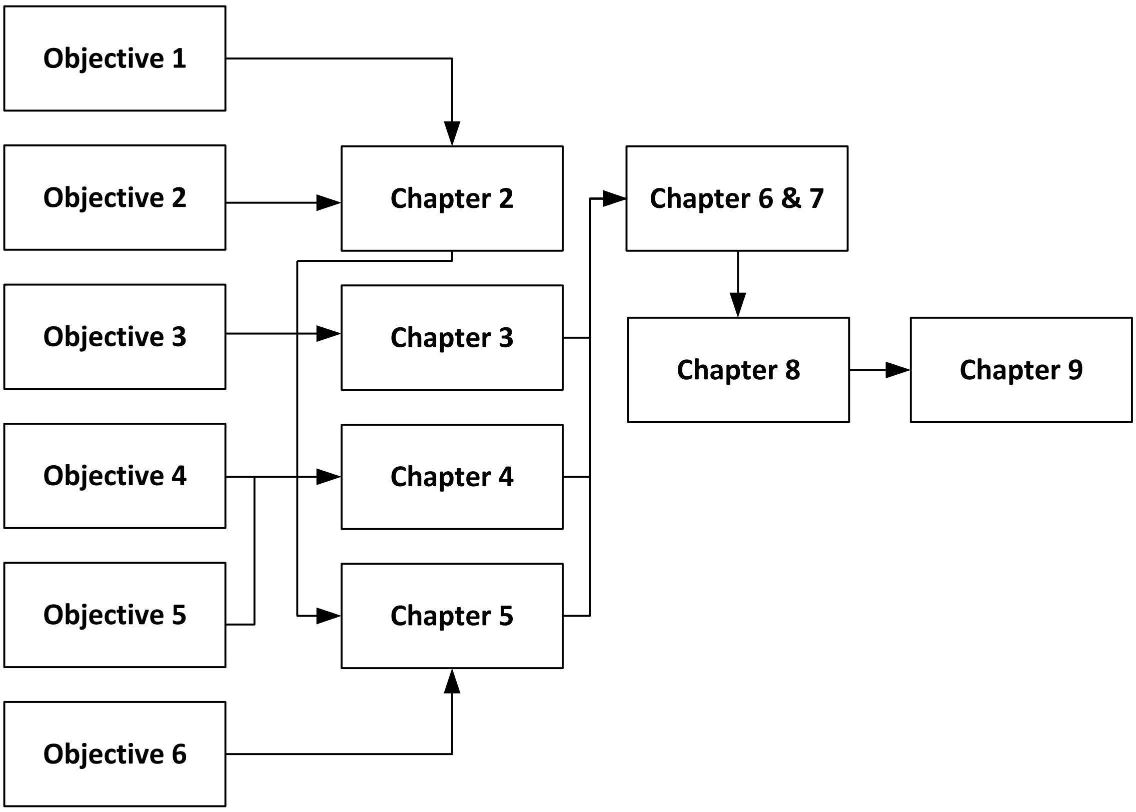


Figure 1.1: Thesis chapters in relation to the research objectives

1.7 Contributions

The most significant contributions of this research are:

- A novel architecture for a conversational NLIDB (ANEESAH) and a generic development methodology for creating similar NLIDBs for other databases/domains.
- A new scripting methodology designed specifically to allow fine control of the conversation, dynamic query formulation and information refinement during the scripting process.
- Proof of concept that it is possible for an inexperienced user to conversationally interact with a NLIDB to dynamically formulate a query to retrieve desired information stored in a database.
- Proof of concept that a conversational NLIDB can engage users in sustained dialogues to further refine query produced information from a database.
- A novel evaluation methodology that can be utilised to perform an evaluation of similar conversation-based NLIDBs from the subjective and objective perspectives.
- Development of ANEESAH prototype one and results from empirical studies that validate the generic architecture and methodology for a NLIDB with conversational abilities to provide an interactive and friendly environment to assist users in extracting desired database information.
- Development of ANEESAH prototype two and results from empirical studies on the further developed and modified architecture and improved methodology as well as enhanced conversational abilities to sustain dialogues for information refinement and dynamic querying the query operations (i.e. reformulation of an existing query to reflect different/refined results etc.)

The contributions listed above are expected to be of value to researchers and developers in the fields of CAs and NLIDBs. Researchers and developers can utilise these contributions as a starting point for their future projects/research. The proposed methodology and architecture can be used as a foundation to build conversational NLIDBs, which can conversationally engage users to extract desired information stored in a database by dynamic query formulation process as well as sustain dialogues to perform further information refinement.

1.8 Thesis Outline

The nature of the research and development of the ANEESAH conversational NLIDB resulted in the substantial amount of documentation and experiment work. Chapter 2 provides a current state of the art of NLIDBs and CAs, the background of the field, different approaches to building similar applications, challenges and limitations relevant to the scope of this research followed by a review of existing conversational NLIDBs.

Chapter 3 presents the development methodology adopted for creating a conversational ANEESAH NLIDB. The development methodology has been described in three phases. Phase one will give an overview of the proposed ANEESAH NLIDB prototype and details on the development of different components. Phase two of the development methodology will highlight the development of scripting methodology and knowledge base. Subsequently, phase three will also provide details on a generic architecture for ANEESAH NLIDB.

Chapter 4 of this thesis presents an implementation of ANEESAH conversational NLIDB prototype.

Chapter 5 presents details on evaluation methodology adopted to evaluate the developed prototype ANEESAH NLIDB, empirical results and statistical tests undertaken to answer research hypothesis and research questions. Chapter 5 will also highlight experimental results based on users' interaction with the developed ANEESAH NLIDB, followed by a survey/questionnaire-based evaluation data highlighting whether the developed prototype was comparable to a human expert with respect to assisting users in task completion. Chapter 5 also provides a detailed discussion of experiments results from an evaluation of the initial prototype initial ANEESAH NLIDB prototype. This chapter also highlights different aspects of the initial prototype's architecture, areas highlighted for improvement (such as spelling mistakes), robustness and information accuracy measures recorded during evaluation. Chapter 5 also bring to light gaps (such as sustained dialogues for information refinement, querying the query operations) leading to the further development of an initial prototype, to evaluate and answer all research questions.

Chapter 6 presents further research and development conducted to improve and strengthen ANEESAH’s architecture to address the weaknesses and areas of improvements outlined through the first stage evaluation. Chapter 6 also provides details on modifications made to the existing architecture (of the initial prototype ANEESAH NLIDB), and new components developed to strengthen its abilities further and overcome shortcomings noted through first stage evaluation.

Chapter 7 reflects on the evaluation methodology and results gathered during the evaluation of ANEESAH prototype two. Chapter 7 also presents a statistical analysis of empirical results to determine if enhancement and further development have led to improvements in ANEESAH’s abilities such as conversational control, query refinement, weaknesses highlighted during first phase evaluation.

Chapter 8 highlights the conclusions drawn from findings, contributions, discussion and comparison of results to evaluate the effectiveness of both prototypes, and describe its limitations and provide recommendations for the direction of future work.

Chapter 2 - State of the Art

2.1 Introduction

The idea of a computer taking the role of a human in a conversation was first proposed by Alan Turing (Turing, 1950). This idea led to the proposal of Turing's test which spurred the research community to develop prototypes that could mimic humans to fool a judge. The research attempts to pass Turing's test appeared in the form of computer programs called chatterbots, which used tricks to create the delusion of intelligence during a conversation. Adding goal-oriented intelligence to chatterbots led to a new generation of conversational partners referred to as "Conversational Agents" (Weizenbaum, 1966). Conversational Agents (CAs) are computer applications that enable people to communicate with computers using natural language (Russell et al., 1995). The term CA comprises a description of different types of CA applications such as text-based, embodied or spoken systems. The implementation of CAs in various domains assist users to achieve their goal easily and rapidly (O'Shea et al., 2011).

Natural Language Interfaces to Databases (NLIDBs) are computer programs, which replace the requirement to react with a skilled programmer to query a database by using natural language. This chapter represents a review of NLIDBs both historical and the current state of the art and contrasts the strengths and weaknesses of existing approaches. Finally, a review of existing NLIDB and CA building techniques, the need for alternative approaches and evaluation methodologies adopted for testing are also discussed in this chapter.

2.2 Natural Language Interfaces to Databases

Natural Language Interfaces to Databases (NLIDBs) are computer applications, which replace the requirement for an end user to commission a skilled programmer to query a database by using natural language. Among early development of NLIDBs, the most popular NLIDB was LUNAR (Woods, 1972). LUNAR was built to perform moon rocks analysis based on an underlying database, but it had functional limitations and could not be generalised to other domains (Woods, 1973). Later, the development of RENDEZVOUS was intended to simulate open dialogues to users in order to formulate database queries. RENDEZVOUS required user anticipated inputs to closely match its knowledge base text for the system to understand the entered text, for processing. LADDER development (Hendrix et al., 1978) was developed to target big data and distributed databases. However, LADDER NLIDB required substantial customisation (e.g. new grammar, domain knowledge) to work with new domains. PLANES designed features were based on the principals of RENDEZVOUS, which used flights and an airport database to answer users' questions (Waltz, 1978; Walts, 1975). PLANES NLIDB also required new grammar and extensive customisation to work with new domains.

In the 80s, research on NLIDBs increased with the main focus on portability and interface designs (Owda et al., 2007). Several NLIDB systems were developed by this time such as CHAT80, which translated natural language in Prolog language and TEAM, which translated natural language queries into Simple Object Database Access query language (Warren and Pereira, 1982; Grosz et al., 1987). Other systems such as PARLANCE (Bates, 1989), were built to resolve domain configuration issues and allow users to configure underlying domains manually. ASK also appeared as a cross application NLIDB that allowed users to input their requests in natural language to generate appropriate responses from the underlying database (Thompson and Thompson, 1985; Thompson and Thompson, 1983).

In the 90s, fewer NLIDB systems made their way to the commercial market with a primary development focus on learning approaches. Despite the lack of acceptance in real-life environments, work on NLIDBs continued to evolve with research on various systems such as CHILL. This was built to analyse the implications of Inductive Logic Programming, which comprised of paired questions with respective parsing (Zelle and Mooney, 1996). INTELLECT's entry in the commercial market was perceived as a motivational step to amplify development of similar NLIDB systems. Some buyable NLIDB options in the market were IBM's Language Access, Q&A Symantec, DATALINKER, and LOQUI from IBM English Wizard (Ott, 1992; Sijtsma and Zweekhorst, 1993).

More recently, the development of NLIDBs continues to evolve with the implementation of advanced technologies in the Natural Language Processing field, integrating language and graphics to take the benefits available from all modalities (Revuelta-Martínez et al., 2013). In recent years some online system also appeared such as Wolfram Alpha, Powerset and TrueKnowledge(Lopez et al., 2012). These were designed to rely and work on initial information imported during their development stage. Recent developments show further efforts to create limited interaction between the user and NLIDBs. The ITG system was developed to offer train ticket information. However, it urged users to repeatedly validate predictive text generated based on their inputs (Revuelta-Martínez et al., 2013).

PRECISE was developed to address issues such as natural language translation and overcome parser errors (Popescu et al., 2004). GeoDialogue was developed to handle conversational grounding and dialogue generation challenges (Cai et al., 2005). NLPQC was implemented to work through templates for translating natural language inputs into queries for a relational database system. NLPQC also employed WordNet feature that would generate from the database schema (Stratica et al., 2005a). C-Phrase was developed on Codd's tuple calculus to allow context-free grammar (Mooney, 2006). NaLIX implementation focused on serving as a search tool to query the web-based datasource (Li et al., 2007).

NaLIR a keyword-based search interface for web developed using Natural Language Processing (Li and Jagadish, 2014). Other developments include NL2CM, which was built to provide an interface that translates the user inputs into the formal query languages that covered mining platforms support (Amsterdamer et al., 2015). ATHANA translated user requests into an intermediate query language over the ontology and later translated them into database queries (Saha et al., 2016).

2.2.1 NLIDB Development Challenges and Limitations

According to (Church and Patil, 1982), development of Natural Language Interfaces (NLIs) for information retrieval from structured data requires a deep understanding of different factors e.g. Natural Language complexities, ambiguities, etc. Some architectural techniques have been adopted to use limited natural language to generate logical queries for structured data. Such techniques include the Syntax-Based Family of

Architectures (Woods, 1972), the Semantic-Grammar Family of Architectures and Pattern Matching (Cui et al., 2007).

2.2.1.1 Syntax-based Approach

Syntax-based systems utilise syntactic parsers to process a user utterance and produce a corresponding to natural language query with the help of generated parsed tree, often referred as a constituent tree (Androutsopoulos et al., 1995). In Syntax-based systems, a user utterance is parsed and analysed syntactically to formulate a relevant query for a database. The LUNAR system is an example NLIDB built using the Syntax-based approach (Woods, 1972). A language specific parser generates a constituent tree in a shape of a syntactic representation based on the user utterance. The parser extracts phrases and words from the user utterance and uses a set of rules to create a relationship between phrases and words. Later, the constituent tree is translated to the database query language (e.g. SQL) with the help of designed rules. The constituent tree contains deep information to formulate a query (Pazos R. et al., 2013).

The syntax-based approach has several problems for its use in natural language interface developments. The NLI systems designed with the Syntax-based approach are dependent on grammatically correct utterances and with correct structure. Poorly structured sentences can lead to system failures (O'Shea, et al., 2011). Domain independence is also a concern highlighted for systems developed with the Syntax-based approach. The possibility of multiple syntactic trees which create various interpretations of user utterances is also highlighted as one of the major problems in adopting this approach (Owda et al., 2011). Defining mapping rules that transform users' utterances to database queries is often difficult (Androutsopoulos et al., 1995), and portability of syntax-based systems is also difficult due to in-depth designed (domain specific) syntactic structures (Pazos R. et al., 2013).

2.2.1.2 Semantic Grammar Approach

Semantic grammar systems (Karande and Patil, 2009) work on a similar principle to Syntax-based systems. The users' utterances are parsed to generate a constituent tree.

However, the difference in this technique is that it uses pre-defined grammar categories for mapping the constituent tree to a SQL query. In semantic grammar based systems, unlike Syntax-based systems, pre-defined grammar categories do not necessarily relate to syntactic concepts but correspond to domain knowledge and help to enforce semantic constraints (Androutsopoulos et al., 1995). The NLIDB systems developed using semantic grammar architecture tend to process user inputs with less complex constituent trees, when comparing to conventional Syntax-based NLIDBs. The semantic grammar approach also makes NLIDB systems flexible in allowing assignment of semantic information to the tree nodes, which reduces elliptical problems during query formulation process. PLANES, LADDER, REL, PRECISE, NLPQC, WYSIWYM are some example semantic grammar-based systems (Karande and Patil, 2009).

The main disadvantage of using this technique is portability, due to its reliance on domain specific knowledge, which is hardwired in the form of semantic grammar. A new semantic grammar is required to configure a given system to work on a different domain. Moreover, the syntactic tree developed using this approach cannot be adapted to other databases. The NLIDB systems developed using semantic grammar approach mostly relied on a corpus of query templates manually created by developers. This approach has been adopted more recently in a system called NLDBI developed by (Rao et al., 2010; Androutsopoulos et al., 1995).

2.2.1.3 Pattern Matching Approach

Some of the early NLIDBs relied on the pattern matching approach (Pazos, et al., 2013). The pattern matching approach is based on a method, which explores all matched occurrences of scripted patterns against user utterance. This approach is also described as the act of evaluation for an input sequence of tokens for the presence of constituents of some pattern. In contrast to pattern recognition, the match usually has to be exact (Liapis, 2013). The patterns are formed in either a tree structure or sequences (Fader et al., 2013). In this pattern matching approach, a user utterance is not required to be grammatically correct, as the pattern matching technique works on a different principle from that of syntax-based approach. Furthermore, the wildcard matching method parses a user utterance to yield matched words and relationships to formulate a ruled based response. The pattern matching approach can be adopted in the development of Chatbots, CAs and NLIDBs based on precise methodologies (Crockett et al., 2009).

The pattern matching approach has shown effectiveness and flexibility to develop extended dialogue applications (O'Shea et al., 2011; Pazos R. et al., 2013). SAVVY (Johnson, 1984), InfoChat (ConvAgent, 2001) are examples of NLIDB systems which employ the pattern matching approach. This approach using a rule-based matching algorithm produces controlled responses and offers flexibility to sustain dialogues with users (ConvAgent, 2005;Crockett et al., 2009). The NLI-RDB system employed the pattern matching approach for its development, discussed in section 2.4 (Owda et al., 2007). According to (Kerry et al., 2009), the pattern matching approach has revealed impressive results in systems with clearly defined domains. However, this approach due to its shallowness can lead to system failures. The pattern scripting is a laborious and time-consuming task (discussed in more detail below).

2.2.2 Datasets used for NLIDBs Evaluation

There exists no agreement on what sample databases (datasets) should be used as evaluation benchmarks for building and testing NLIDB systems. Most NLIDB developments have relied on three main example datasets with sample records namely; jobs domain dataset with jobs related information, a restaurant information dataset and a geo-base dataset containing locational information. Inherently, these datasets are simple due to basis or non-relational structure and contain fewer records. Therefore, the selection of one or more of these datasets often required researchers to modify its structure in order to evaluate their NLIDB applications (Tang and Mooney, 2001). Historically, researchers have also used custom created datasets to evaluate their NLIDB applications. Table 2.1 gives an overview of few example datasets used for NLIDB developments and evaluation, in the past decade.

Prototype System	Author(s)	Development Approach	Database Used	Accuracy Recorded
NL Query		Custom/Domain ontology	Custom	89%
NLI-RDB (2)	(Alghamdi et al., 2017)	PM	Custom (Unknown)	Unknown
NaLIR	(Li and Jagadish, 2014)	NLP	Microsoft Academic Search database	89.79%
CPC-NLIDB	(Akula et al., 2013)	Semantic Grammar	Unknown Database	96.6%
ITS	(Revuelta-Martínez et al., 2013)	NLP	Trains Database (Unknown)	80%
AskMe	(Llopis and Ferrández, 2013)	NLP	A subset of Northwind Database	94.8%
PNLIDB	(Kaur and Bhatia, 2010)	PM	Generic Agriculture (Unknown)	Unknown
GINLIDB	(Faraj et al., 2009)	NLP	Generic Employees Database (Unknown)	Unknown
NLI-RDB (1)	(Owda et al., 2007)	Knowledge Trees - PM	Generic Sales Database (Unknown)	Unknown
NaLIX	(Li et al., 2007)	NLP	Timber XML Database	Unknown

Table 2.1: Sample datasets used for NLIDBs evaluation

Some NLIDB systems have been evaluated using Northwind Traders' dataset (Microsoft.com, 2017), Pubs Books dataset or the CINDI library dataset (Stratica et al., 2005b) that either have a simple structure or fewer example records. Moreover, the ATIS dataset with airline flights information is regarded as most complex in a relational structure comprising 27 tables and 123 columns. There are only a few NLIDBs evaluated with ATIS dataset or similar databases due to elliptical and complexity of structure (Pazos R. et al., 2013; Castillo et al., 2014). The selection of an appropriate dataset is fundamental to the testing and evaluation of a NLIDB capable of conversationally interacting with users to automate query formulation process and allowing access to desired information with query refinement abilities.

2.2.3 Current known weaknesses in the field of NLIDBs

There have been many attempts to develop NLIDB systems in the past decades. However, these attempts have yet not received a wider acceptance in real-life environments. There are several factors involved leading to the lack of acceptance and widespread use of NLIDB systems in real-life environments.

2.2.3.1 Linguistic Coverage

The linguistic coverage problem has been one of the major weaknesses, as users are often ignorant of the linguistics abilities of NLIDBs (Cohen, 1992). In real-life environments, it is not practical for users to remember what questions an NLIDB system can or cannot cope with/handle. An NLIDB system will only give a definitive answer after it has understood user utterance (Ramasubramanian and Kannan, 2004).

2.2.3.2 Domain Coverage Failure

In the case of NLIDB's failure to answer user question, it is often not clear for users to determine whether the system failure was caused due to linguistic limitations or domain coverage (Copestake and Jones, 1990). The users in this situation try to rephrase their utterances to make the system understand, staying unknown of the actual problem. In some cases, a few NLIDBs respond through error messages (i.e. unknown answer, unknown syntax, etc.), which adds to the lacking abilities of such systems (Carbonnell et al., 1982; Crockett et al., 2011).

2.2.3.3 Users Assumption of System's Intelligence

The users' assumption of system intelligence is also among factors associated with the weaknesses of NLIDBs. The users naturally assume NLIDBs to be intelligent, sensible and active enough to understand and extract facts from their utterances. However, the existing NLIDBs lack conversational and reasoning abilities, which ultimately adds to the disadvantages of NLIDBs from a user perspective (Hendrix et al., 1978; Crockett et al., 2013).

2.2.3.4 Interface Problems

The use of natural language in communicating with computers has been disputed by some researchers, who have regarded its use as inappropriate. The basis for these arguments is related to the inability of computer applications to understand and cope with user's requirements when using natural language. The users are often displeased by having to express their requests formally, with correct grammar, and in short sentences, etc. (Binot et al., 1991).

2.2.3.5 Configuration and Maintenance

One of the major factors hindering the acceptance of NLIDBs is referred to post implemented configuration and maintenance. Most commercially available NLIDBs were later cancelled from parent companies purely due to the issues related configuration, maintenance and portability to a different environment (O'Shea et al., 2011; Pazos R. et al., 2013).

2.2.4 Challenges for NLIDBs

The development of Natural Language Interfaces has been around for over 50 years, but to date, these systems are not in wider use. There are a number of associated factors which led to discouraging the industry from accepting NLIDB systems as useful real-life tools (Pazos R et al., 2013). Amongst note able discouraging factors are the unsolved issues and weaknesses in developed systems, as researchers who worked in this area, did not further improve their prototypes. Some of the common challenges and concerns identified by most NLIDB researchers are: conversational/linguistic problems, domain independence issues, poor translation processes (database query formulation), poor result refinement, multimodality issues and ease of configuration (Pazos R. et al., 2013; O'Shea et al., 2011). In line with the scope of this research, challenges can be summarized below:

Lack of conversational abilities in NLIDBs has been outlined as a frequent problem by the users (Carbonnell et al., 1982; Tennant et al., 1983; Cohen, 1992). For users to remember or memorise what kind of questions a NLID system can answer or cannot answer is not ideal. In the case where a NLIDB fails to understand user requirement, it is often difficult for that user to judge the reason behind system failure (i.e. scope of the system, system abilities or coverage of domain, etc.). In complex and distributed databases, many NLIDB systems have revealed ellipsis problems. Selection of conceptual models in constructing NLIDBs is also highlighted as an issue for linguistic problems (Tennant et al., 1983). Other problems such as anaphora (i.e. repetition/mentioning of same word/term in a one sentence etc), grammatical utterances, quantifier scoping are some of the linguistic problems that a NLIDB has to tackle when attempting to interpret a user utterance. The query translation process has also been described as one of the major aspects of an NLDB (Androutsopoulos et al., 1995).

The query translation process undertakes the understanding of natural language in contrast to syntax and query. Query formulation problems that originate are relevant to the adopted NLIDB development approach. Major problems confronting NLIDBs in the formulation of queries can be defined as semantic ellipsis and wrong words or phrases (i.e. missing key information, adjectives, verbs prepositions), coverage capabilities of SQL such as several tables, aggregative functions, excessive information and user errors (Androutsopoulos et al., 1995). The results produced by NLIDBs can contain encoded information such as department identification number instead of department name (Pazos R. et al., 2013; Binot et al., 1991).

There are a number of other challenging areas that are relative to the lack of advancement of NLIDB technology. These areas can be described namely; achieving high accuracy rates with domain independent architecture, portability of knowledge domain and underlying database to work in a different environment, the ability to read and explore big data in real-time (Giordani and Moschitti, 2009; Pazos R. et al., 2013).

2.2.5 Existing Methods of Evaluating NLIDB

There has been a substantial amount of work done on the evaluation of NLIDBs. However, unlike other mature areas of research, evaluating NLIDBs is regarded as a challenging task due to lack of generally accepted evaluation frameworks. The lack of evaluation standards are considered to be one of the main problems that have prevented the progress of NLIDBs. Many researchers have relied on evaluation metrics such as precision and recall to determine the accuracy and performance of their NLIDBs (Castillo et al., 2014; Sujatha and Raju, 2016). Equation 2.1, Equation 2.2 and Equation 2.3 show accuracy, recall and precision equations used for NLIDBs evaluation:

$$\text{Accuracy} = \frac{\textit{Total number of correct queries generated by NLIDB}}{\textit{Total number of queries parsed/attempted/Failure}} \quad X \quad 100$$

Equation 2.1: Accuracy Equation (Castillo et al., 2014; Sujatha and Raju, 2016)

$$\text{Recall} = \frac{\textit{number of correct system answers (with excessive info)}}{\textit{number of gold standard/correct answers}} \quad X \quad 100$$

Equation 2.2: Recall Equation (Lopez et al., 2013)

$$\text{Precision} = \frac{\textit{number of correct system answers}}{\textit{number of system answers}} \quad X \quad 100$$

Equation 2.3: Precision Equation (Lopez et al., 2013)

Equations (2.1, 2.2 and 2.3) make use of correct queries produced by NLIDBs, in determining the appropriate measured values. The accuracy equation is the percentage of correctly formulated and executed queries with respect to the total number of queries including queries with inadequate and excessive results. The recall equation measures are based on correctly formulated/executed queries with respect to the total number of queries attempted including incorrect or failed queries, incomplete or inadequate results. There are other evaluation metrics that can be used to determine the performance of NLIDB e.g. Precision, F measure etc. (Castillo et al., 2014).

The F-Measure value combines the metric of Recall and Precision. This measuring technique has been widely used to evaluate NLIDB systems for their accuracy (Lopez et al., 2013; Srirampur et al., 2014). Following is the equation Eq. (2.4) for F-Measure calculation.

$$f = \frac{\mathbf{2} \times \boldsymbol{Precision} \times \boldsymbol{Recall}}{\boldsymbol{Precision} + \boldsymbol{Recall}}$$

Equation 2.3: F-Measure Equation (Lopez et al., 2013)

Unfortunately, the definition of "correct query" lacks uniformity as some researchers consider a query containing requested results only, as a correct or an ideal query. Others have regarded, a query that contains results in excess of requested results, as a correct query (Castillo et al., 2014). Considering all aspects of evaluation is important in relation to the expected outcome from NLIDB. Comparing performance of different NLIDBs is difficult, as there is no uniformity on what benchmark should be used for evaluation. For a business, the term "correct query results" is very important if NLIDB was to be deployed and used as an information tool in decision-making (Castillo et al., 2014; Pazos R. et al., 2013). Historically, the evaluation of NLIDB has focused on determining conventional measures such as accuracy, precision, recall and F measure, and very little attention has been given to evaluate to subjective metrics e.g. social adoptability, interaction and user experience etc (O'Shea et al., 2011).

Furthermore, research on evaluation of other conversational systems has produced number of models, methodologies and evaluation frameworks, which can be combined with conventional measures (accuracy, precision, recall and F measure) to take a holistic view of a NLIDB. The evaluation metrics can be divided into two categories namely; subjective metric and objective metrics. The subjective metrics help in determining system's overall performance from user satisfaction perspective. Objective metrics are measured by employing a different approach (O'Shea et al., 2011; Forbes-Riley and Litman, 2011).

2.3 Conversational Agents

Conversational Agents (CAs) allow users to communicate with computer systems using natural language. CAs have been implemented in various domains to assist users to achieve their goals easily and rapidly. CAs have the advantage of replacing skilled, expensive human advisors with a consistent 24/7 service (O'Shea et al., 2014). CAs offer the conventional benefits of computer systems. CA applications are available for use at all times and they present consistent advice do not require rest. CAs have been used effectively in many applications, such as web-based guidance and database interfaces (Latham et al., 2010). CAs have been used effectively across many fields i.e. advice & guidance, customer services, computerised learning (Reis et al., 1997). Following are the three main types of CA:

- Text-based CAs were originally intended to hold conversation with users, often referred as chatbots (Carpenter, 2007). Other type of text-based CAs are goal-oriented designed to address specific problems in a specific domain or environment. Goal-oriented textual CAs adopt goal achievement methodology such as “InfoChat” (Michie and Sammut, 2001), supported by an AI decision making component.
- Embodied conversational agents (ECA) are computer-based graphical characters that impersonate represent several properties of humans when engaged in face-to-face conversation. ECA are capable of producing and responding to verbal and non-verbal interaction (Cassell, 2000).
- Spoken dialogue-based CA applications employ speech as communication method for interaction. The possibility of using spoken language is attractive for several reasons as speech is natural method of communication. Spoken dialogue systems are also goal driven (Sadek, 1999).

In the context of this thesis, the term CAs refers to text-based CA systems to separate the signal processing challenges of automatic speech recognition (ASR) from the semantic requirements of NLIDB. The implementation of voice feature to a CA helps in widening access and giving demonstration of more human-like interface. CA applications engage users in conversation by accepting their inputs in natural language and producing an appropriate answer. The answers are usually pre-defined “generic text” that can be dynamically changed with variable information to reflect the conversation context. For example, in a pre-defined greeting response variable information can be used to replace name e.g. Nice to meet you ‘Craig’. However, there are several different techniques to understand user utterances in CAs:

- The natural language processing (NLP) technique attempt to parse/understand user input by analysing constructs, meaning of natural language, sentence structure and by application of rules to process important parts of sentences. NLP understand user input based on a hierarchy of parts-of-sentence that makes this technique dependent only on the sentence's structure rather than on context and domain-specific information. The users' utterances are expected to be grammatically correct that is often not the case. NLP technique also require huge amount of computational power ultimately impacting on scalability and speed for real-time user in different environments such as web, office (Khoury et al., 2008), particularly if repair mechanisms are used to correct grammar etc.
- The pattern matching approach has also been adopted for the development of CAs. As discussed in section 2.2.1.3, rather than attempting to understand the input; this approach utilises an algorithm to match scripted patterns and key words within an utterance to pattern-based stimulus-response pairs. Whilst developing a set of scripts is a laborious and time consuming task, this approach can understand/translate grammatically incorrect and incomplete inputs (Wallace, 2009). For example, the InfoChat , ADAM (Convagent, 2005), OSCAR (Latham et al., 2012) and UMAIR (Kaleem et al., 2014) are few CA applications, built using pattern matching technique.
- Another approach used in CAs to translate and understand user inputs is AI driven, which utilises a semantic similarity measure. Research in semantic similarity measures is at its earlier stage. This method intends to reduce the development time required to build CA applications and efforts of scripting. However, the benefits of this approach are not yet fully realised (O'Shea et al., 2011).

CAs are regarded as good for question and answering systems because of intuitiveness to utilise and enable users to access desired information. However, for an environment requiring sustained dialogues (such as application in database operating environment), expertise and time required to create sophisticated CA scripts that impersonate human conversation is a challenge rarely evaluated.

2.3.1 Pattern-matching Text-based CAs

Most text based CAs rely on the pattern matching approach as it supports sustained dialogues (O'Shea et al., 2011). This approach works by requiring development of conversation scripts. The example of conversation scripts used for pattern matching approach similar to the scripts used in call centre enviornments where key input words and phrases are matched to appropriate responses. The conversation scripts often comprise numerous patterns leading to many stimulus response pairs in the CA's knowledge base. Scripts are initially developed by anticipating user utterances followed by writing response (stimulus) pairs to match them. The development of conversation scripts for a CA is a time consuming and complex task. The maintenance of CA scripts requires continuous improvement by reviewing incorrect CA responses from conversation logs and altering or enhancing stimulus response pairs to address the issues. This is labour intensive, time consuming and requires considerable language experties.

A CA script contains collection of pattern-based stimulus-response pairs called rules. The rules represent current status and a response pattern. A script pattern contains wildcards that are used to match any number of words or characters, increasing the matching probabilities of rules to match utterances containing specific words and key phrases. The pattern matching process does not grammatically correct input or complete sentence structure. However, non-specific/declarative user utterances (e.g. "what do you mean?") remain one of the major challenges. Different conversation histories and topic groups are utilised to help discover appropriate matches i.e. the meaning of an example user utterance "Yes, let me take a look" can only be understood with respect to the previous utterance and current context.

Rule execution/selection (as CA response) is driven through an algorithm, which works through scripts grouped into categories and linked in a tree structure (Sammut, 2001). The grouping of categories is sometimes structured over various levels such as a filter script for capturing abusive words. The efficiency of the matching algorithm and the organisation of the scripts have a direct impact on the real-time use of CAs as interfaces. CA systems built with pattern matching approach can be applied to social (chatbots) or goal-based conversations (e.g. conducting a sales), depending on the development methodology chosen (Crockett et al., 2009).

2.3.2 Background

Most general-purpose CA's were developed with the restrictive knowledge base to engage in shallow conversations only. Nevertheless, ELIZA received the global endorsement, as being the first best Chatbot prototype, which convinced users to believe that it was listening and understanding their inputs. Parry (theparanoind) chatbot which was also convincing because it implemented a narrow domain (Pereira and Coheur, 2013). Among best known early developments were ALICE (Artificial Linguistic Intelligent Computer Entity) and ADAM (Wallace, 2008). These agents were developed and implemented in several fields. ALICE was designed on the same principle as ELIZA to share one built purpose of using questions to draw a conversation out of the user. ALICE and ELIZA were intended to keep the conversation going by asking users common questions (Rzepka and Araki, 2015). ALICE was developed to work with knowledge base stored in Artificial Intelligent Markup Language (AIML). AIML is a pattern scripting language derived from Extensible Markup Language (XML).

The use of AIML helped symbolic reduction in order to analyse user utterances and generate responses. The symbolic reduction process broke user inputs in constituent parts to find appropriate matches against patterns. Approximately 41,000 elements (referred as categories) were scripted/developed later referred as ALIC's brain. Each element comprised of a question (stimulus) and answer (response) known as the "pattern" and "template" respectively. The scripted patterns were stored in a tree-like structure and managed by an object called "graphmaster" (Wallace, 2009). The AIML technology was also responsible for pattern matching and to relate a user input with a response in the chatterbot's Knowledge Base (Marietto et al., 2013).

Table 2.2 illustrates an example AIML category that comprises a pattern and a template. The pattern consists of a key pattern that has a wildcard (*) character to match any word or number of words at its position. In the example template, a variable (known as predicate) value is retrieved to prepare the response. The predicate will have been set previously in the dialogue using the markup <set name="name">Craig</set>. The predicates allow information about the conversation to be stored e.g. commonly used bind pronouns (i.e. he, she) to subjects (such as Einstein).

Source: ALICE AI Foundation, 2007

```
<category>
  <pattern>CAN I PLAY * TURING TEST</pattern>
    <template>
        We are already playing the Turing Game,
        <get name="name"/>.
        Now it's your turn.
    </template>
</category>
```

Table 2.2: Example AIML Category

The AIML recursion operator works as 'goto' command, repeatedly matching categories to divide up utterances or match keywords. The conversation context is driven by last utterance. Also, the categories are grouped into topics that are treated like ordinary words to responses. Both ALICE and AIML are widespread as the chatbot are freely available as open source. The distributed development of the ALICE's knowledge base allows the new patterns being added by many users. However, (Crockett et al., 2011) earlier agents faced criticism because of limited autonomous properties, lack of features, intelligence and context awareness that could influence, track and direct the conversation.

The InfoChat agents (Convagent, 2005) and OSCAR (Latham et al., 2012) are examples of CAs built using pattern matching technique. InfoChat is a goal-oriented CA engine, which is used by ConvAgent Ltd for commercial applications (2005). InfoChat has been successfully deployed in various environments as guidance and advice system such as the Bullying and Harassment Advisor (Latham et al., 2010) and Adam, the Student Debt Advisor (Crockett et al., 2009).

InfoChat is built with pattern matching approach, using spreading activation inspired by human consciousness and empowered with a sophisticated scripting language known as Pattern Script (Michie and Sammut, 2001). Scripts are engineered with set of rules that comprise stimulus patterns and responses. Each matched pattern to user input leads to generate a response. Pattern Script extend InfoChat's abilities by including more features than AIML i.e. better organised scripts that make development/maintenance more efficient and shorthand features like macros. Pattern Script enables organising of scripts of rules into contexts/topics that manage specific parts of a conversation. Table 2.3 illustrates an example Pattern Script rule that combines a number of ‘patterns’ with associated strengths, a ‘response’ and ‘activation’ level.

Source: (Latham et al., 2010)

```
<What-is-Bullying>
a:0.01
p:50 *<explain-0> * bullying*
p:50 *bullying *<explain-0>*
p:50 *<remind-0> * bullying*
p:50 *bullying *<remind-0>*
p:50 *<explain-0>* a bully*
p:50 *a bully*<explain-0>*
r: Bullying is persistent, threatening, abusive, malicious,
intimidating or insulting behaviour, directed against an
individual or series of individuals, or a group of people.
*<set BullyDef true>
```

Table 2.3. An example Pattern Script Rule

For example, in the rule “What is Bullying” consist of a rule name; ‘a’ is the activation level maintained to provide conflict resolution, and p is the pattern strength derived from the matched pattern against user utterance, ‘r’ is the CA’s. As each rule can contain number of patterns which match individually, therefore, scripts (PatternScript) are shorter and easier to maintain than AIML scripts. As shown in Table 2.3, patterns contain the wildcard (*) character, which allows matching of any number of words, which can later be retrieved for use in the response. PatternScript enables development of scripts in modular nature by grouping rules into sets referred to as contexts (Michie and Sammut, 2001). InfoChat has been successfully employed as a goal-oriented CA where the conversation domain is explicitly defined (O’Shea et al., 2011). Several features such as organising of scripts and managing conversations allow InfoChat to lend itself in goal-based environments.

2.3.3 Review of Challenges for CAs

CAs have proven their abilities as alternative tools for scenarios and platforms, which conventionally require human operators to provide information such as advice, guidance. There have been many CA developments in past decades but the success and their acceptance is limited. Following are some of the challenges that influence CA developments and their wider acceptance:

- The development of CA scripts is a labour intensive and time consuming process that puts an impact on development costs.
- The CA scripts are developed anticipating (what users will or might say) and backward looking (applying correction to incorrect responses) leading to a lengthy development time. This applies to CA developed techniques such as XML/AIML and PM.
- Skills in developing dialogues are essential and CA responses must be carefully scripted to maintain flow of conversation. In goal-oriented CAs, this drives the conversation towards its goal.
- Development skills are required for the selection of patterns and key words to match the required user utterances and give an appropriate response.
- Maintaining CA scripts is a difficult job as rules interact and compete with one and other, and even one rule change can destabilise a CA or fire incorrect responses.
- When applied to extended conversations rather than answering direct questions, e.g. about products, CAs lack the social intelligence of humans. To genuinely mimic human behaviour, CAs additionally need to be able to pick up and react to user affect, such as mood, personality, boredom, confusion or frustration (Becker et al., 2007).
- The CA applications lack scalabilities to cope with eventual expansion in conversational load. Reliability of information is another issue, which worries real business users to allow clients to be served through CA applications.
- Real-life environment users lack confidence in CAs' treatment of users with respect to their desired goals.

- The users often feel insecure revealing sensitive information to CA applications with fear of their information being shared or unauthorize use of it (O'Shea et al., 2011).

Although there exist several challenges, CA applications are able to communicate with users adequately in clearly defined domains. More recent advances incorporate human like behaviour into CA's, to enhance the user experience by developing CAs that are perceived more natural and less machine-like. There has been research in observing and reacting to human social behaviour in CAs such as social conversational skills, socio-emotional interaction (Kumar et al., 2010; Mairesse et al., 2007) detecting the type of user personality during the conversation from the entered text using linguistic cues. (Ma et al., 2005) used keyword spotting to estimate emotions from text-based conversation. Furthermore, CAs help in reducing costs by taking over responsibilities of repetitive tasks and can adopt to different domains. CAs enable exploitation and making the most of extendable knowledge base and record iterations automatically, which can help in performing analysis and improvements (O'Shea et al., 2011).

2.3.4 Existing Methods of Evaluating CA

In software engineering, the quality is described as the degree to which a system, component or a process meets user or customer needs (Hilliard, 2000). As suggested by (Roy and Graham, 2008) the quality of an application is determined mainly by the degree to which requirements, such as reliability, correctness and usability are met. The aspects that impact quality are known as quality attributes, which can be categorised in different ways. The international standard (ISO 9241-11) for usability can be defined as (UsabilityNet, 2017):

"The extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use."

This consists of three following areas (UsabilityNet, 2017):

- Effectiveness – Objective metrics i.e. completion rate, no of errors etc.
- Satisfaction – Objective metrics i.e. SUS questionnaire etc.
- Efficiency – Subjective metrics i.e. time to complete tasks.

Information, on user behaviour and perception, is required for the evaluation of a conversational system (Skantze and Hjalmarsson, 2013). The evaluation of dialogue systems has been considered as a difficult challenge (Martinez et al., 2008). There are no evaluation standards within the NLIDB community. Also, it is difficult to explore performance figures from real world systems that can be ported/utilised to other systems. Each conversational system has been developed and evaluated through directly related performance figures or measures. Turing test is one of the early examples of evaluating the success of a dialogue-based system. The Turing test (Turing, 1950) stipulates that a conversational application should make a human believe that they are speaking to a human and not to a computer program.

Considering what is expected from goal-oriented conversational systems, the Turing test approach is not suitable to measure factors such as usability and effectiveness as both the (goal-oriented and general purpose CAs) applications differ in nature. The 'usability' has been agreed and regarded as one of the most important performance figures (Turunen et al., 2006; Walker et al., 1997). Other commonly used measures are "flexibility" or "naturalness". Although, functionality is one of the major measures, however, in the absence of usability a conversational system wouldn't be able to demonstrate functionality. Besides efficiency and quality measures, computed or autonomous properties, subjective measures have been conducted in order to evaluate and measure perceived user perception of the system, advantages and highlight shortcomings of a system (Martinez et al., 2008).

There has been a substantial amount of research conducted on to evaluate CAs systems. Work from (Walker et al., 1997) has been regarded as influential with the creation of PARADISE framework. The PARADISE evaluation framework employs the application of linear regression to derive abstract and indirect attributes i.e. using directly measurable attributes to determine user satisfaction (Fenton and Bieman, 2014). Several aspects are of interest when determining the quality of dialogue systems. Moller et al. (2009) presented a taxonomy of quality criteria, which included quality as two separate aspects consisting of Quality of Experience (QoE) and Quality of Service (QoS). The quality of Experience describes the user experience with subjective metrics i.e. user understanding of the system etc. Quality of Service relates to objective metrics such as total number of dialogues, dialogue duration. Others have explained, such as (Silvervarg and Jönsson, 2011), that evaluation of a conversational system can be carried out by studying the history of dialogues or by the distribution of questionnaires to the users to reveal their subjective assessment. Later research supports this idea carried out by (Rauschenberger et al., 2013) who propose a framework to measure software quality and user experience.

A general agreement among researchers suggests that a combination of subjective and objective metrics should be utilised for the evaluation of CA/Dialogue systems (Alobaidi et al., 2013; O'Shea et al., 2011; O'Shea et al., 2009). A combination of subjective and objective metrics will ensure that CA's usability (from users perspective) is evaluated and not only the effectiveness of its functionality.

2.3.5 Formulation of Evaluation Metrics

Software development is conventionally driven using measurement mechanism for evaluation and feedback. Measurement enables us to gain insights into the quality of specific products and processes as well as revealing strengths and weaknesses in processes (Van Solingen et al., 2002). Recognising the improvement and outcome of the process is achieved through a set of clearly defined project goals for the processes and systems. In the absence of the destination, it isn't possible to determine if one is going in the right direction (Fenton and Pfleeger, 1998).

An evaluation methodology has more probability of success if it is devised with project goals in mind (Fenton and Pfleeger, 1998). Goal Question Metric (GQM) methodology is an example that is based on the understanding or assumption of an organisation on evaluating applications or processes in a focused way. For example, first, identify the goals for the projects followed by tracing those goals to the data intended to define to describe selected goals operationally. Finally, facilitate with a framework for translating the data into the stated goals (Van Solingen et al., 2002).

The quantifiable sources of information should be selected in line with the information needs (wherever possible) so that, quantified information can be analysed to determine whether goals are achieved.

The GQM approach comprises a framework including following three steps:

1. **Goal** – List the main goals of the development
2. **Question** – Each goal dictates the questions that must be answered to evaluate that goal is achieved. Use of questions helps in describing the objects of measurement (such as resource, process or product) for the selected quality issue and to evaluate its quality from the selected viewpoint. The next step is relating each question with appropriate metric.
3. **Metric** – Derive from each question determine what must be measured in ordered to answer all questions. Specific data is associated to each in to answer it in a quantitative way, which can be:

Objective, which includes the object's evaluation from a single perspective that is being measured; e.g. a number of staff hours spent on a task etc.

Subjective, which includes users' viewpoint e.g. as dialogues naturalness or how likely the user would use the software again.

(Van Solingen et al., 2002; Fenton and Pfleeger, 1998)

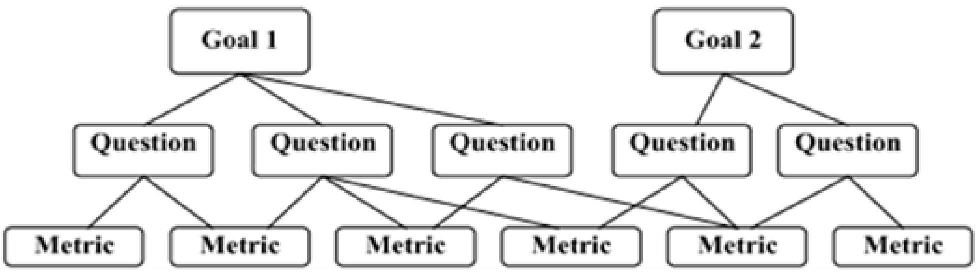


Figure 2.1: Goal Question Metric model ((Van Solingen et al., 2002))

As illustrated in Figure 2.1, GQM is a top-down hierarchical model. The top level starts with a goal (the purpose of measurement or expected achievement from the project). The goals can be further divided into questions that decompse the issue into its main components. Each question is then translated into an evaluation metric, which can be objective or subjective in nature. However, a single metric may also answer or provide information to answer more than one question (Van Solingen et al., 2002).

2.4 Existing Conversation Enabled NLIDB Systems

As the strengths and weaknesses of different applications become better understood, it is possible by combining CA and NLIDB approaches to take advantage of one another. The development of NLIDBs has been largely focused on single query response transactions. Little attention has been given to social adaptability, sustained interaction, information refinement and how domain knowledge is structured (Owda et al., 2007; Damljanovic et al., 2011). Conversation or dialogue based NLIDB systems can offer users a friendlier interaction experience, clarify facts and produce desired results, as well as refine produced results from extended dialogues (Shabaz et al., 2015; O'Shea et al., 2011).

As mentioned in section 1.3, CAs have proven their abilities in real life deployments and have shown effectiveness in guiding users during conversation towards achieving their desired goals, across many domains. For example, ADAM conversational agent developed by ConvAgent (2005), worked as a debt advisor for eight years in real life environment. CAs are an ideal candidate for NLIDBs to increase their potential and capabilities not only for information retrieval but also for conversational aspects such as social adaptability, friendliness, information refinement and querying the query operations, etc. The CA implemented NLIDB systems for structured data, can help in tackling problems linked to conversation such as extended dialogue, domain knowledge (Pudner, et al., 2007).

An example of CA implemented system is NLI-RDB (Owda et al., 2007), which was built on the same principals as ADAM (Crockett et al., 2009). The NLI-RDB relied on the Knowledge Tree approach. The NLI-RDB system was restrictive by nature as it initiated interactive sessions by presenting users with selective hard-coded conversation topics. The use of the Knowledge Tree approach limits users to choosing from menu-based options on the screen and steers the conversation in a pre-set path. NLI-RDB was implemented with hand-coded SQL query templates stored as responses. The system was not able to produce any results for user requests not covered in existing hand-coded SQL query templates. Also, the NLI-RDB system was evaluated using a generic database comprising a simple schema structure, adding considerable disadvantage in measuring other features. The NLI-RDB system reflected nominal conversational features and possessed limitations in its abilities to only respond to tractable (i.e. where user request lead to the end note of tree corresponding a stored SQL query template only) user requests, thus widely lacked in several aspects in contrast to desired NLIDB features.

NaLIR is another example of NLIDB, which was developed by (Li and Jagadish, 2014), to act as a Keyword-based document search interface. NaLIR was evaluated using Microsoft Academic Search database. The development NaLIR primarily focussed on interactively helping users to translate their requests into database queries accurately. The system was built using Syntax-based approach relying on the Stanford NLP parser for the query translation process. The interactive abilities of NaLIR allowed shallow disambiguation in users requests based on multiple possible query translations. However, NaLIR was built as a single query response system and where necessary displays user to choose from possible translations (multiple sub-trees) for resolve the part of a request. The system relies on a Syntax-based approach, which inherently requires users' requests to be grammatically correct (Castillo et al., 2014) and can only react to ambiguous situations in a non-intelligent manner.

More recently, Anand and Farooqui (2017) developed a NLIDB system based on NLP, which also adopts a rule based technique to identify a context and then triggers contextual dilaogues. The system has been developed for single transaction queries, and due to the use of NLP building technique it requires user input to be grammatically correct for appropriate query transation. The developed system can only formulate basic SQL queries, therefore lacking conversational and query formulation abilities to address the historical challenges highlighted in section 2.2.4. Another attempt towards building conversational NLIDB is NLI-RDB (Alghamdi et al., 2017), which has been developed based on an earlier (Owda et al., 2007) principles to allow users to interact with objects directly in natural language and through navigation, rather than by using

SQL queries. The Object-Relational Mapping (ORM) framework (with the help of Hibernate framework) has been used that communicates with and has a shallow understanding of the underlying domain database. The use of the Hibernate framework technology has advantages, but it has also been criticised for several reasons. This method requires the developer to persistently define the mapping between the object model and database schema, which express database-access operations regarding objects. Additionally, organisations operating the Hibernate framework, often ignore performance and testing of the data persistence layer that can lead to the system failure (Wu et al., 2010). This attempt was also single query transaction oriented and failed to address the historical challenges such as sustained dialogues, information refinement with dynamic query formulation and querying the query operations.

2.5 Conclusion

This chapter has introduced Natural Language Interface to Databases, which allow inexperienced users to retrieve desired information stored in databases. A number of NLIDB systems were discussed. Different NLIDB building approaches were described. There are a number of associated factors which led to the lack of acceptance for NLIDBs in real-life environments (Pazos R et al., 2013). Among note able discouraging factors, a significant proportion remains unsolved, as researchers who worked in this area, did not further improve their prototype systems. The most common challenges and concerns identified by majority researchers are namely; conversational/linguistic problems, domain independence, natural language translation process (database query formulation), result refinement, multimodality and ease of configuration (Pazos R. et al., 2013; Crockett et al., 2011). Different NLIDB building techniques in relation to their pros and cons were described. (Kaleem et al., 2014; Becker et al., 2007)

The concept of conversational agents as computer systems that facilitate communication between users and computers using natural language was considered. The research advancements in the field of CAs, their ability to interact and engage users in sustained dialogues were described. A review of historical CAs along with recent successful implementations has been discussed. For example, one of the earlier CAs (ELIZA) was built to engage users by asking random questions to prolong the conversation and did not have any meaningful purpose or goal to achieve. Similarly, ALICE (Chatbot) was built to rely on a large knowledge base comprising rules for general conversations, whereas, for goal-oriented scenarios such as tutoring or passport applications assistance, InfoChat and UMAIR are more powerful and offer features of the scripting languages. Different CA building techniques with pattern matching text-based technique were discussed in more detail.

Furthermore, the feasibility of combining both CA and NLIDB to work together towards a common goal "providing a conversation based interaction to users for accessing and refining desired information stored in a database with querying the query operations" was also discussed. A CA enabled NLIDB system can allow users to extract and refine their desired information stored in a database. This advantages of such as making information retrieval interactive/conversation-based, friendly and effortless with the possibility wider implementation in the real-life environment. Finally, existing methods of CA and NLIDB evaluation were also described.

2.6 Chapter Highlights

- Natural Language Interfaces to Databases (NLIDBs) allow users to retrieve desired database information using natural language.
- Existing NLIDB building approaches still pose limitations and challenges leading to their lack of success and acceptance.
- Conversational agents (CAs) enable natural language communication between a user and a computer.
- Pattern matching approach is mostly used in text-based CAs as and is fast enough to respond in real-time environments. The system build using PM approach can handle different language related challenges such as grammatical and sentence structure.
- Conversational agents (CAs) can support extended dialogues during the conversation with users, therefore, can be integrated into NLIDB to enable features such as information refinement from multiple dialogues.
- Developing CA scripts is a time-consuming, complex and labor-intensive task.

- There exists no evaluation benchmark to evaluate NLIDB or CA enabled NLIDB applications.

Chapter 3 - A METHODOLOGY FOR DEVELOPING A CONVERSATIONAL NATURAL LANGUAGE INTERFACE TO DATABASE (NLIDB)

3.1 Introduction

This chapter introduces methodological steps for the development of a novel conversational natural language interface to database (NLIDB) with abilities to engage users in conversation to extract and refine desired information stored in a relational database. The next section (Section 3.2) will give an overview of the proposed NLIDB system (known as ANEESAH) followed by a section (3.2.1) detailing the considerations, approaches and development of different components. The development and engineering of conversation scope, scripting methodology and knowledge base is detailed in section 3.2.2. Finally, section 3.2.3 includes the detail on a generic architecture for ANEESAH, which is modular by design and flexible for further development and maintenance. Throughout this thesis, the term ANEESAH refers to an overall NLIDB system that engages with inexperienced users conversationally, to formulate complex queries dynamically to extract database information and can perform continual querying the query operations for information refinement.

3.2 ANEESAH Conversational NLIDB

The main aim of the proposed NLIDB (ANEESAH) is to interact with users and conversationally guide users towards achieving their desired information from a domain specific database (explained in section 4.4.1 in chapter 4). Instead of working as a single transaction query system, ANEESAH is expected to mimic or act as a human structured language query (SQL) expert to engage with users to dynamically formulate SQL queries to extract database information with the ability to offer the continual information refinement supported with querying the query operations. As identified in chapter 2, there are a number of development techniques and approaches available for NLIDB and Conversation Agents (CAs). The development of ANEESAH will combine these techniques to develop both NLIDBs and CAs, i.e. pattern matching, rule-based algorithm, structured query language formulation. In summary, the key expected features from the proposed NLIDB are:

- ANEESAH will simulate a human SQL expert and offers an intuitive, natural language interface to help users to engage in conversation to make their requests.
- ANEESAH will use and apply conversational features (such as clarification, resolving or completing missing information) to illicit goals from user requests and understand their requests in the light of information stored in system's knowledge base.
- ANEESAH will translate complex users' requests to formulate queries for a specific domain database used for evaluation.
- ANEESAH system will execute queries to extract information stored in a specific domain database and display results and offer users to refine query produced information.

In the next section (3.2.1), a generic methodology has been proposed for developing different components such conversational agents, natural language interface to database, conversation design, query formulation techniques, evaluation methodology for evaluation.

3.2.1 Phase 1: Components Development

This section of the thesis will explain methodology adopted for the development of different components for proposed ANEESAH.

3.2.1.1 Adopt a NLIDB Building Approach

Building computer applications that mimic a human (in constructing database queries) is complex and time-consuming. Such applications, along with scripting knowledge, require expertise in extraction and formatting of expert knowledge (O'Shea et al., 2011) i.e. subject knowledge, database schema information, structured query language knowledge, etc. Formalising the development of a conversational NLIDB that can be adapted to different domains will help to speed up the development. The NLIDB building techniques should be investigated to produce a suitable development method in line with the scope of this research.

3.2.1.2 Selection of a Domain Database

The selection of a sample dataset or domain database is one of the preliminary steps in NLIDB building. This sample dataset is also called domain database, which plays a key role in its evaluations by serving as an information source for conversationally formulated queries from which results may be extracted. Ideally, the domain database for NLIDB ought to be complex in schema structure (information about tables, columns, constraints, etc.) with large number records. As discussed in chapter 2, there is no universally accepted benchmark or standard specifying in what sample dataset should be used for the evaluation of similar NLIDB systems. This, adds to the difficulties in building a NLIDB with ideal properties. However, a domain database in line with the aim of this research should have properties such as complex relational structure and a large number of historical data records.

3.2.1.3 Analyse Real Life Information and Query Requirements

A user's desire to access database information has two main aspects namely; what information is requested, and requirement of a query to produce that information. Seeking an in-depth understanding of how information is requested in real life environments. Most modern enterprise resource planning (ERP) applications (i.e. Oracle, Microsoft, SAP, etc.) deliver ready to use department-specific reporting suites. For most organisations, off the shelf business applications are not able to deliver unique/bespoke business needs such as coping with ever growing database records, the constant requirement for information and development challenges, which in turn leads to the requirement for help from database experts. A critical review and research on ERP systems and chatting with information users from different organisations will provide an idea of information request and delivery process. Conducting meetings with database experts will highlight what implications and challenges involved in their roles.

3.2.1.4 Determine Conversation Scope and Structure

This phase of the methodology includes capturing of real life information and query requirements for the development of conversation structure in proposed system. A number of scenarios and information requests can be devised not only to construct the conversation style but also for evaluation to see if, the proposed system can produce the required outcome by means of conversation. The conversation style should lead the proposed system to formulate complex structure queries (i.e. multiple database tables joins to collate information, use of mathematical/aggregation functions, etc.) where required. The use of real life information requirements to build conversation structures will provide users with a natural experience in interacting with the agent to gain access to desired information.

3.2.1.5 Develop Knowledge Base Structure

This step will involve selection and design of a knowledge base structure, which will maintain information such as scripted responses for conversation, query related syntax and database schema information. The information sought in sections 3.3.1.2, 3.3.1.3 and 3.3.1.4 can be used to prepare different blueprint documents to design different conversational topics, knowledge base contexts and query relevant factors (i.e. scripting of query syntax, database schema information, etc.). The proposed knowledge base should offer support query and non-query (text based) based responses. The proposed knowledge base will be scripted with relevant rules and patterns to enable the ANEESAH system to react with appropriate responses. The proposed knowledge base structure for the ANEESAH System will be comprised of three divisions namely; (1) domain database scripts for query based responses, (2) Frequently Asked Questions scripts that will deal with database structure/system related questions such as “how many tables are there in the database?”, and (3) General Chat to manage off topic questions such as “what is the weather like today?”. The introduction of multiple contexts will strengthen ANEESAH’s conversational abilities to allow users to interact freely.

3.2.1.6 Devise Methodology for ANEESAH's Evaluation

As discussed in chapter 2 no standard for the evaluation of CA or NLIDB applications exists, which in turn creates the need for designing a new evaluation methodology for the proposed system. The evaluation methodology design should include a wide spectrum of subjective and objective metrics, as described in chapter 2 Section 2.3.5. The design focus should be aimed the evaluation of major features of the proposed system such as dialogue naturalness, robustness and information accuracy. The evaluation methodology design should also include set experiments (in line with the selected subjective and objective metrics). The information obtained from experiments should be measured through statistical tests such as descriptive and inferential statistics.

3.2.2 Phase 2: Conversation Scripting and Query Formulation

The conversation will be scripted with a view to evaluate if, users can conversationally achieve desired database results, by using the prototype ANEESAH NLIDB system, that will otherwise require a SQL query. The structured query language is widely used in information systems environments to perform various operations on the data such as retrieving subsets of information, or adding, deleting, updating records. The conversation designed and query formulation technique following phase 2 of the methodology, is described in the following sections.

3.2.2.1 Selection of a scripting methodology

The pattern matching technique implemented in similar conversational agent and natural language interface to database systems has proven effectiveness in sustaining dialogues during conversation with users (Kaleem et al., 2014). This technique not only allows scripted dialogues based responses to support conversation to take place but can support matching and extraction (from user utterances) of database relevant information to formulate queries. The implementation of a pattern matching technique will be coupled together with sentence similarity approach to support ANEESAH's CA to produce accurate and definitive responses. Although, pattern matching allows scripting of the dialogue responses it has limitations in checking for rigid expressions requiring case-sensitivity, non-alphanumeric symbols or the syntax of programming code, such as SQL query syntax.

3.2.2.2 Map/Organisation of Conversation Scripts

As discussed in section 2.1.5, the knowledge base structure for the proposed architecture will be segmented in three divisions (domain database scripts, frequently asked questions, general chat). Adopting a strategic approach is pinnacle to develop conversation contexts and organise CA scripts for ANEESAH. Sammut (2001) described a technique of managing conversations by grouping rules into sets, called contexts. The contexts with stored scripts represent individual conversation topics. This approach helps the system recognise the current state of the activated context at any point during conversation. Pattern scripts allow control to switch from one context to another, which is useful in a structured conversation such as sustained dialogue for information refinement. The use of context techniques makes development and maintenance of scripts easier. This technique offers flexibility for further development. Contexts based collection of conversational scripts will be used to develop and maintain scripts to cover full conversational scope. The scope will require responses to events such as contexts containing rules scripted to discuss frequently asked questions about the database or system, where a user utterance has failed to match any rule script or a partially matched utterance.

3.2.2.3 Develop Technique to Determine System Responses

This step of the development methodology involves designing of a technique for ANEESAH's CA to handle scripted query and text-based responses as well as recognising and supporting of non-scripted responses. The technique will enable ANEESAH's CA to work with scripted responses stored in the knowledge base discussed in section 2.3.5. This technique should also allow ANEESAH's CA to perform user request matching across knowledge base scripts in a sequential order to determine the appropriate response. This technique should equip ANEESAH's CA to deal with each user request and relevant response (whether query, non-query, text-based) accordingly. This technique should clarify when a user request might trigger a single transaction response (once matched in the knowledge base against scripted responses), and detect requests that will direct the system to follow a non-scripted query response process.

3.2.2.4 Develop a Layered Based Request Matching Approach

Due to the implications of conversational structure and natural language translation into database information, it is necessary to develop mechanism and conversation that can work to formulate queries in events such as non-scripted database information requests. The query building mechanism/algorithm from user requests should adopt a sequence of steps that will support mapping and capture of database and query syntax related information. The capturing of the database and syntactic information will be used for the query formulation purpose. This mechanism should be engineered with a set of conditions and rules such as deciding on when minimum query information is collected when to offer users to display results.

3.2.2.5 Develop/Adopt a SQL Query Formulation and Refinement Engine

The SQL query formulation ability will play a fundamental role in the proposed architecture. The query formulation ability will be implemented as a SQL query engine. The SQL query engine will be responsible for the dynamic formulation of database queries, information retrieval and refinement process. The proposed query engine should be modular in nature. The SQL engine should not rely upon pre-authored query templates but should only receive query syntax and type instructions, which should be sufficient to perform the query formulation process. The SQL engine should be constructed with a designed path that, working in a pre-defined sequence of steps, prepares and brings together the required syntactical parts of a query. In addition to the initial formulation, the SQL engine should be equipped with query refinement techniques. The SQL engine should be scripted with expert knowledge/techniques, which can be used to analyse syntactical gaps, extract database schema information, formulate queries that are complex in nature and perform enhancement/alteration to the existing queries.

3.2.3 Phase 3: Design Architecture for ANEESAH NLIDB

The ANEESAH NLIDB architecture will include several components such as a CA, the knowledge base, a Graphical User Interface and the SQL query engine. The proposed ANEESAH NLIDB architecture is generic and incorporates related components, which will be briefly discussed below.

3.2.3.1 ANEESAH NLIDB Architecture

The ANEESAH NLIDB should enable users from any background to interact conversationally without having to ask their inputs/requests in specific order. Therefore, it is important that ANEESAH NLIDB architecture maintains a comprehensive knowledge base to understand user requirements. The architecture should allow domain database to be changeable and incorporation of a new domain knowledge base with minimal effort. In addition, ANEESAH's response in helping users to extract their desired information from domain database should be minimised. Therefore, an architecture with modular approach has been proposed as most suited, which will allow customisation, enhancement and replacement of individual components. Figure 3.1 highlights the proposed generic architecture of ANEESAH NLIDB that has been designed with a modular structure. Next, the major components of the architecture are briefly discussed below.

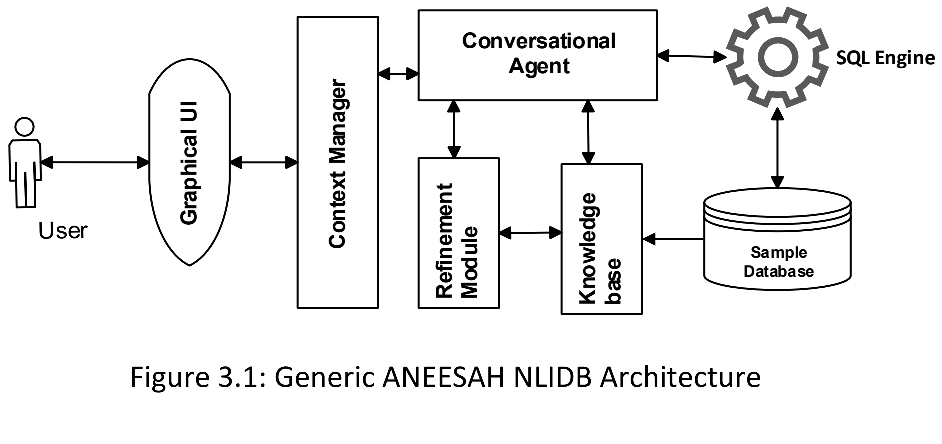


Figure 3.1: Generic ANEESAH NLIDB Architecture

3.2.3.2 Graphical User Interface (GUI)

The Graphical User Interface serves as interaction interface for users to converse with ANEESAH NLIDB. The Graphical User Interface is responsible for sending and receiving communication to and from the users. The GUI displays instructions, form control (such as a clickable button, check boxes), database results and formulated SQL queries. The GUI component contains an interactive chat area used to communicate with users. ANEESAH NLIDB directs conversation when interacting with users; no navigation buttons are included as there is no menu system. The modular nature of the architecture allows changes to be made to the GUI component to reflect application needs such as including voice or speech recognition feature, or deployment on portable devices.

3.2.3.3 Conversation Manager

The Conversation Manager is one of the major components of the architecture. Conversation Manager is developed to manage the conversation flow in a predefined path. The Conversation Manager works closely with other components (Conversational Agent, Short-term Memory) to help ANEESAH to behave naturally. All communication (to and from users) and the information is passed through conversation manager module.

3.2.3.4 Conversational Agent (CA)

The conversational agent plays a key role in ANEESAH's ability to interact with users conversationally. The conversational agent has been developed to receive users' requests related to the topics stored in ANEESAH's knowledge base, perform user request validation and user request matching process to generate natural language responses (including formulating database query based responses to display information from a database).

3.2.3.5 Knowledge base

ANEESAH's knowledge base is responsible for managing information about three main topics namely; the domain database, frequently asked questions about the system and the domain database, and general chat or off topic questions such as weather, sports, etc. The knowledge base will receive information from the graphical user interface and conversational agent's components via conversation manager, and will send information back to the graphical user interface via conversational agent components.

3.2.3.6 Information Refinement Module

Information Refinement Module is responsible for supporting users to refine query produced information from a domain specific database. Once a successful query is executed, the refinement module is activated to receive further refinement/adjustment requests. With the help of the information refinement module, users can perform refinement of information such as add/remove database information, selection of results, etc.

3.2.3.7 SQL Engine

SQL engine component is responsible for the dynamic formulation of queries, execution of formulated queries in a domain specific database, and transferring of queries produced results to the conversational agent to display in graphical user interface. In line with the user request, the controller component of conversational agent collects and then transfers relevant query information to the SQL engine, which uses this information to construct a database query. The SQL engine will perform dynamic formulation of queries and perform querying the query operation (e.g. ability to reformulate/change existing query to produce different results as per user request).

3.3 Conclusion

This chapter has proposed a novel conversational natural language interface to database call ANEESAH, which can process users requests interactively to allow access to desired information stored in a sample database. Conversational agents (CAs) and natural language interfaces are easy to use, as in real life we are used to communicating using natural language. An improved natural language interface coupled together with a CA can feel more natural and accessible to users, improving their effectiveness and motivation in the day to day tasks.

A conversation enabled natural language interface can adopt the role of an SQL query expert and bring ease of access to database information for inexperienced or naïve users (users of information with no structured query language or database knowledge). ANEESAH NLIDB can understand users' conversational requests, translate these into query relevant information, formulate queries and perform query execution to produce database results. ANEESAH's ability to allow and detect refinement requests also enable users to take insights into database information. After a query produces responses, ANEESAH can perform alterations to the formulated queries for information refinement. By continually offering to refine information it enables users to users to extract, manipulate and analyse information at will.

Instant availability of database desired information using natural language eliminates the requirement for information users to rely on structured query language experts. A similar system to ANEESAH, can support users with real-time database information without any delays, aid effectiveness and improve productivity. A generic methodology was proposed to design ANEESAH NLIDB, which will involve three-phase methodology as discussed above.

Chapter 4 - ARCHITECTURE FOR DEVELOPING ANEESAH NLIDB

4.1 Introduction

Based on the research conducted into the development of NLIDBs, it has been established that to date and to the researcher's knowledge there exists no approach to building text-based conversational NLIDBs that can address key challenges such as social adaptability, sustain dialogues, dynamic querying the query operations (discussed in Chapter 2).

This chapter describes the development of a novel architecture and framework components for the proposed conversational Natural Language Interface to Database (NLIDB) called ANEESAH. ANEESAH is a new conversational NLIDB that can perform as a human structured query language (SQL) expert, engages users in conversation and dynamically formulates queries to extract information stored in a specific domain database. The first phase of this research aims to validate a developed NLIDB architecture and perform a proof of concept that the ANEESAH prototype can engage users in conversation and formulate queries to extract their desired information stored in a specific domain database.

The novel NLIDB architecture has been designed specifically to offer conversational abilities (such as conflict resolution) to provide an interactive and friendly environment to assist users with desired information stored in a specific domain database. The proposed architecture comprises several components shown in Figure 4.1. The main features of the ANEESAH prototype in the first of phase of the research are:

1. ANEESAH is able to interact with users conversationally to produce database information in a limited/specific domain.
2. ANEESAH can support multiple inputs from users to elicit their desired goals and offer single transaction-based query results.
3. By following a Pattern Matching (PM) approach (discussed in chapter 2 Section 2.3.1), ANEESAH can understand user requests by way of matching utterances against scripts stored in the knowledge base.

4. ANEESAH employs developed features to extend its conversational abilities such as perform conflict resolution and controlling the direction of conversation with users.
5. ANEESAH is able to translate complex users' requests into SQL data manipulation statements.
6. ANEESAH is able to execute queries to extract information stored in a specific domain database and display results.

4.2 Overview of Architecture

Figure 4.1 gives a high-level overview of proposed architecture and different components in supporting ANEESAH's role to create structured query language (SQL) equivalent to that from a human expert, by leading users in conversation to produce desired information from a database. The proposed architecture is modular in nature for ease of development and maintenance with flexibility to integrate and adopt other domains with customisation.

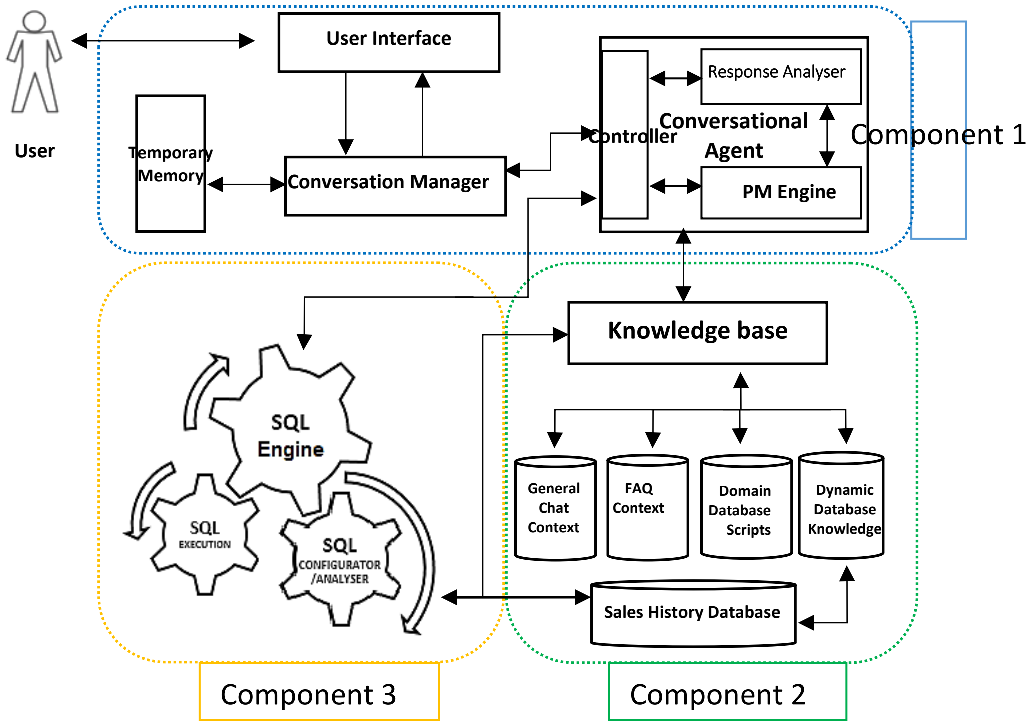


Figure 4.1: High-level overview of proposed architecture (prototype one)

ANEESAH's architecture comprises of the following components which will now be described.

Conversational Agent:

A Conversational Agent (CA) for ANEESAH had to be developed with its own associated mechanisms because the literature review (in chapter 2) revealed that existing techniques did not contain features necessary to be applied to NLIDB problems such as conversational limitations, conflict resolution.

- A Controller that is used to control the utterance processing steps (discussed in Section 4.3.1) within the proposed system in order to achieve user desired response.
- A PM engine has been developed to take the role of user utterance matching processes. The PM engine does this in two parts/tiers namely; the first tier deals with user utterance matching against scripted patterns/rule-based responses stored in the knowledge base and second tier deals with user utterance processing against database information to build a manual query-based response (discussed in Section 4.3.2).
- A Response Analyser feature which is responsible to fully analyse users' utterances before the developed system initiate any query based response discussed in Section 4.3.9.
- A Conversation Manager has been developed to manage conversation flow in line with predefined path discussed in Section 4.3.10.
- A sentence similarity feature has been implemented to determine match strengths between two phrases in Section 4.3.3.
- A new scripting language had to be developed because the literature review revealed that existing scripting language techniques did not contain features necessary to be applied to NLIDB problems discussed in chapter 2. The new scripting language enabled the following development:
- Develop context based (e.g. Domain Specific Scripts, Frequently Ask Questions and General Chat contexts) rules and patterns (e.g. scripts written to match

against user utterance to find appropriate response) necessary to enable conversation in the system's knowledge base (explained in the component 2 called knowledge base).

- Develop dialogue responses and results set from queries.
- Develop features to clarify and provide conflict resolution (e.g. that might arise from poorly structured or unclear user utterances) while interacting with users.
- Develop for providing query related syntax to formulate SQL queries.

Each sub-component of Conversational Agent is discussed in section 4.3.

Knowledge base:

Component 2 consists of the following features of ANEESAH's architecture.

- A knowledge base has been developed for ANEESAH prototype to hold all the domain knowledge in a relational database which includes scripts, rules and patterns based on different contexts. Scripts has been developed based on the knowledge engineering techniques to specifically work with a sample domain database (Sales History database). The developed knowledge base can be customised with knowledge engineering to work with a different domain database.
- Responsible for supporting a two-tier based user utterance matching across three different domains/contexts e.g. Domain Database Scripts, FAQ Domain, and General Chat Domain (discussed in Section 4.4).
- Responsible for supporting user utterance mapping to match across different contexts.
- A Dynamic Database Knowledge module (part of the knowledge base that is updated/initialized at system start-up with most up to date domain information) to perform user utterance mapping against domain database to detect and extract information/syntax necessary to formulate SQL queries.

SQL Engine

Component 3 of the ANEESAH's architecture comprises of following novel features:

- SQL engine, after having received database relevant syntax/information from the controller component. The SQL engine is responsible for the dynamic formulation of queries to extract information/results stored in a database (discussed in Section 4.5).
- The SQL engine works together with the SQL Configurator component, which is responsible for analysing, exploring and acquiring the necessary database syntax (i.e. primary keys, tables, joining conditions, etc.) to formulate complex queries. Formulated queries are forwarded to the SQL execution components. However, when query formulation fails, the user is displayed with an appropriate message (discussed in Section 4.5.1).
- The SQL execution component is responsible for executing queries in the database followed by producing, storing and displaying results back to the users (discussed in Section 4.5.2).
- The SQL analyser performs analysis of query structures in the case where system failure is related to a formulated query. The SQL analyser works by recognising the actual query failure reasons (such as incorrect structure, missing syntax), leading to system failure. Also, each query failure in the database results in an appropriate message (often in the form of vendor code detailing the actual reason for failure), which is utilised by the SQL analyser to apply an adjustment (if possible) before re-running it (discussed in Section 4.5.3).

Each component with respect to its sub-components will now be fully described.

4.3 Components Development for ANEESAH NLIDB (Component 1)

Component 1 of ANEESAH's architecture comprises of Conversational Agent, Conversation Manager, User Interface and Temporary Memory. The CA component is a fundamental component of the proposed architecture that enables users to interact with ANEESAH for database information retrieval purpose. The CA comprises of sub-components highlighted in Figure 4.2.

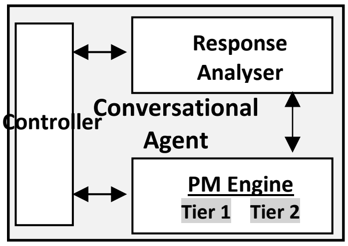


Figure 4.2: CA Components of ANEESAH NLIDB

The following sections will provide detail on different components of ANEESAH's CA followed by sections detailing Conversation Manager, User Interface, Temporary Memory. The Conversational Agent (CA) is comprised of the following components:

- Controller
- Pattern Matching (PM) Engine
- Scripting Language
- Response Analyser

4.3.1 Controller

The controller module has been designed to control and handle the direction of conversation with the users focusing on retrieving their desired information from the database. The controller leads the conversation during dialogue sessions between users and ANEESAH. The controller takes responsibility for validating each user utterance before forwarding it to the Patterning Matching engine for further processing. The role of controller module can be described as following:

1. The controller validates user utterances and deals with invalid utterances. Utterance validation is performed to ensure that the utterance does not contain bad/offensive or swearing words, or contains unnecessary characters and it is not off-topic or irrelevant to the domain/contexts.
2. Firstly, the controller makes a copy of user utterance in ANEESAH's log file, followed by removing any unnecessary symbols (such as "!", ".", "*", "^" etc.) to simplify the utterance before proceeding with the matching process.
3. The controller features a built in three warnings rule in situations where it detects offensive utterances or abuse of the system. The three warning rule has

been successfully implemented in other conversational systems such as ADAM debit advisor (Crockett et al., 2009), UMAIR (Kaleem et al., 2014) and OSCAR (Latham et al., 2012). The user is warned, or session is terminated depending on how many times the unacceptable language is used in the session.

4. The successfully validated utterances are forwarded to the PM engine by the controller. After the utterance matching process, the controller is responsible also for receiving matched responses and displaying to users where necessary any accompanying query according to the fired rule/query. The controller works closely with the PM engine and other components to ensure and manage the entire conversation. The controller is the core of the CA and works in conjunction with several other components to ensure the conversation goal is achieved.

4.3.2 Pattern Matching (PM) Engine

ANEESAH has been implemented using a novel Pattern Matching (PM) engine. The PM engine controls user utterance matching against the system's knowledge base. The PM engine works based on a two-tier (Tier 1 and Tier 2) approach. The PM engine is responsible for finding matched responses from both tiers. The Tier 1 deals with user utterance matching against information stored in domain database (a sample sales history database used for system evaluation discussed in Section 4.4.1). The Tier 1 part does not adopt a rule based utterance matching. A user utterance is matched across information stored in sample database to capture co-occurrence of attributes leading to the formulation of a query based response (see Figure 4.5 for more detail). The Tier 2 adopts rule based utterance matching technique to find appropriate/matched response across hand written patterns stored in system's knowledge base. The PM engine works on principle of a conventional rule-based pattern matching approach implemented in other conversational system such as OSCAR (Latham et al., 2012). The PM engine has been designed to work with rule based and non-rule based response handling. A rule based response can be described as a scripted textual response, executed following a successful utterance matching in either Domain Database context, FAQ context or General Chat context. A non-rule (Tier 1) based response follows a manual query following from a successful utterance processing against Domain Database Scripts, as shown in Figure 1.5.

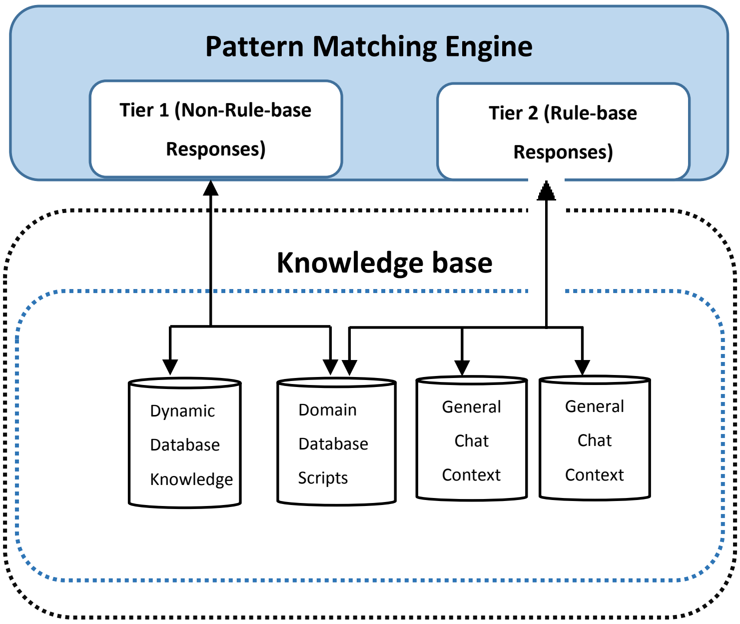


Figure 4.3: Pattern Matching Engine

The PM engine has been designed to work with other components of ANEESAH's architecture that work collectively to determine appropriate responses. Such components can be described as pattern matching, scripting language (discussed in section 4.3.5) and sentence similarity calculating algorithm (discussed in section 4.3.3). The user utterance is categorised once a match has been found in the knowledge base. Further, the relevant context (Domain Database context, FAQ context, or General Chat context) is activated. The context activation is used by the PM engine to engage the user in conversation and stay relevant to the topic. Following up from a context activation, the PM engine continues to assume/relate subsequent user utterances to the same context until a successful response is fired. The PM engine is equipped to recognise changing contexts and allow users to switch conversational topics at desire. The PM engine follows an implemented matching algorithm (discussed in Section 4.3.2 Table 4.2), which performs matching of each user utterance across all knowledge base domains (Domain Database context, FAQ context and General Chat context) to detect context or topic switching. Figure 4.4 gives a high-level description of ANEESAH's functional flow.

The PM engine can deal with more than one matched responses. The PM engine employs (an on-demand) sentence similarity strength function to determine an appropriate response. In the case, where a user utterance has attracted duplicate responses, the PM engine uses a sentence similarity calculation to execute the highest matched response based on the similarity matched strength value.

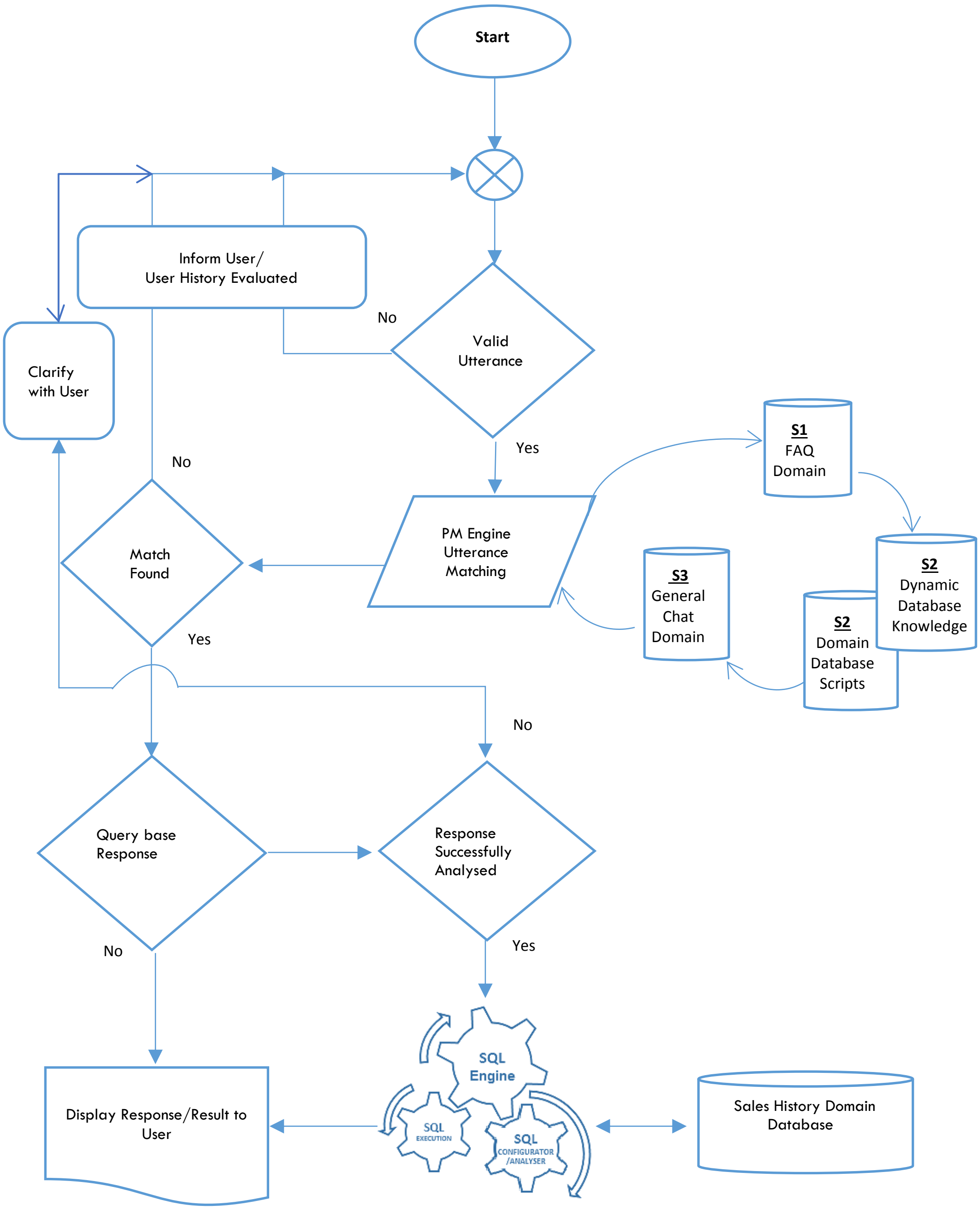


Figure 4.4: High-Level Overview of ANEESAH's functional flow

4.3.3 Sentence Similarity Feature

ANEESAH has been implemented with a sentence similarity technique. The sentence similarity feature utilises a similarity measuring algorithm called Dice Coefficient (discussed in Section 4.3.4). The role of this feature is to determine match strengths between two patterns or word sets. This feature works in conjunction with the wildcard pattern matching process (discussed in chapter 2) to support the PM engine to carry out match process against different domains. The PM engine utilises wild card pattern matching technique to shortlist matched pattern followed by selecting associated rule-based response. In the case where a user utterance leads to duplicate matched responses, the sentence similarity values are used to select a response based on highest matched strength. A sentence similarity strength value can be described as a degree of match between two strings based on calculated numeric value. The match value is measured based upon the number of factors, such as matched keywords in two strings, the number of matched pattern wild cards and algorithm-specific parameters. The main purpose of introducing pattern strength feature is to increase the versatility of the matching abilities of PM engine. The implemented sentence similarity feature replaces scripted patterns by a few natural language sentences in each rule. Applying the sentence similarity technique in CA building is more effective as it reduces the scripting effort to a minimum (Li et al., 2004; O'Shea et al., 2008; O'Shea et al., 2009). Table 4.1 highlights the differences in rule handling for example topic "Payment of fees" obtained from (O'Shea et al., 2010).

S: Example user sentence **P**: Scripted patterns **R**: Rule-based response

A Rule scripts using Pattern Matching	A rule scripts using sentence similarity
<Rule 1>	<Rule 2>
P1: * money	S: I have no money
P2: * cash	R: I'm sorry to hear that
P3: * funding	
....	
P250: * no money	
R: I'm sorry to hear that	

Table 4.1: Rule scripts using PM and sentence similarity

4.3.4 Dice Coefficient Algorithm

The Dice Coefficient matching algorithm is used to determine the similarity between strings and is widely used in sciences, digital library and other fields to determine match differences (Kondrak, 2004). The sentence similarity feature relies on Dice Coefficient algorithm only in terms of determining the match strength values. The Dice Coefficient algorithm works on principle to the derived degree of overlapping, between two-word sets (i.e. user utterance and scripted patterns). Let **A** and **B** be the character bigram sets for the word **a** and **b** respectively (Lin, 1998). Equation 4.1 shows the equation used for similarity measure.

Eq. (3.1):

$$\text{Dice}(a, b) = \frac{2\,|A \cap B|}{|A| + |B|}$$

Equation 4.1: Dice Coefficient Equation

A value of 0 is used to show no overlap or mismatch between two-word sets (i.e. number of words in a user utterance) and a value of 1 shows perfect overlapping or absolute match. The difference between two-word sets is determined based on several bigrams (a pair of adjacent letters in a string). Equation 4.2 shows an example of dice coefficient calculated match value using bigram for two example words “context” words “contact”.

{ co, on, nt, te, ex, xt } = **A**
{co, on, nt, ta, ac, ct} = **B**

$$\text{Dice}(\text{context}, \text{contact}) = \frac{2\,(3)}{6 + 6} = 0.5$$

Equation 4.2: Dice Coefficient similarity match between two words

The calculated match value of 0.5 between word “context” and “contact” reflects a degree of match determined using Dice Coefficient similarity algorithm as shown in Equation 4.2.

4.3.5 Pattern Matching Scripting Language (PMSL)

This section details the new scripting language used by the pattern matching engine that allows users to engage in conversation. The development of NLIDBs have been researched by a number of approaches, which represent associated challenges as discussed in the literature review (discussed in chapter 2).

The PMSL has been developed in order to represent the two separate sections of the implemented knowledge base. The first represents set of rules, which further comprised of scripted patterns and relevant rules based textual responses, whereas, the other section deals with domain database relevant scripts, which lead to query based responses (see detail example in Figure 4.5). A rule can be defined as sub-section of a context that can be the target of a user utterance. A pattern consists of a few words, symbols and spaces. The wildcard character sequence can be used as a substitute for other mentioned characters in each set of words. The patterns have been scripted with symbolic wildcard characters and regular expression operators i.e. a pattern containing (*) can match any number of words. The utterance matching process takes place as follows:

```
1    > START
2      Update Knowledgebase
3      Get user input
4       IF (User input valid = TRUE)
5           Take input forward to controller for processing (GO Step 9)
6       ELSE (Ask user for a valid/relevant input) //Allow user to make three attempts
7             IF (Valid input violated > 3)
8                 END Session
9          Match input across domain contexts – (Database | FAQ | General)
10            IF (Input matched/Response found = TRUE)
11                  ELSE (Ask user to enter a relevant input)
12           IF (Default response found (Non-Query) = TRUE && Match Strength = TRUE)
13                 Execute Rule-based Response (Reset System (0))
14          ELSE IF (Default response found (Non-Query) = TRUE && Match Strength = FALSE)
15              IF (Match found is what was requested) // Check with user
16                  Execute Rule-based Response (Reset System (0))
17          ELSE IF (Input matched database = TRUE) // (Query-based response = TRUE)
18                      List matched query syntax   // (Keyfields, Attributes, Functions, Filer etc.)
19                            <Analyse query syntax>
20                             IF (User Intention Exists = TRUE) // Select Statement
21                                     ELSE (Ask user to rephrase/or further clarification)
22                             IF (Database Keyfields Exist = TRUE || Database Attributes Exist = TRUE)
                               // Table, Column names, cell level information etc.
23                                     ELSE (Ask user to rephrase/select/clarify database information)
24                             IF (Function Exist(s) = TRUE) //Aggregation function or sub-function etc.
25                                        <Analyse excessive use of syntax/function in input>
26                                            IF (Excessive Syntax Used = TRUE)
27                                                  {Ask user to make selection}
28                                            ELSE (Move to the next step) (GO Step 30)
29                             ELSE (Move to the next step)
30                      IF (Minimum Query Formulation Condition Met = FALSE)
31                              {Ask user to provide missing information to meet minimum condition}
32                     ELSE (Minimum Query Formulation Condition Met = TRUE)
33                             SWITCH (SQL Query Generator)
                               // Query Types engineered based on complexity
34                            CASE: Query Type 1
35                            CASE: Query Type 2
36                            CASE: ……………………
37                            CASE: Query Type n
38                            CASE:  Query Type
39                                   {Collection of query related tables
40                                    Formalise database Keyfields and Attribute
41                                    Formalise functions and sub-functions
42                                    Formalise appropriate joining & result filter}
44                       EXECUTE (SQL Query);
45             IF (SQL Query Results = TRUE)
46                   Display (Display (0))        //   Reset Session (0)
69            ELSE Reset Session (0)            //   Treat new user input as new request
70      END;
```

Table 4.2: Algorithm to for utterance processing and query formulation

The next section (4.3.5.1) will highlight a high-level utterance processing flow to demonstrate how ANEESAH follows a predefined step by step process in assisting users in accessing desired information stored in a domain specific database. Following up from a user utterance failing to attract/match an appropriate response, ANEESAH tries to build a query based response manually. ANEESAH's knowledge base responsible for providing patterns to aid manual query-based response building are scripted using Regular Expression approach, discussed in next section.

4.3.5.1 Regular Expression based Pattern Matching

The scripting for patterns for a domain has been highlighted as a laborious task (Latham et al., 2010). ANEESAH utilises an information matching feature based on regular expression pattern scripts, also implemented in another similar application (Latham et al., 2011). This feature has been implemented as part of the scripting language. A regular expression is a pattern describing a certain amount of text. A regular expression, which is a programming feature that has been designed to work in functional languages, is derived from pattern matching approach and utilises wild card characters for the assembly of patterns (Hosoya et al., 2005). Regular expression types are used as a natural generalisation of text definition and description of text structure.

The regular expression pattern matching has been used for utterance processing to build manual query response. To develop a flexible and robust approach for analysing and matching table level information in the database, regular expression operators are used in pattern scripting of the knowledge base. The implementation of regular expression pattern matching not only provides benefit in intelligently extracting matches from user utterances, but it also reduces the need for excessive scripting. For example, Figure 4.5 is a simple pattern that will match "city", "citys", "cities", "town", "towns" in an utterance.

Example 1:

Figure 4.5: Example pattern with regular expression

The use of regular expression has allowed the development of reusable pattern scripts to construct the full domain database knowledge. The database columns and tables information is referenced by using named groups. A named group is the name of the group reflection specific database key information (i.e. column name, table name, etc.) and the pattern is the regular expression in the group itself. The atomic grouping feature of regular expression has also been used to increase the efficiency of ANEESAH in performing pattern matching.

4.3.6 Utterance Processing Flow

To find an appropriate match, a user utterance is matched across all knowledge base contexts (Domain Database Scripts, FAQ Context and General Chat Context) in the flow of conversation. Once a match has been found, ANEESAH displays a relevant response back to the user. In addition to the utterance matching in the knowledge base, ANEESAH offers conversation based assistance to the users for directing conversation towards achieving their desired information. For example, if a user utterance is not very specific, ANEESAH clarifies ambiguities raised during user utterance processing. Figure 1.9 gives an overview of workflow of ANEESAH. After receiving an example utterance “total profit in manchester”, the controller initiates the matching process by transferring the utterance to the PM engine. The user utterance is matched against rule-based and non-rule-based patterns stored in the knowledge base. However, if a query based response is matched in the system knowledge base, user utterance is fully analysed before a query is formulated. For a non-query response match, a text-based response is shortlisted and displayed to the user. As shown in Figure 4.6, ANEESAH can handle and control various outputs arising from the utterance matching process such as a poorly structured sentence, conflict resolution or an utterance failed to match a response.

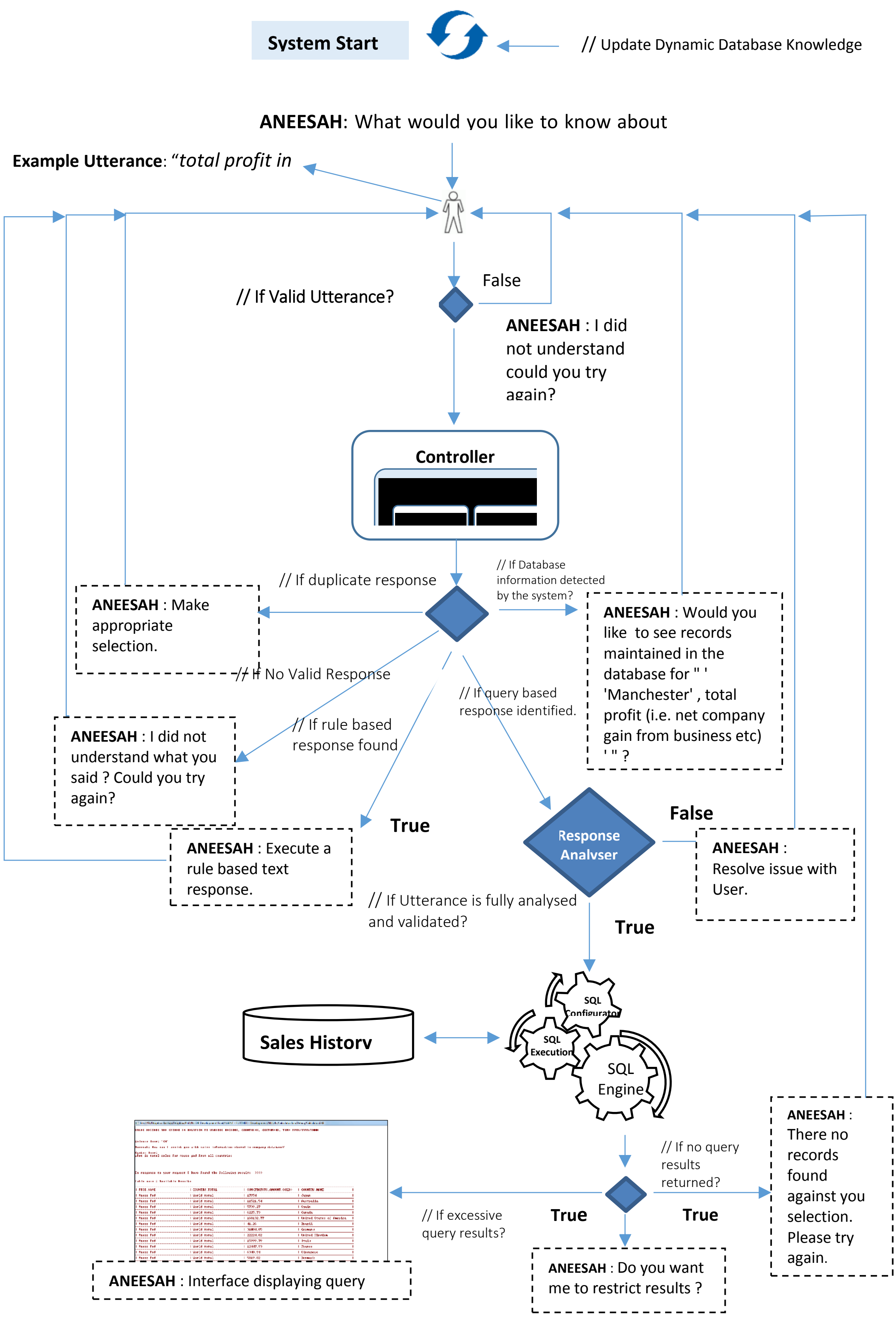


Figure 4.6: Overview of user utterance flow in the ANEESAH

The next section (4.3.7) will explain a novel matching technique used to capture database information reflected in user utterances.

4.3.7 Utterance Processing for Database Information Mapping

In order to develop a practical NLIDB system for a real-time environment, it was necessary to contrive an information capturing approach that will allow ANEESAH to perform user utterance matching against information maintained in the pre-selected domain database. In this work, an Oracle sample database comprising sample records was selected for evaluation and is described in Section 4.4.1. When a user utterance fails to attract/match an appropriate rule-based response, this layer based matching process is used to map user input across database relevant information. This process is used to build a manual query response in different parts (referred as layers). The following information discovery model (Figure 4.7) represents the flow of utterance matching process.

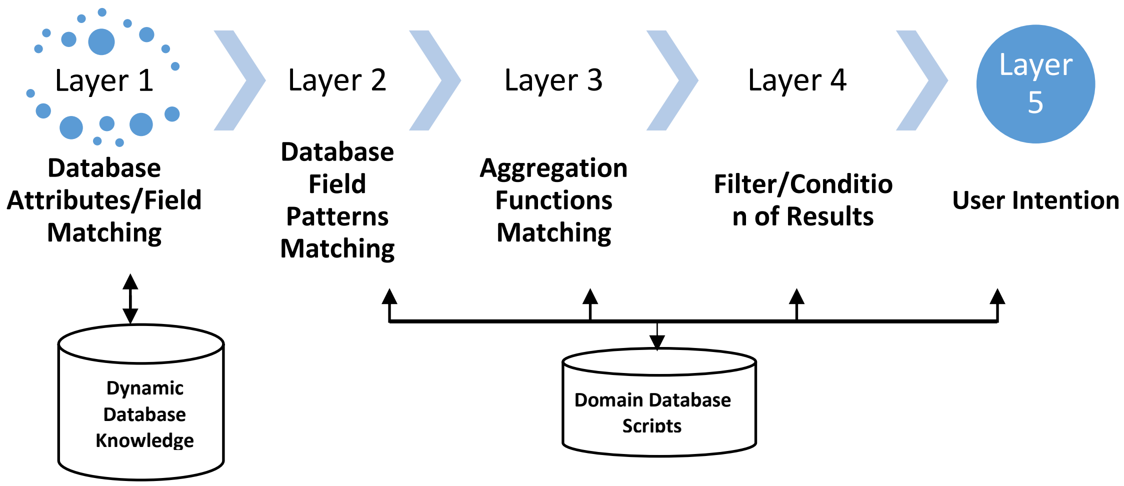


Figure 4.7: Layer-based Database Information Mapping/Discovery

Layer 1:

Layer 1 involves matching of each user utterance containing words against domain specific database information. The database information is pulled into Dynamic Database Knowledge (DDK) module, which is a part of the knowledge base and serves as a container. The DDK module makes available selective database information updated at system run times, to allow attribute and key information level matching. The database metadata (column names, table names, data type) is also loaded and matched against a user utterance. The Domain Database Scripts working together with DDK component helps in capturing database relevant information from the users' utterances. The both components Domain Database Scripts and Dynamic Database Knowledge are explained in detail in knowledge engineering section 4.4.2.

Let us consider the following user utterance as an example:

Example User Utterance: "*who are our top* ***five*** *best customers from '****Spain'*** *based on total purchase orders for '****mouse pads'*** *in '****1999'****?"*

At the first layer level, example utterance is checked against database information in DDK module of the knowledge base, in order to capture any co-occurrences of an attribute or key information. As shown in Figure 4.8, the attribute ("spain", "mouse pad", "1999") are recorded in the developed system's temporary memory, and user utterance is altered by replacement of matched attributes with corresponding database fields, for further processing.

COUNTRY_ID	COUNTRY_NAME	C
52790	United States of ...	No
52789	United Kingdom	We
52788	Turkey	We
52784	The Netherlands	We
52778	Spain	We
52791	South Africa	Afr

"**spain**" is matched in countries table

PROD_ID	PROD_NAME	
28	Unix/Windows 1-user pack	Ur
29	8.3 Minitower Speaker	8.
30	Mouse Pad	M
31	1.44MB External 3.5" Diskette	1.
32	Multimedia speakers- 5" con...	M
33	PCMCIA modem/fax 19200 b...	PC

"**mouse pad**" is matched in products table

TIME_ID	CALENDAR_YEAR	DAY_NAME
31-OCT-99	1999	Sunday
01-NOV-...	1999	Monday
02-NOV-...	1999	Tuesday
03-NOV-...	1999	Wednesday
04-NOV-...	1999	Thursday
05-NOV-...	1999	Friday

"**1999**" is matched in times table

Figure 4.8: Matched attribute values in database tables

ANEESAH generates an identical surrogate user utterance before applying any alteration.

Altered Example Utterance at Layer 1: "who are *our top* **5** *best customers from* **Country_Name** *based on total purchase orders for* **Prod_Name** *in* **Calendar_Year***?"*

The example utterance is altered to simplify matching process at next layer levels. Any numerical value mentioned in user requests is also captured.

Layer 2:

This layer refers to the matching of example utterance against scripted patterns (explained in section 4.4.3) stored in Domain Database Scripts module of the knowledge. The domain database scripts module contains patterns relevant to the sales history database (i.e. key information, column and tables information, etc.). Following is the matched information (highlighted grey) from example utterance that corresponds to the database fields.

Altered Example Utterance: "*who are our top five best* ***'customers'*** *from* Country_Name *based on total* ***'purchase orders'*** *for* Prod_Name *in* Calendar_Year*?"*

Matched Pattern:	^(?=.*\b((?:customers?|clients? |consumers?|accounts? holders?|buyers?|purchasers?))\b).*$

The patterns have been scripted for all domain database schema fields including table names, column names, selective information stored in columns. Therefore, information could be asked across the whole of the used sample database and in any combination of objects. The matched part of the user utterance is replaced with the database fields.

Example Utterance at Layer 2: "*who are our top* 5 *best* Cust_Id *from* Country_Name *based on total* Amount_Sold *for* Prod_Name *in* Calendar_Year*?"*

Layer 3:

The user utterance in this layer is matched to detect any aggregation function request. ANEESAH is equipped to formulate SQL queries with number of aggregation functions i.e. Sum(), Count(), Avg(), Max(), Min() etc. The knowledge base is equally developed with aggregation function patterns. Once an aggregation function is matched from user utterance, it is included in existing syntax collection to become part of query structure.

Altered Example Utterance: "*who are our* **top** 5 ***best*** Cust_Id *from* Country_Name *based on* ***total*** Amount_Sold *for* Prod_Name *in* Calendar_Year*?"*

In the case of above example utterance, following is the syntax matched as part of the aggregation function to be included in the query.

Example Utterance at Layer 3: "*who are our* top <u>5</u> (**Oder by + Desc**) <u>Cust_Id</u> *from* <u>Country_Name</u> *based on* (Sum ()) <u>Amount_Sold</u> *for* <u>Prod_Name</u> *in* <u>Calendar_Year</u>*?"*

Matched Pattern:	^(?=.*\b((?:best|top|most valueable|valueable|value able|effective|highest))\b).*$

Layer 4:

Layer 4 is designed to analyse user utterances for any condition or a specific selection of results. The developed system allows users to restrict results returned by queries (i.e. first ten, last five, etc.). The system also features a method to prompt users with excessive results, before displaying them in the system's user interface. ANEESAH utilises this feature in a situation where data rows returned by a query will not fit in user interface screen (e.g. if query result rows were in the thousands). The users in such scenarios can ignore or restrict results to their desire before query results before viewing in the user interface. The system does this by displaying the number of results returned by the query and offers the user a chance to restrict the result to his/her desire. However, in a situation where the user explicitly requests conditional results, ANEESAH does not offer restriction of results because of the pre-existing selection choice. The query result condition applied in example utterance is "top 5" rows, which is recorded in existing collection of syntax, captured in earlier layers.

Example Utterance at Layer 4: "*who are our* RowNum <= '5' (**Oder by + Desc**) <u>Cust_Id</u> *from* <u>Country_Name</u> *based on* (<u>**Sum ()**</u>) <u>Amount_Sold</u> *for* <u>Prod_Name</u> *in* <u>Calendar_Year</u>*?"*

Layer 5:

The user intention is checked at final layer to ensure that user intention to visualise a query based response is present. The intention of the user is matched against scripted intention patterns, which refers to "Select" syntax to be part of response query.

"<u>Who are our</u>" from the example utterance will correspond to ∞ "SELECT" statement. The example utterance can now be broken down into segmented syntax to understand query logic.

Example Utterance at Layer 5:

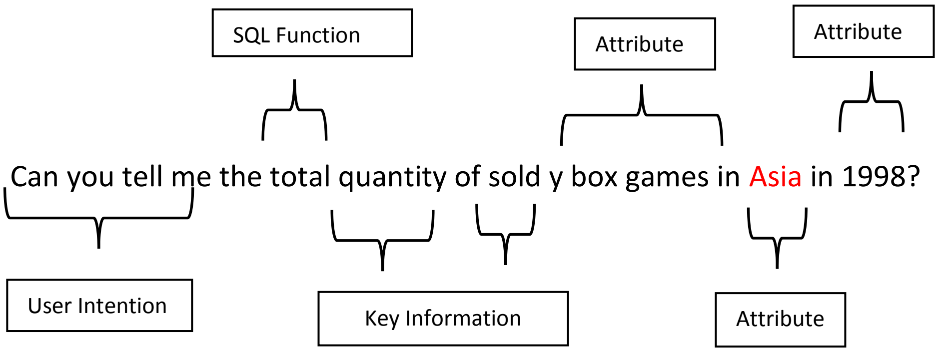
SELECT	*// Select records from the database*
ROWNUM <= ‘5’	*// restrict them by five rows*
(ODER BY, DESC)	*// sort records by descending order*
CUST_ID from COUNTRY_NAME	*// customer results from country table*
Based on (SUM ()) AMOUNT_SOLD	*// aggregation function to transaction field*
Mouse Pads PROD_NAME	*// include specific product with condition*
In CALENDER_YEAR?”	*// selection based on a specific year*

Next, the response analyser component (explained in section 4.3.9) is activated by the PM engine for the user utterance to be fully analysed, before a final response (in the form of a database query) is generated. ANEESAH is equipped with conversational features to interact with the users actively. The implemented conversational features extend ANEESAH’s ability to engage with the user to formulate queries from multiple dialogues. The conversational features are aimed at handling a number of inconsistent situations that might arise during the response formulation process i.e. resolving ambiguities, non-existing information related requests, duplicates database records handling, etc. These features are explained in more detailed in the following sections of this chapter.

4.3.8 Conflict Resolution

This feature enables ANEESAH to perform conflict resolution for issues that might arise during conversation with the users (i.e. duplicate product names, duplicate street names, multiple query functions in a request, etc.). A database can contain duplicate records, which might represent or refer to different information. During the utterance matching process, ANEESAH is equipped to detect and capture the duplicate occurrence of database information. In a situation where duplicate records are detected, ANEESAH informs the user about duplicate matches and allows the user to make appropriate match selection for his/her desired query. Let us consider the following user utterance as an example (Example 1).

Example 1 user utterance:



The attribute “Asia” from above example (Example 1) utterance is an attribute that is detected as a duplicate match in the system, as shown in Figure 4.8.

	COUNTRY_ID	COUNTRY_NAME	COUNTRY_SUBREGION	COUNTRY_REGION	COUNTRY_TOTAL	COUN
1	52791	South Africa	Africa	Africa	World total	ZA
2	52769	Singapore	Asia	Asia	World total	SG
3	52771	China	Asia	Asia	World total	CN

Figure 4.9: Example duplicate records maintained in database

The attribute “Asia” exists in different columns (COUNTRY_SUBREGION and COUNTRY_REGION) of the database table (COUNTRIES), which requires clarity from the user, as random selection would lead to ambiguous or incorrect query results. In this situation (as shown in Figure 4.10), ANEESAH displays the users with duplicate matches captured in its temporary memory with match description to help the user in making the correct decision.

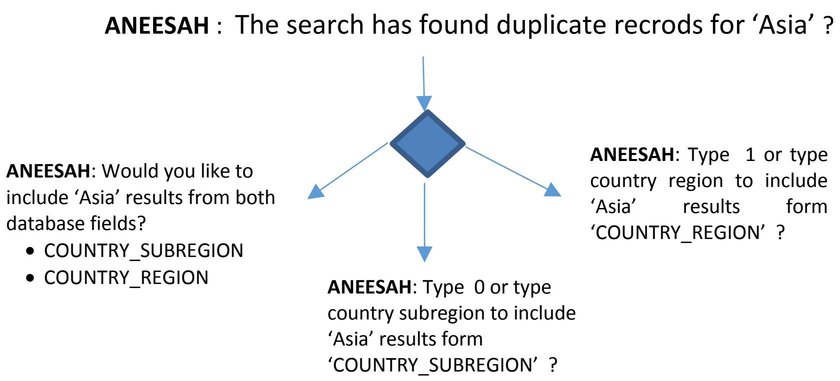


Figure 4.10: Example duplicate records match situation, choices offered to the user

ANEESAH is implemented with this feature not only to handle duplicate matches relevant to database information, but it also detects and deals with user requests containing excessive information (beyond query formulation abilities of ANEESAH). For example, ANEESAH deals with utterance containing more than one aggregation function (such as Sum, Avg, Max) to be included in a single query, by way of asking user to select one function at a given time. An utterance containing excessive database fields is also interactively clarified with the user before a query response is formulated.

4.3.9 Response Analyser for Query-based Responses

ANEESAH has been implemented with a novel response analyser to work with SQL query based responses. The role of the response analyser is to ensure that each user utterance, shortlisted for the query based response, is fully analysed. The users' utterances are forwarded to response analyser by the controller before the SQL query formulation process is initiated, as shown in Figure 4.11 below.

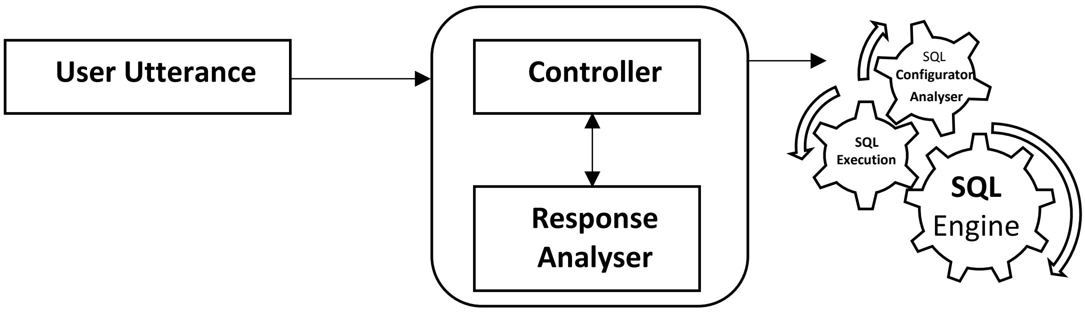
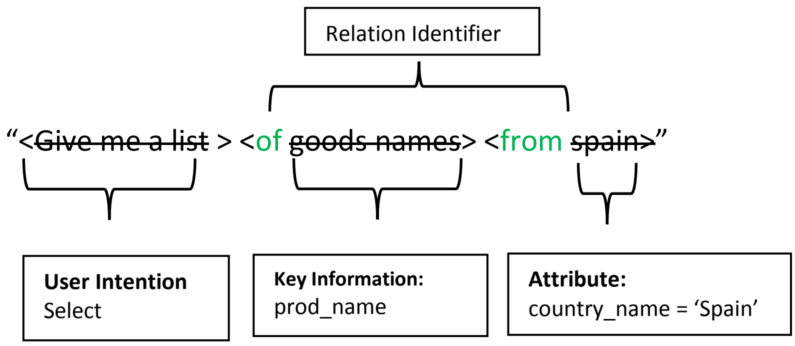


Figure 4.11: Response analyser working flow

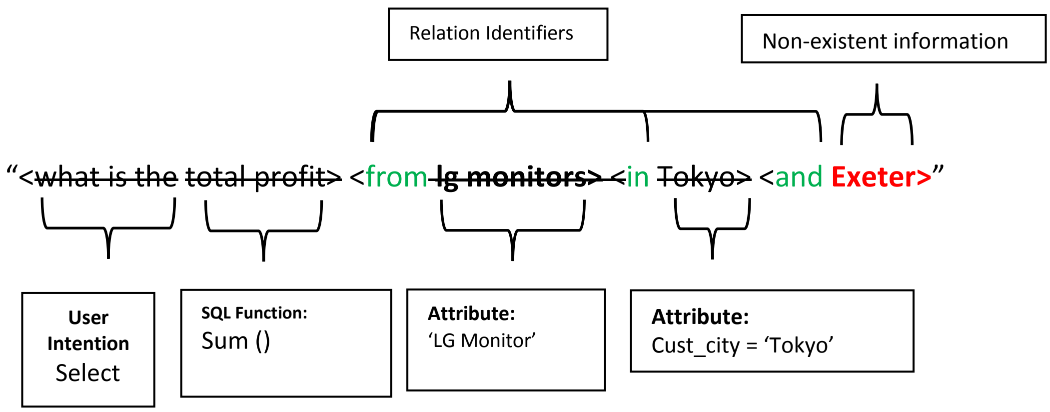
The response analyser performs validation of user utterances. The user utterance is analysed to address the most pressing problems i.e. non-existing information requests, excessive information requests, unknown requests, etc. Let us consider the following as an example (Example 2) user utterance ("***Give me a list of goods names from spain***"), which meets all utterance validation requirements for the response analyser to trigger the controller to proceed with SQL query formulation processing.

Example 2 - user utterance:



Where a user utterance is not fully validated, the response analyser reacts by interacting with the user to resolve it (i.e. user utterance might contain non-existent information). Let us consider the following (Example 2) as a sample user utterance (“***what is the total profit from lg monitors in tokyo and exeter***”), which does not meet utterance validation requirements.

Example 3 - user utterance:



In Example 2, the user utterance has partially matched database relevant information. ANEESAH offers matched records and makes further enquiries to the user about non-existent information (e.g. “Exeter”). The user is asked to either rephrase non-existing information or drop it from his/her request and proceed to view matched/available database records.

4.3.10 Conversation Manager (CM)

This component manages the direction of conversation towards achieving the user’s desired information. The CM also ensures that the user stays relevant to the conversation topic and ensures that required information is made available to the developed system before it executes any response. The conversation manager controls the conversation in line with the activated domain context and allow users to switch topic or contexts, as and when necessary. The use of repeated words is common in a formal or informal conversation. In aiding query formulation from multiple dialogues, the conversation manager has the ability to ignore repetition of words during a conversation with the user. The conversation manager does this by verifying the existing collection of matches in system's temporary memory. The conversation manager works closely with other components of ANEESAH, to handle inconsistent conversational situations i.e. requests to produce nonexistence information, ambiguities in users' requests, duplicate handling etc. Figure 4.12 explains the role of conversation manager in directing the user in the given conversation to produce his/her desired database results from multiple dialogues.

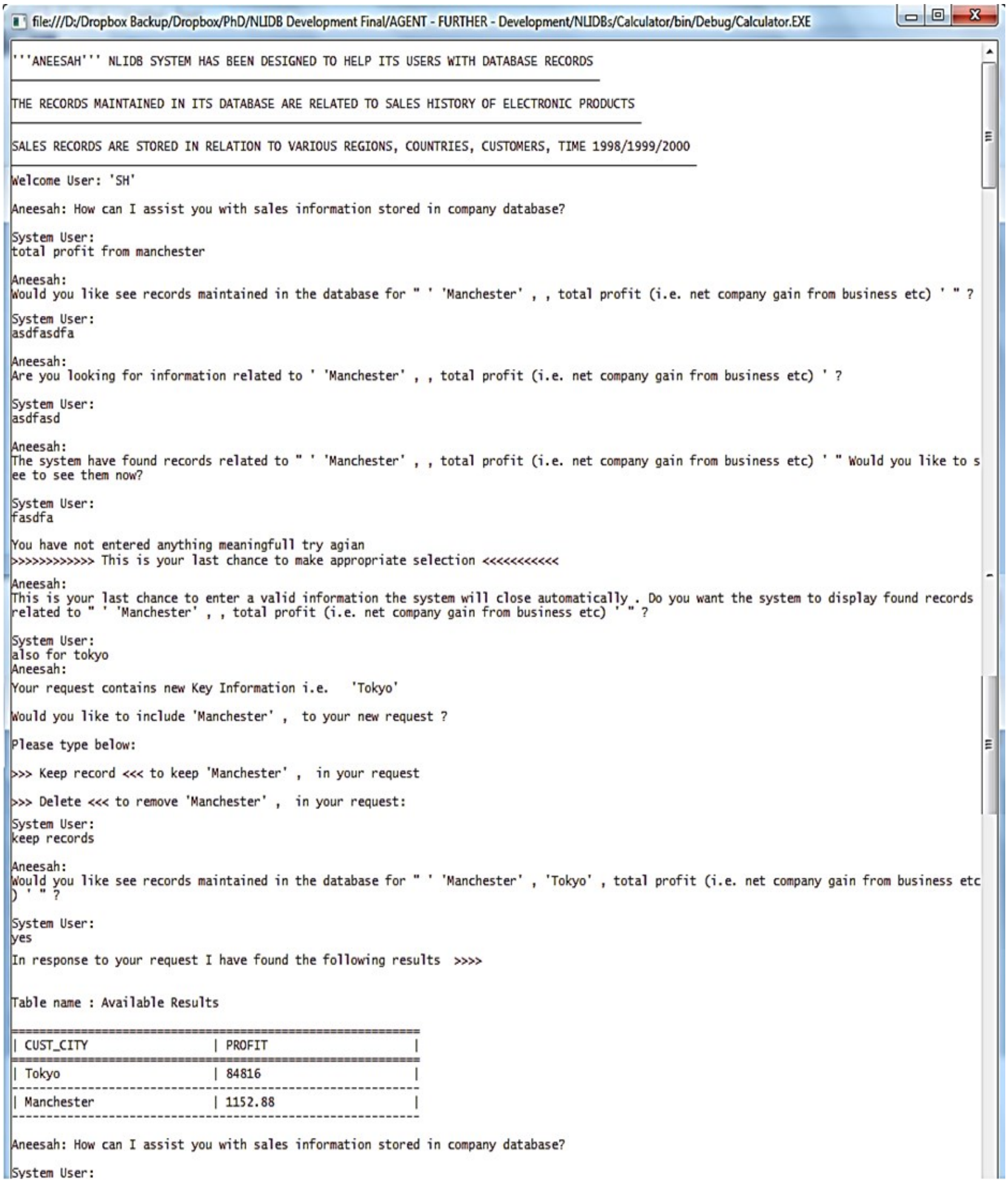


Figure 4.12: Conversation handling example in the user interface

4.3.11 Temporal Memory

ANEESAH features a temporary memory component to store a variety of information during the conversation. The temporary memory component works closely with the system log file providing assistance to the system during the conversation and in extracting database results. ANEESAH uses temporary memory component during conversational sessions with the users for the following:

1. The user utterance received by ANEESAH system is stored in the database before its normalisation (i.e. cleaning, validation, etc.). The use of inappropriate, rude or offensive words are matched against scripted rude word context in the knowledge base. A user utterance with the rude word is highlighted as a violation of rules, at which point the user is shown with a warning message. In the case of repeated use of rude words, ANEESAH refers to counter in temporary memory to analyse any repeated violation of rules.
2. Upon matching the user utterance is categorised to a relevant context in the system's knowledge base. The matched context (Database domain, FAQ domain or General Chat domain) is activated in system's temporary memory to allow the controller to direct conversation with the user. During user utterance processing, any database relevant information captured (key information, attributes, database objects, etc.) at runtime, is retained in temporary memory and stored in the system log file.
3. ANEESAH is equipped to process multiple utterances leading to one database query, which requires it to remember information exchanged during a conversation with the users. The temporary memory component provides assistance to the system in staying relevant to the topics and sustaining a conversation with the users by way of retaining variable information (i.e. database information, user chat history, rules, responses, etc.).

4.3.12 Log file

The implemented log file helps ANEESAH by storing events, time & date stamps, unique session ids, user utterances, system's responses, and domain activation, captured information, fired rules, correct responses, incorrect responses, formulated queries, system breakdown and failed responses, etc. The log file module helps ANEESAH, where necessary, refer to the historical information stored in log file i.e. calling previously executed response, etc. This component also helps in carrying out statistical analysis such as number of utterances recognised by the system and not recognised or failed responses. The information recorded in the log file is aimed at aiding in future improvements and developments of ANEESAH.

4.3.13 User Interface

ANEESAH is implemented with command line based user interface, which will be replaced with a graphical interface at next stage development. The user interface is responsible for serving as an interaction platform for the user and ANEESAH system. The user interface receives user utterances and displays back responses in the interface window. The development of the information displaying technique in the command line interface has made it possible to display variable system responses in all forms i.e. text, quoted words/phrases, tables, query results, etc. Also, the user interface supports the presentation of query results in tabular structures.

A well-structured command line user interface possesses several advantages, when compared with others. Command line interfaces are easy to learn, less complex, faster in processing information, simplicity and uniformity make it easy to use (Pazos R. et al., 2013). The Figure 4.13 shows the ANEESAH's interface interacting with a user.

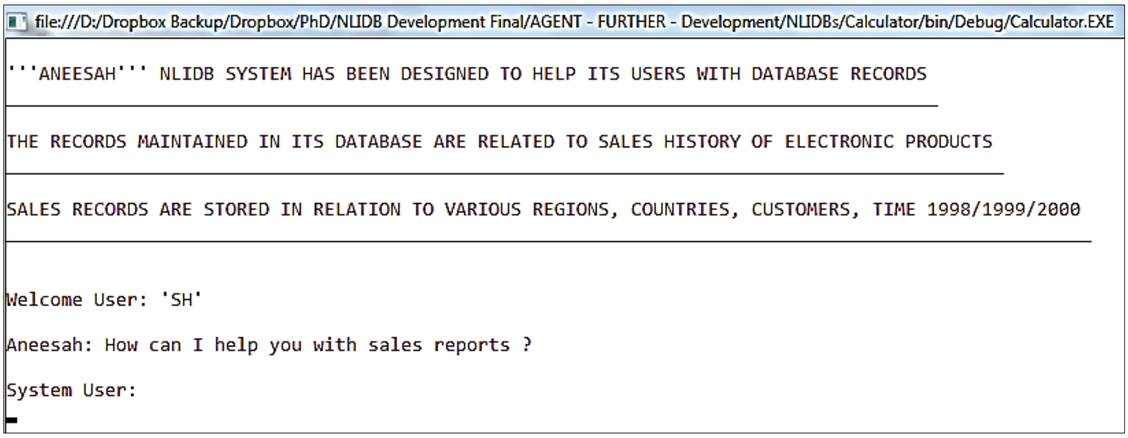


Figure 4.13: ANEESAH's User Interface

The next section (4.4) will provide detail on the knowledge base structure and related components.

4.4 Knowledge Engineering the Domain (Component 2)

A relational database in a commercial environment often represents a domain with complex structure and storing millions of records. The records maintained in an organisation's database are used for various purposes and can be classified into a variety of layered information. The layered information can be further divided into different types such as master data, transaction data, enterprise information, and database objects/metadata (Elmasri and Navathe, 2010). Each type of data stored in a database reflects individual informational roles, for example, master data refers to the key business entities i.e. product descriptions, customer details, operating countries, etc. The transaction data defines business events i.e. quantities sold, the number of sales, profit, loss, etc. Similarly, metadata information refers to the structure of information models i.e. a number of tables, columns, fields, etc. The following Figure 4.14 (Sapinsider.wispubs.com, 2017) shows the scale of complexity involved in real life organisational database structures. The nodes and spheres in above figure (4.14) represent informational roles in a database and denote relationship among one another.

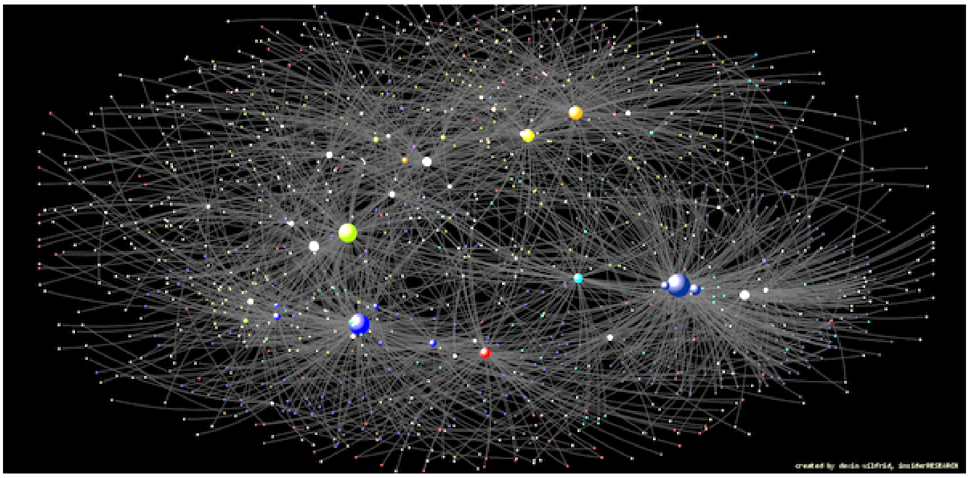


Figure 4.14: Information/Data universe of an example organisation

The spheres in the middle take the main relational point of individual information models or schemas, which might represent individual departments (i.e. sales, finance, accounts, production etc) in an organisation (Sapinsider.wispubs.com, 2014). In a real life organisational environment, the information is often analysed by users working at different levels. To access and analyse database stored information requires knowledge of SQL language. Complex information analysis is a challenging task as users not only require knowledge of SQL query language but also the knowledge of database schema.

A conventional knowledge engineering technique was adopted to script the domain database. This information modelling technique relies on capturing, representing, encoding and evaluation of expert knowledge (Chu et al., 2008). ANEESAH has been equipped with the expert knowledge to lead the user in a conversation towards generating a relevant response based on multiple dialogues. The knowledge base for ANEESAH has been implemented with a generic approach in mind to allow naïve users to interactively retrieve and analyse information stored in chosen database for the developed system.

4.4.1 Adapting a Domain Database for System Evaluation

The domain database (Sales History database) used for the prototype system's evaluation plays a fundamental role in ANEESAH NLIDB as it serves as main information retrieval source during conversation sessions with the users. The sample database has been selected based on the ideal properties for ANEESAH's evaluation, discussed in chapter 2. Oracle plc had developed the database used for ANEESAH's evaluation. The chosen database comprises complex schema structure with records of large data related to Sales History (SH) records (Oracle.com, 2014).

The short-listed SH schema for ANEESAH is particularly designed to demonstrate a large amount of data in complex relational structure (Docs.oracle.com, 2014). The structure of the SH schema has been developed with sample sales records of electronic products belonging to an assumed company in view, which maintains a high volume operating business. The business operating information is utilised by the company to perform business analysis from multiple database tables to aid in decision making. The SH database information can be used to run multidimensional analysis on sales trends, temporal reports, geographical reports, products analysis etc. The structure of SH database comprises 8 database tables with 114 columns containing domain relevant information. Figure 4.15 (Docs.oracle.com, 2014) below illustrates the full structure of the SH database schema:

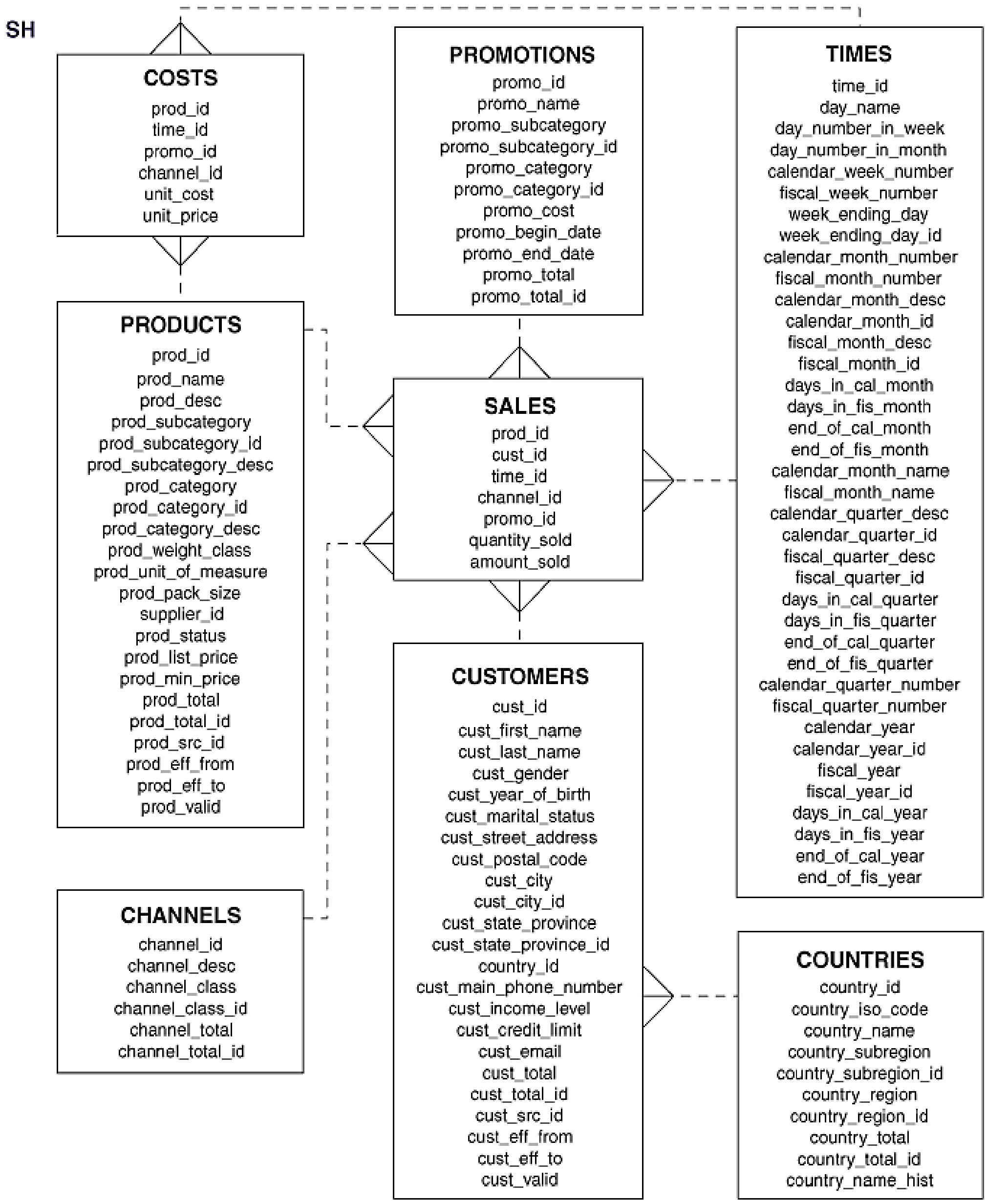


Figure 4.15: Sales History Database used for ANEESAH's Evaluation

The SH database is installed using the documentation provided by the Oracle company.

The Figure 4.16 below shows an example data tuple in chosen domain database.

	COUNTRY_ID	COUNTRY_NAME	COUNTRY_SUBREGION	COUNTRY_REGION	COUNTRY_TOTAL	COUNTRY_ISO_
1	52773	Argentina	Southern America	Americas	World total	AR
2	52774	Australia	Australia	Oceania	World total	AU
3	52775	Brazil	Southern America	Americas	World total	BR
4	52772	Canada	Northern America	Americas	World total	CA
5	52771	China	Asia	Asia	World total	CN
6	52777	Denmark	Western Europe	Europe	World total	DK
7	52779	France	Western Europe	Europe	World total	FR
8	52776	Germany	Western Europe	Europe	World total	DE

Figure 4.16: Sample table records stored in the Sales History database

4.4.2 Knowledge Base Structure for the Scope of Conversation

The implemented knowledge base structure for ANEESAH is comprised of three contexts namely: Database context for SQL query based responses, Frequently Asked Questions context which deals with database structure related questions, and General context for off the topic chat with the users. The scope of ANEESAH's conversational abilities is not only restricted to simulating an SQL query expert but also allowing users to interact with ANEESAH freely, in natural language. The knowledge contexts are scripted with relevant patterns to enable ANEESAH to react with appropriate responses. The knowledge base was constructed by a critical review and research on existing NLIDB systems and commercial enterprise reporting systems (i.e. Oracle, Microsoft, SAP, ASAP, etc.) to take an in-depth understanding of information management and knowledge engineering. An in-depth study was also conducted in relational and non-relational databases (i.e. IBD DB2, Oracle, Dynamo DB, etc.) for the purpose of adopting a comprehensive knowledge base approach. ANEESAH uses an implemented knowledge base to support the user retrieving his/her desired information conversationally. The Figure 4.17 below is an example of implemented knowledge base structure in ANEESAH.

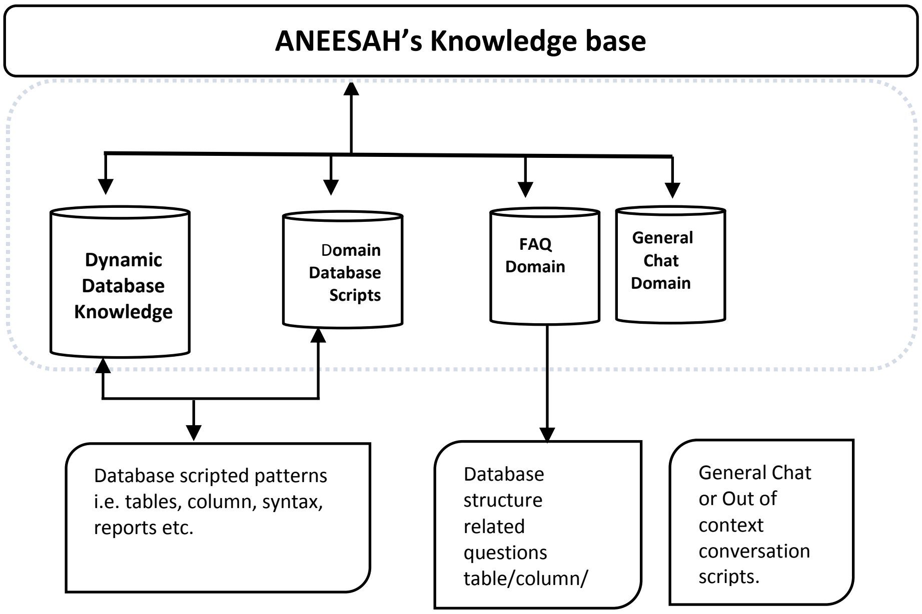


Figure 4.17: ANEESAH's knowledge base structure

4.4.3 Domain Database Scripts

The Domain Database Scripts module plays fundamental role in the developed system by housing domain database relevant scripts and dynamic database information for the utterance matching process. The domain database scripts repository is responsible for assisting the PM engine to perform user utterance mapping against database information (i.e. key information, attribute information, database tables information etc) to extract database relevant information. The domain database scripts are also responsible for providing selective query syntax to aid in formulation of database queries.

4.4.4 Frequently Asked Question (FAQ) Domain

FAQ contexts handles user questions related to the database structure itself for example, a user might want to find number or name of database tables, description of tables, or types of information stored in tables, or information about store/company etc.

4.4.5 General Chat Domain

The General context handles user utterances outside database relevance for example user might want to talk about football, weather, etc. In this situation, the system attempts reply briefly on topic, then motivate the user to return to the process of query generation.

4.4.6 Dynamic Database Knowledge (DDK)

Dynamic Database Knowledge (DDK) is a key component of the ANEESAH's knowledge base. The DDK component has been developed to allow PM engine to map and capture database relevant information during the utterance matching process. The information maintained in the sales history database, used for ANEESAH's evaluation, is dynamically loaded into the DDK module on execution of the system. During conversation with the users, each utterance is screened against extracted database information (i.e. master data information, database metadata, and selective transactional data). The DDK module allows the PM engine to perform matching of data records at all times, and releases/frees the actual database from being reserved for match processes. The database information extracted and stored in the DDK module is sessional, which is deleted and reloaded with most up to dated information, upon each execution to aid real time analysis. Figure 4.18 shows the flow for prepopulating DDK module with updated database information.

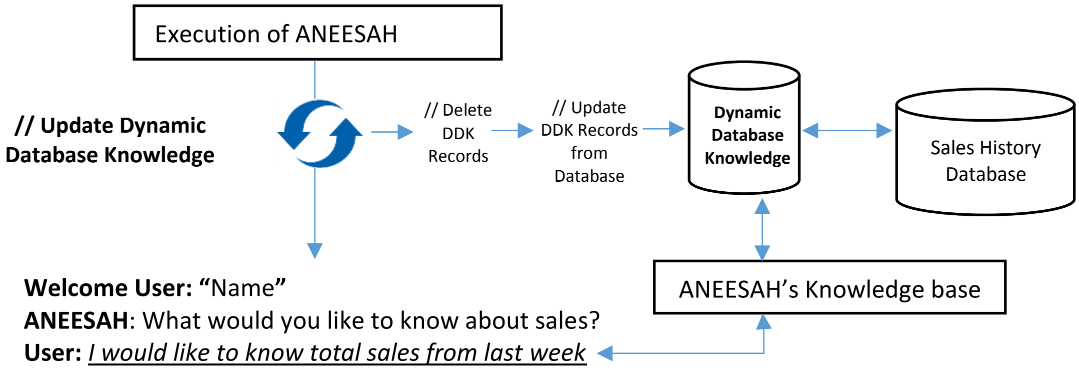


Figure 4.18: Dynamic Database Knowledge update process

In real world scenarios, database information (information tables, database objects, database structure, etc.) can change with the passage of time. The implementation of the DDK module eliminates the need to make simultaneous changes, as it updates and houses the most up to date database information upon start-up. The DDK module makes ANEESAH effective in understanding users' requirements and broadens its database knowledge to produce correct responses. The database information loaded into the DDK is selective and contains characteristics information (i.e. products description, customer details, operating countries, etc.).

4.4.7 Domain Grammar

The database grammar is a key component of Domain Database Scripts and DDK, maintained in the system's knowledge base. The domain grammar contains the classification of users' requests. The questions are often started as interrogative sentences, but statements comprise of a collection of words assembled to yield a meaning. For example, a user utterance can have situation leading to multiple response scenarios i.e. a question-based utterance can trigger SQL query based or non-query based response from the knowledge base. The database grammar provides information related to a language specific grammar i.e. exclamatory sentences, question-based sentences, etc (Shaalan et al., 2009). ANEESAH has been implemented with a library of database grammar features which help it to understand user utterances. User utterances are categorised with the help of implemented classification strategies specifically designed to analyse user utterances intelligently. The Response Analyser components also work in conjunction with domain grammar and other system components to perform utterance analysis. This aids the system to identify correct responses and to strengthen conversational abilities.

4.4.8 Knowledge Engineering the Domain for Query Scenarios

The knowledge engineering for the prototype system was carried out by researching query formulation techniques and leading commercial business intelligence reporting systems used in different organisations. Most modern enterprise resource planning systems deliver department-specific reporting suites, which consist of pre-designed SQL template based reports. A number of sales reports were studied in the interest of understanding business process mapping. The structure of sales reports was also analysed in contrast with the common SQL query syntax used to generate information from the database. The research conducted on sales reports and complex query structures helped in constructing SQL query formulation features for ANEESAH. The gathered knowledge was also used in scripting ANEESAH's knowledge base by adopting a keyword based approach for identifying SQL query syntax from user utterances. The system detects SQL query relevant syntax corresponded in user utterance.

However, a simple user utterance can lead to the requirement of a complex query structure. The users in their utterances often (are unknown of the database structure or system limitations) and do not mention all relevant syntax (aggregation functions, joining keys/tables, etc.) required to complete a query structure. Therefore, where other NLIDB fail in such situations, ANEESAH's knowledge base has been developed to work with minimal query syntax information. The users are not required to express detailed query syntax notions in their requests.

ANEESAH has the ability to autonomously analyse, explore and include required query syntax required when formulating complex queries. Suppose the user input is "*show me top two countries for extension cables sales*" (shown in Figure 4.19).

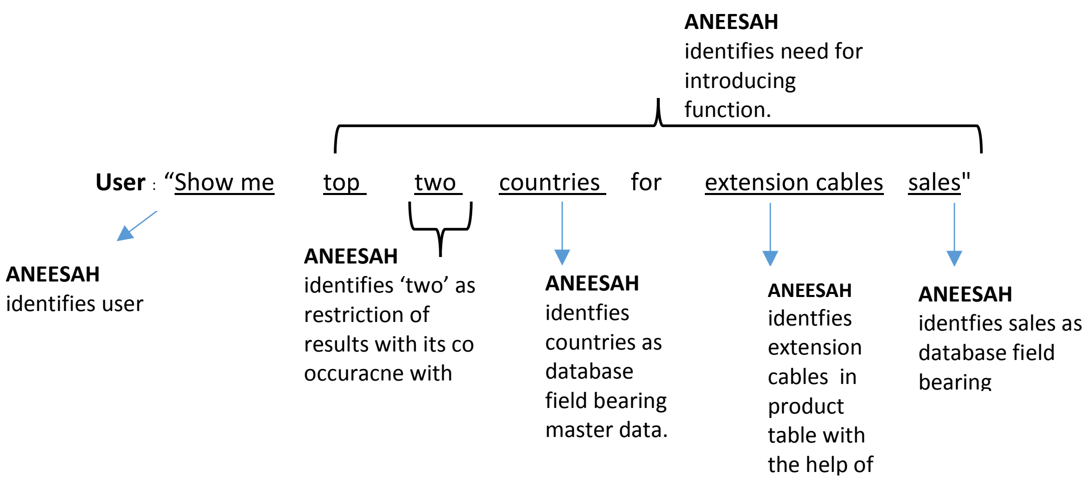


Figure 4.19: Query required syntax extraction from user utterance

The information from utterance is matched in a layer-base approach as discussed in section 4.3.6. For the given example utterance; Table 4.3 shows required query syntax to successfully produce database results and formulated SQL query by ANEESAH.

"show me top two countries for extension cables sales"					
Database Fields	**Attributes Values**	**Aggregation Functions**	**Filter**	**Schema Tables Require**	**Table Joins Required**
Prod_Id Cust_Id Country_Id Prod_Name Amount_Sold Country_Name	'Extension Cable'	Sum(Amount_Sold) Group by Order by Desc	rownum <= 2	Profits Customers Products Countries	3

ANEESAH system generated query for example utterance:

"select * from (select products.prod_name, sum(amount_sold), countries.country_name from countries
join customers on countries.country_id = customers.country_id
join profits on profits.cust_id = customers.cust_id
join products on products.prod_id = profits.prod_id
WHERE PROD_NAME = 'Extension Cable'
group by products.prod_name, countries.country_name
order by sum(amount_sold) desc) where rownum <= 2;"

```
In response to your request I have found the following results  >>>>

Table name : Available Results

| PROD_NAME | SUM(PROFITS.AMOUNT_SOLD) | COUNTRY_NAME |
|---|---|---|
| Extension Cable | 32975.10 | United States of America |
| Extension Cable | 5800.55 | Germany |
```

Table 4.3: SQL query syntax used for successful query results

The implemented knowledge base has enabled ANEESAH to formulate complex queries for database information retrieval. ANEESAH's query authoring knowledge extends users abilities to request information from any of 118 database columns in complex combination. ANEESAH has been tested to join 5 different tables to combine information in formulated queries. Table 4.4 illustrates the scope of SQL query formulation capabilities. The SQL engine utilises implemented query syntax to formulate database queries depending on the nature of users' requests.

SQL Query Syntax	Description of SQL Query Syntax
SELECT	- used to select data from a database.
FROM	– used as assignment of source.
WHERE	- used to filter records.
JOIN	- used to combine rows from two or more tables.
AS	– used for surrogate assignment
ON	– used to specify joining value
SUM( )	– used to perform summation function
COUNT()	- used to count values
AVG()	- used to perform average function
GROUP BY	– used to group values based on specific column.
ROWNUM	– used to select specific rows.
ORDER BY	- used to sort the result-set.
GREATEST ( ) DESC	– used to select values with condition.
DISTINCT	- used to return only distinct (different) values
AND	- filter records based on more than one condition

Table 4.4: SQL query syntax implemented in ANEESAH

ANEESAH is able to recognise information relevant to the domain database and converse with users in the case where information is insufficient to execute a valid response. The Sales History database chosen for ANEESAH's evaluation was reviewed from various angles such as schema structure, tables, columns, metadata, and information lying inside database tables, etc. In contrast with sales history database information, the knowledge base was written with various scripts required to formulate queries and yield database results conversationally. The sales history database (schema structure), used for ANEESAH's evaluation, is fully scripted with synonymous information in Domain Database Scripts module. The query syntax (as shown in Table 4.4) reflecting information is also scripted as part of the domain database scripts. A part of the knowledge base (explained in DDK section 4.4.6) is filled with information from the database at system runtimes. The scripting language scripts database information as it arrives in DDK module for simplifying user utterance matching process, and allows PM engine to perform user utterance matching.

4.5 SQL Engine (Component – 3)

The SQL Engine takes a pivotal role in ANEESAH's architecture. The SQL engine is responsible for performing query translation/formulation process from user utterances. The SQL engine works based on implemented techniques together with other subcomponents (SQL Configurator, SQL Execution, and SQL Analyser) in order to retrieve information from the database. The SQL engine relies on database relevant information delivered by the system's controller to analyse syntax requirements to engineer a query structure. The SQL engine is equipped with the expert knowledge to identify the level of complexity involved in a query formulation. The following figure (Figure 4.20) shows a query formulation flow and its execution in the database.

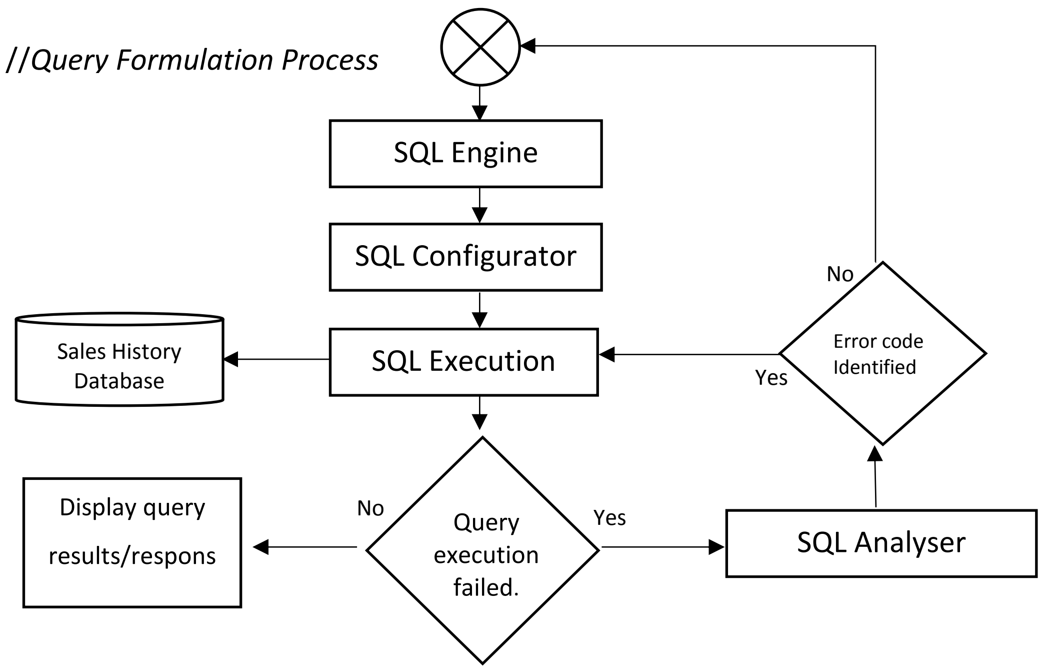


Figure 4.20: SQL query formulation flow

ANEESAH features an automated query structuring approach, which can gather necessary syntax to formulate query structures. The query formulation process adopts a step by step query syntax preparation process. The SQL engine classifies a query based on information received from ANEESAH's controller. The query classification is determined based on query syntax, and complexity. The PM engine also initiates the SQL configurator to formulate a query. The query information is forwarded to the SQL configurator for further processing. The SQL engine works closely with the SQL Configurator seamlessly to generate appropriate syntax to form query structures.

4.5.1SQL Configurator

The SQL configurator is responsible for generating syntactic information required to put together a database query structure. This component works in a query formulation flow, which collects and prepares necessary syntax in a step by step procedure, shown in Figure 4.21.

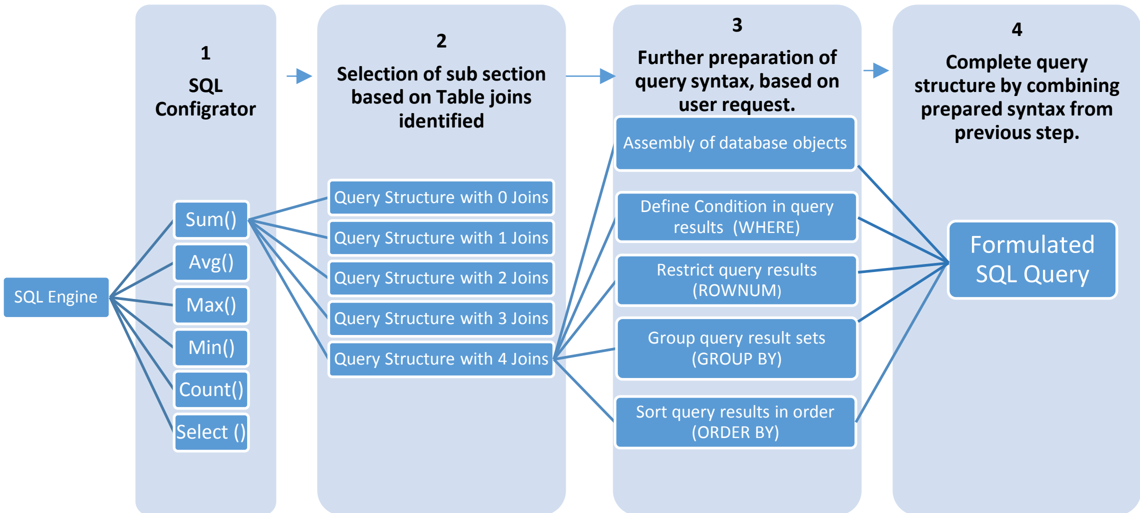


Figure 4.21: SQL Configurator's working flow

For example, a user utterance can require database objects, complex database tables' relation, key information, aggregate functions, or restricted results, etc. Let us consider following example (Example 1) to take an insight into the implemented solution.

SQL Configurator - Example 1:

User utterance: "**can you show me top 5 best selling products in Japan through internet**"

Database objects:

Country_Name, Channel_Desc, Prod_Name, Amount_Sold, Quantity_Sold

Attributes Values Extracted from above utterance:

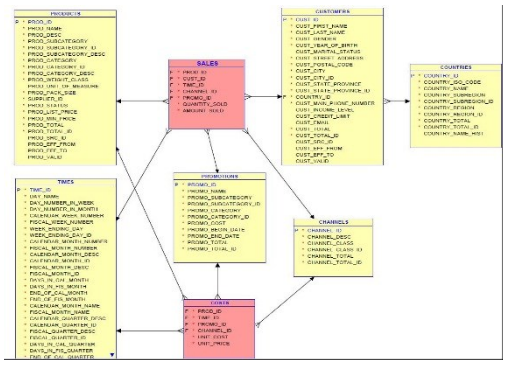
Japan, Internet.

Aggregate functions and selection of results identified with the help of domain knowledge:

Sum() + Group By + Order By + Desc + <= 5

Following a user utterance processing, the information captured (Database Objects, Attribute Value, Aggregate functions, filter, etc.) is received by the SQL configurator, through SQL Engine. Based on information received, the SQL configurator first determines required database tables, primary & foreign keys relations, and table joining conditions. The Configurator does this by using database metadata to match captured objects (Country_Name, Channel_Desc, Prod_Name, Amount_Sold, and Quantity_Sold) against belonging tables in the database. For example (Example 1) utterance, Figure 4.22 shows a selection of relevant tables.

Field Name	Table Name
Country_Name	Countries
Channel_Desc	Channels
Amount_Sold	Profits
Prod_Name	Products
Quantity_Sold	Profits
Additional table selected to complete joining relation.	Customers



Sales History Database Tables/Columns Structure

Figure 4.22: Selection of tables relevant to captured database objects

The relational database's structures are comprised of primary and foreign key based connections, which reflect information relations between database tables or entities. For each query formulation, the selection of relevant tables is gathered based on the shortest table combinations to avoid the addition of unnecessary complexity to the SQL query structure. The SQL configurator can intelligently perform complex table joins depending on primary and foreign key based relations. For example, user utterance (Example 1), Figure 4.23 shows automated table joins formed by selecting relevant database tables based on key (primary & foreign) relations between them.

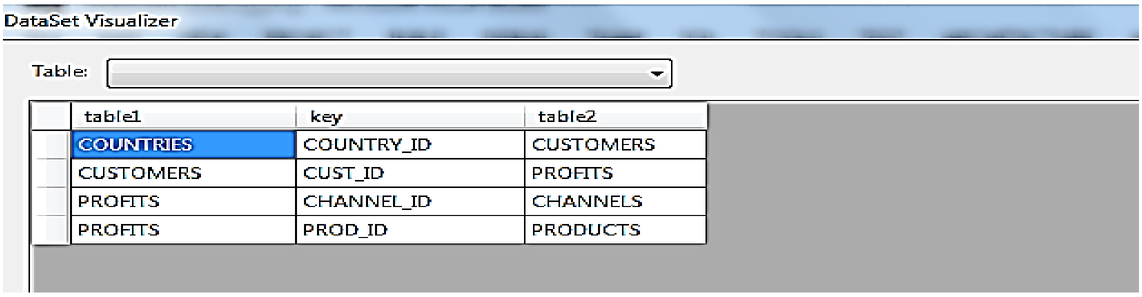


table1	key	table2
COUNTRIES	COUNTRY_ID	CUSTOMERS
CUSTOMERS	CUST_ID	PROFITS
PROFITS	CHANNEL_ID	CHANNELS
PROFITS	PROD_ID	PRODUCTS

Figure 4.23: Selection of tables in relational structure

After the collection of required tables, the SQL configurator performs assembly of captured database objects (Country_Name, Channel_Desc, Prod_Name, Amount_Sold, and Quantity_Sold) in required order to fit SQL standard query structure. Also, database objects are also prefixed with parent database tables to avoid any ambiguity that may be due to the coexistence of objects in different database tables, as shown in Figure 4.24.

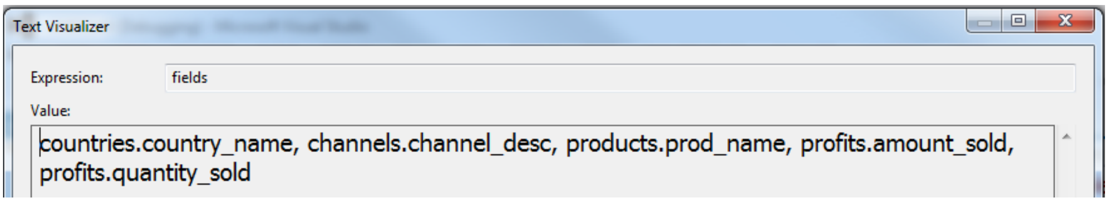


Figure 4.24: Assembly of database objects with source table identification

Following the assembly of database objects, in line with attribute values (Japan, Internet), the condition is applied to the results maintained in selected tables. At this stage, the configurator also identifies the need to specify the number of query records to return, depending on the filter value mentioned in user utterance (i.e. top 5 in case of Example 1). The SQL configurator does this by introducing a “where” clause the in query syntax, which limits database results returned to a predefined condition depending on the user request. The Figure 4.25 shows formulation of conditional syntax to be used as part of the overall query structure.

Figure 4.25: Syntax prepared to restricted database results to user’s desire

For users requests containing aggregation functions, the SQL configurator performs integration of identified aggregation functions (Sum(), Avg(), Max(), Min() etc) with existing query syntax, shown in Figure 4.26.

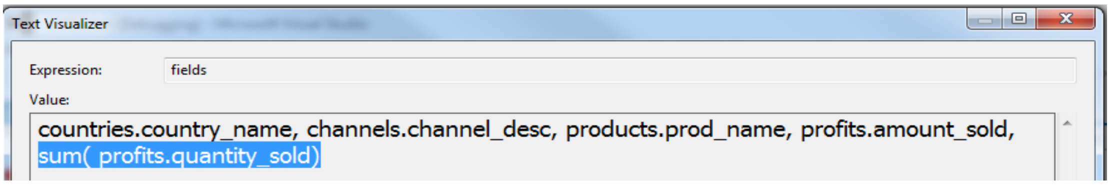


Figure 4.26: Aggregation function integrated into SQL query syntax

Further, SQL configurator identifies the need for grouping result sets or sorting of results in specific order, based on the information transferred by the SQL engine. Following Figure (4.27) shows the introduction of the “group by” syntax to be used in conjunction with the aggregate function to group query result set by columns.

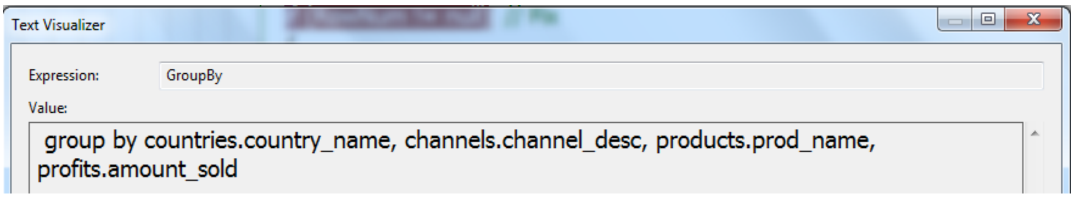


Figure 4.27: Formulation of syntax to group query results

Similarly, for the purpose of Example 1, Figure 4.28 shows the introduction of “order by” keyword, which will sort query records in descending order.

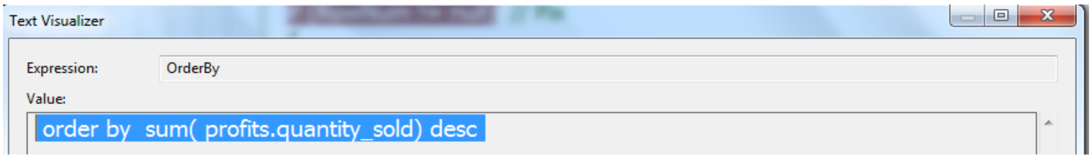


Figure 4.28: Formulation of syntax to sort query results in specific order

Finally, the Configurator completes the query structure by joining segmentally prepared syntax in required order. For the given example (Example 1) user utterance, the SQL configurator formulated following query, shown in Figure 4.29.

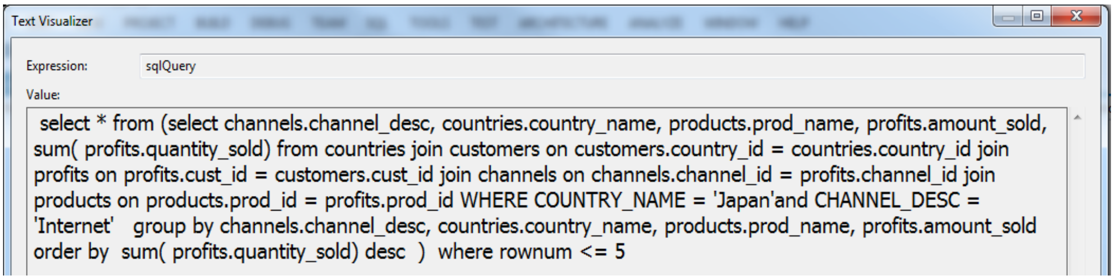


Figure 4.29: Formulated query by the SQL configurator

4.5.2 SQL Execution

The formulated queries are transferred to SQL execution component, which takes the responsibility of executing and retrieving queries relevant results from the database. This component also stores formulated queries in ANEESAH's log file before and after their execution in the database, in order to allow further analysis and improvement. A valid query execution returns database maintained results in the system's temporary memory (memory table), at run times. Followed by each query execution in the database, the SQL execution component validates returned results from system's temporary memory. The SQL execution component is also responsible for calling query returned results from temporary memory, and displaying in the user interface screen, as shown in Figure 4.30.

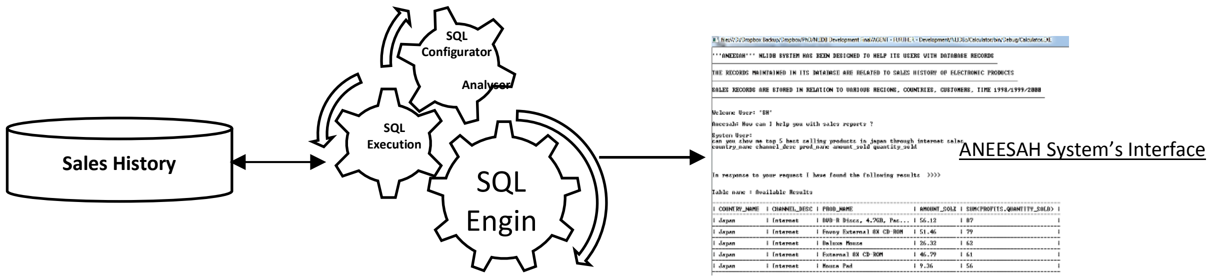


Figure 4.30: SQL Query Execution within ANEESAH

In addition to the query returned results, a scripted text response is also displayed to the user. For example, the user utterance mentioned in the SQL configurator section 4.5.1 ("**can you show me top 5 bestselling products in japan through internet**"), Figure 4.31 shows the representation of results returned by the query, in the user interface.

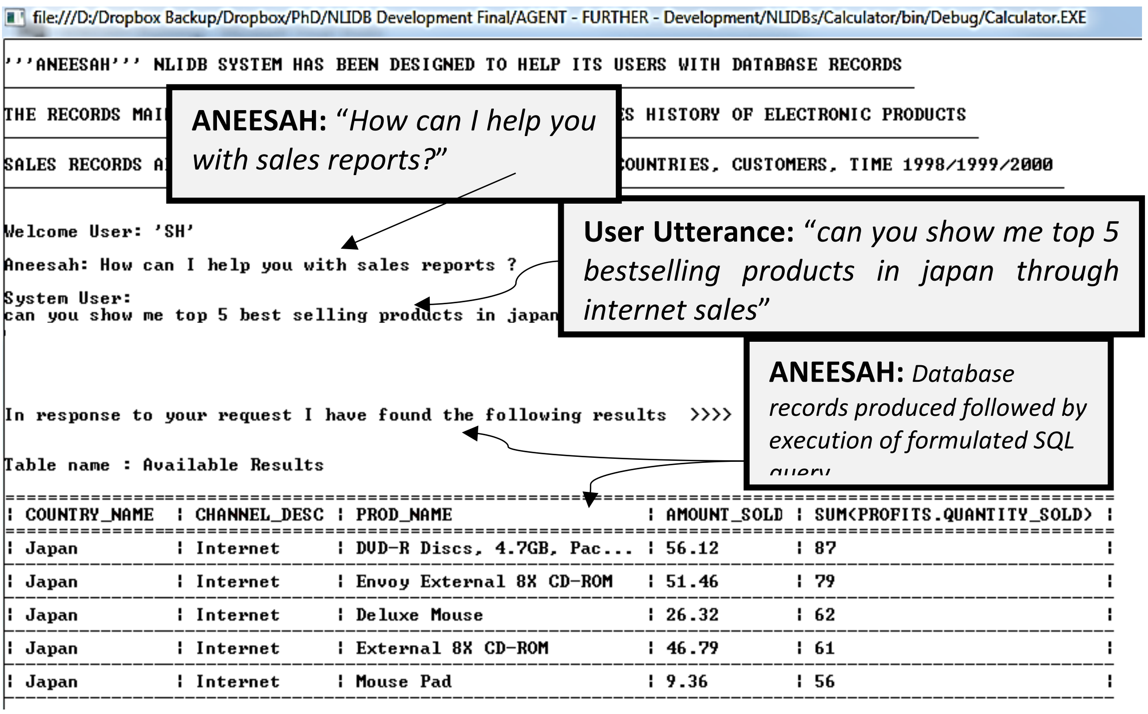


Figure 4.31: Example query returned results display in user interface

4.5.3 SQL Analyser

The SQL analyser component has been implemented as a failsafe tool, which takes the role of a database query structure analyser. The SQL analyser component has been developed to support ANEESAH's abilities to assemble and readjust queries at run time. In a real-world scenario, database queries can fail for a number of reasons i.e. missing syntax, incorrect structure, memory issues, etc. The query failure mostly end up in error(s) referring to the problem in query structure, and addressing such errors (error code i.e. ORA-00936) validates query to generate database information successfully. Query failure errors reflect coded information identifying reasons leading to the query failure, which helps the database developers in resolving problems. The Figure 4.32 explains the detected error code ORA-00937, which refers to incorrect/missing syntax in SQL query executed by the developed system.

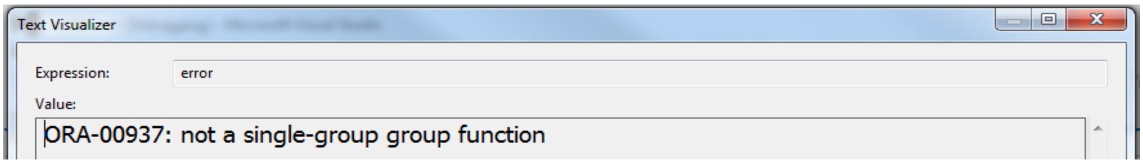


Figure 4.32: Error code showing failed execution of a SQL query

The implemented SQL analyser works by identifying error codes in the event of a query failure. The SQL analyser attempts to readjust query structure if falls within its scope. Failed queries are re-executed after readjustment applied by the SQL analyser in order to extract database information.

4.6 Conclusion

This chapter has detailed the methodology and implemented components which comprise the ANEESAH prototype. Due to the nature of this research and the current state of NLIDB developments, it was not feasible to create a conversational NLIDB using existing development techniques and components. The existing techniques do not contain features necessary to be applied to NLIDB problems such as conversational limitations, conflict resolution, information refinement, etc. The development of the ANEESAH prototype has steered research, design, testing and implementation of several key components (e.g. CA, PM engine, scripting language, SQL engine, etc.) to deal with unique challenges. The most significant contributions of this work are:

- ANEESAH mimics a human query assistant and allows users to access desired information stored in the domain specific database conversationally.

- ANEESAH is able to understand and extract system relevant information by analysing user utterance and map them to perform translation to formulate appropriate responses from the system.
- ANEESAH has the ability to react and perform conflict resolution interactively during conversation with the users.
- A novel CA enabled NLIDB (ANEESAH) architecture has been implemented to process the user's requests interactively. ANEESAH incorporates a novel CA developed using pattern matching and new scripting language.
- The scripting language has been implemented to work in different contexts, where each context is responsible for a conversation topic pursued by the user.
- A knowledge base has been developed based on a domain specific database and other contexts e.g. Frequently Asked Questions (FAQ) and General Chat domain.
- The SQL engine and related components have been implemented to perform dynamic formulation of SQL queries.
- A user interface has been created to deal with the exchange of dialogues and display ANEESAH responses.

Next, the components ANEESAH's architecture will be evaluated for their effectiveness and robustness in order to gather evidence to answer the main research question of 'can user interact with a NLIDB to formulate a query to retrieve and refine desired information from a relational database, successfully?'. The testing/evaluation methodology, experiments and results are detailed in the ensuing chapters.

Chapter 5 - ANEESAH Prototype One – Evaluation Methodology and Results

5.1 Introduction

This chapter describes the empirical studies that were undertaken to validate the proposed ANEESAH architecture highlighted in chapter 4. The evaluation ANEESAH's architecture was performed to analyse the implementation of a CA enabled NLIDB and gauge, whether or not the resulting prototype can converse interactively with the users to automate the query formulation process and allow access to desired information stored in a sample domain database. The preliminary evaluation was conducted to determine functionality, effectiveness and robustness of ANEESAH's architecture and components. Also, the main aim of the evaluation was to answer the fundamental research question "Can a general user interact with a NLIDB to formulate a query to retrieve and refine desired information from a relational database?". The following list gives an overview of points sought for investigation:

- Can the implementation of a CA in NLIDB architecture help and engage users conversationally?
- Can ANEESAH simulate a human query expert in reasoning, logic and information, and lead conversation to formulate database queries?
- Can ANEESAH recognise user requirements and perform dynamic formulation of queries to extract desired database information?
- Can the developed architecture and comprising components address the challenges related to NLIDBs?

There is no benchmark approach for the evaluation of CAs and/or NLIDBs; therefore, a novel evaluation methodology was designed, which aides to evaluate ANNEESAH from subjective and objective aspects. A set of experiments involving end user interaction with ANEESAH were intended to assess the effectiveness and functionality of the developed prototype from the subjective aspects. Also, the ANEESAH system was evaluated from objective perspectives through collection and analysis of user interactions, system responses, formulated queries and results produced from the database to gauge system's robustness, accuracy, task completion and effectiveness.

The following sections outline the research hypotheses, evaluation methodology, designed experiments and the evaluation metrics measured through the testing. The results of the experiments are analysed statistically and presented.

5.1.1 Hypothesis

At this stage of the research, the hypothesis to be tested through proposed evaluation is related to the effectiveness of ANEESAH prototype system. The research hypothesis as follows:

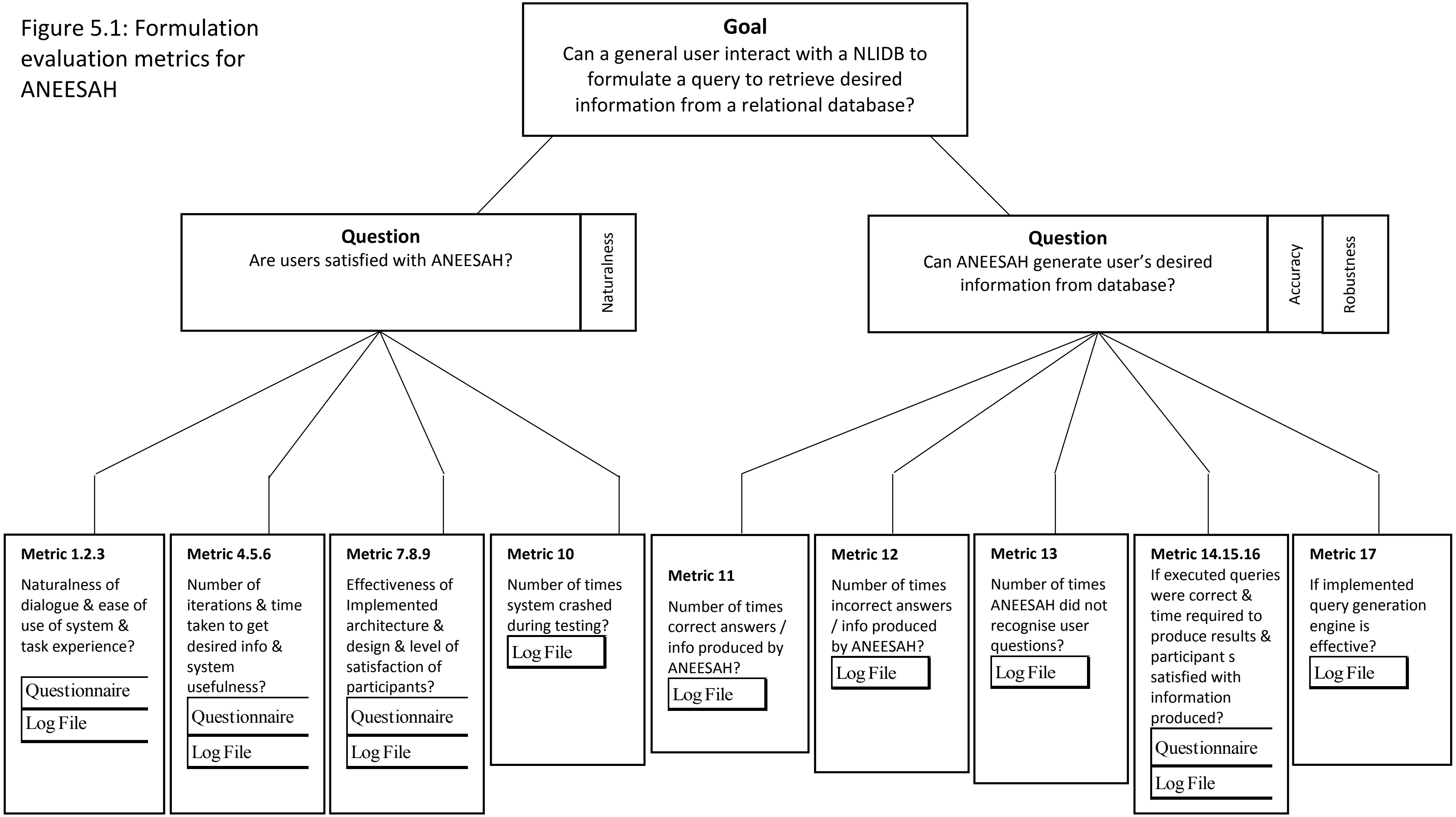
- **H0** – A general user cannot interact with a Natural Language Interface to Database to formulate a query and retrieve desired information stored in a database.
- **H1** – A general user can interact with a Natural Language Interface to Database to formulate a query and retrieve desired information stored in a database.

A Goal, Question, Metric (GQM) model (discussed in section 2.3.5 in chapter 2) was utilised to formulate evaluation metrics to the hypothesis.

5.2 Evaluation Metrics

The proposed evaluation methodology for ANEESAH required the use of a wide spectrum of subjective and objective metrics. As established in the literature review in chapter 2 section 2.2.5 and 2.3.4, there is no standard for the evaluation of CA and NLIDB applications. This created the need for designing a new evaluation methodology for the prototype system. The evaluation was focused on determining the performance of several key components of ANEESAH's architecture such as the implemented CA, knowledge base and SQL query engine (discussed in chapter 4). The formulation of metrics (subjective & objective) included different aspect and features of ANEESAH such as dialogues naturalness, robustness and information accuracy, as shown in Figure 5.1.

Figure 5.1: Formulation evaluation metrics for ANEESAH



5.3 Experimental Methodology for ANEESAH NLIDB

The evaluation of ANEESAH system was achieved through two experiments. The aim of the proposed experimental methodology was to analyse whether or not ANEESAH can provide end users with an interactive environment, understand their requirements and formulate database queries to access desired information. The experiments were designed to gauge the role of different components of the developed architecture such as dialogue responses, effectiveness, robustness and reliability of the database information produced by ANEESAH (as shown in Figure 5.1). For the purpose of designing the evaluation, two methodological questions mentioned in Figure 5.1 were associated with each experiment. The high-level question “Are users satisfied with ANEESAH?” is related to the first experiment. A second question “Can ANEESAH generate user-desired information from the database?” is related to the second experiment.

The first experiment was designed to provide an understanding of the participants’ opinions on system's naturalness, dialogue responses and interaction experience. There were 20 participants in total who to took part in the evaluation of ANEESAH through a scenarios-based evaluation model. The test scenarios were all derived from sample sales history records (discussed in chapter 4 section 4.4.1). All participants reviewed an experiment information sheet and gave consent to take part. The participants were divided into two groups based on their knowledge namely; participants with SQL and database knowledge (referred to as Group A), and other participants who did not possess SQL or database knowledge (referred to as Group B). Each test scenario comprised of a business example, which required the end user to discover specific information from the domain database. The participants were asked to interact with ANEESAH using natural language to find scenario described information.

Following the interaction with ANEESAH, the participants' feedback was gathered in the form a survey questionnaire.

The second experiment was designed to examine various attributes and behavioural factors of ANEESAH prototype system. The information stored in the system's log file (discussed in chapter 4 section 4.3.12) such as dialogues, queries, etc. was utilised for the purpose of the second experiment.

5.4 Evaluation Scenarios

The evaluation of ANEESAH was conducted through devised test scenarios (see Appendix B). The test scenarios were developed in line with knowledge engineering (discussed in chapter 4 section 4.4), a critical review of the existing NLIDBs and mainstream business reporting and database systems such as Oracle, SQL database, SAP used in real life environments. The development scope of ANEESAH and its query formulation abilities were mapped to test scenarios. There were seven scenarios developed in total, each containing an example business situation (i.e. "*As part of product analysis you are required to find company's top five bestselling products in France during the year 1999?* Ask the system to give you this information"), which will require participants to interact with ANEESAH to retrieve scenario described information. All participants (from Group A and Group B) received test scenarios instructions in the form of printed sheets. very participant completed the seven scenarios using a computer on his/her own. The test scenarios were related to query structuring difficulty (i.e. scenario 1 requires a simple/simple query, and scenario 7 requires formulation of a complex structure query) to evaluate the query formulation and execution abilities of ANEESAH.

5.5 Experiment 1

The aim of Experiment 1 was to analyse participants' feedback after their interaction with ANEESAH. The participant filled evaluation questionnaires (Table 5.1) to help determine and analyse their interaction experience and conversational abilities of ANEESAH. The evaluation questionnaire was designed with ten questions structured in a Likert scale format for user experience rating, presented on a five-point scale i.e. (1-5). The evaluation questionnaire also included two questions that the user can respond to with Yes or No, and an open-ended question for participants to write any comments about the prototype system (shown in Table 5.1). The questions were designed based on the evaluation metrics (subjective and objective) selected for ANEESAH's evaluation, as shown in figure 5.1. Some questions in this questionnaire have been used to evaluate other similar systems (Latham et al., 2014).

			1	2	3	4	5	
1	Are you satisfied with interface design & level of dialogue naturalness during conversation?	Very Low	☐	☐	☐	☐	☐	Very High
2	It was easy to understand and use the system.	strongly disagree	☐	☐	☐	☐	☐	strongly agree
3	I can effectively complete my work using this system	strongly disagree	☐	☐	☐	☐	☐	strongly agree
4	I am able to complete my work actively using this system.	strongly disagree	☐	☐	☐	☐	☐	strongly agree
5	I am able to complete my work quickly using this system.	strongly disagree	☐	☐	☐	☐	☐	strongly agree
6	I found this system to be useful	strongly disagree	☐	☐	☐	☐	☐	strongly agree
7	ANEESAH's level of understanding your requirement	Very Low	☐	☐	☐	☐	☐	Very High
8	I feel comfortable using this system	strongly disagree	☐	☐	☐	☐	☐	strongly agree
9	Are you satisfied with ANEESAH's dialogue responses?	Very Low	☐	☐	☐	☐	☐	Very High
10	Are you satisfied with information produced from domain Database?	strongly disagree	☐	☐	☐	☐	☐	strongly agree

11. Would you use these kind of systems in the future?

YES ☐ NO ☐

12. Would you use ANEESAH system instead of taking help from a SQL expert?

YES ☐ NO ☐

Any further comments you may have:

____

Table 5.1: Evaluation Questionnaire

5.5.1 Experiment 1 Results

The questionnaire results from both participant groups (Group A & Group B) show that ANEESAH was well received (as indicated in Table 5.2). For question (1, 2 and 3), Overall 75% of participants from both groups have rated the system interface (frontend) and level of understanding at high, however, 30% of participants rated these features between low and medium. For question (4, 5, 6 and 7), around 70% of participants from both groups perceived ANEESAH to be active, useful and level of understanding, whereas 30% felt less confident in its activeness.

			1	2	3	4	5	
1	Interface and Level of dialogue naturalness during conversation.	Very Low	0%	5%	20%	60%	15%	Very High
2	It was easy to understand and use the system.	strongly disagree	0%	10%	20%	50%	15%	strongly agree
3	I can effectively complete my work using this system	strongly disagree	0%	10%	10%	50%	30%	strongly agree
4	I am able to complete my work actively using this system.	strongly disagree	0%	0%	30%	60%	10%	strongly agree
5	I am able to complete my work quickly using this system.	strongly disagree	0%	5%	15%	60%	20%	strongly agree
6	I found this system to be useful	strongly disagree	0%	0%	25%	70%	5%	strongly agree
7	ANEESAH 's level of understanding your requirement	Very Low	5%	5%	25%	45%	20%	Very High
8	I feel comfortable using this system	strongly disagree	0%	10%	25%	50%	15%	strongly agree
9	Are you satisfied with ANEESAH's dialogue responses?	Very Low	0%	0%	25%	60%	15%	Very High
10	Are you satisfied with information produced from domain Database?	strongly disagree	0%	5%	20%	65%	10%	strongly agree
				Yes	**No**			
11	Would you use these kind of systems in the future?			90%	10%			
				Yes	**No**			
12	Would you use ANEESAH system instead of taking help from a SQL expert?			65%	35%			

Table 5.2: Overall Questionnaire Results from Group-A & Group-B

For question 8, overall system comfort level was rated high by 65% of the participants with 25% rated at moderate, and 10% rated comfort and usability at low. For question 9, the dialogue responses of ANEESAH highly satisfied 70% of participants from both groups. For question 10, most participants were satisfied with information produced by ANEESAH and rated its ability highly (75%), with only 20% giving the medium rating, and 5% participants were less satisfied with its capacity to produced information.

Additionally, the participants from both groups showed very high acceptance level when asked, if they would use a similar system in the future. For question 11, overall 90% of participants agreed on using a similar system with only 10% who showed unwillingness. In response to the question (question 12), 65% of the overall participants from both groups agreed they would take help from ANEESAH instead of a SQL expert. The next section will evaluate ANEESAH's ratings from each group (Group A and Group B).

5.5.2 Experiment 1 Discussion (Group-A)

The participants selected in Group-A had structured query language and database knowledge. Figure 52. and 5.3 show moderate but not significant differences in opinions from both groups. The participants from Group-A (30% of the participants) appear to have for a low opinion of ANEESAH’s user interface, dialogue naturalness and user requirements understanding. 50% of members of Group-A agreed and further 20% strongly agreed with the effectiveness of ANEESAH.

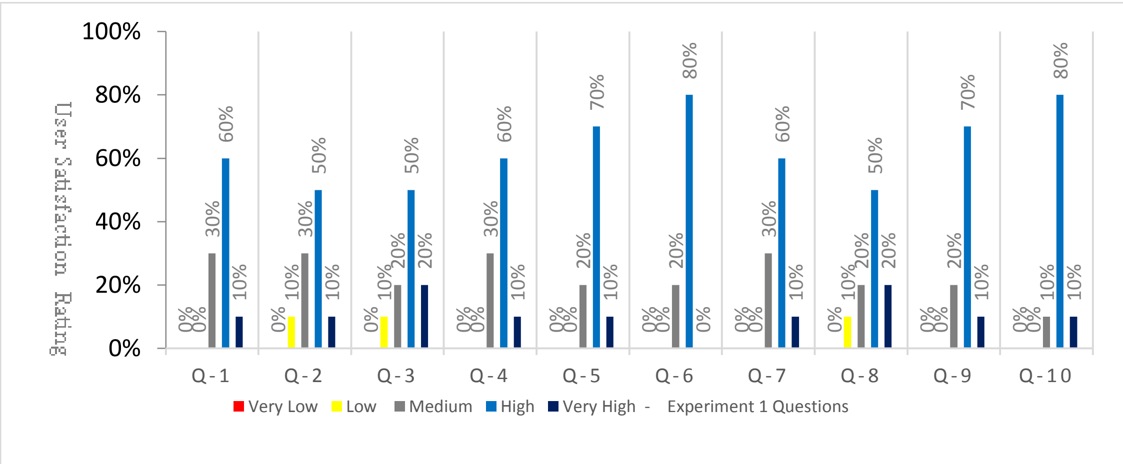


Figure 5.2: Participants Rating from Experimental Group-A

The remaining 30% participants rated ANEESAH’s effectiveness between low and medium. Figure 5.2 also shows that ANEESAH was well-received and majority Group-A participants gave high ratings for evaluation questions. The questions such as response time, usefulness and level of understanding received high ratings from most participants. However, the question regarding the comfort in using ANEESAH received ratings from moderate to low. Overall 80% of participants from Group-A stated their high satisfaction in ANEESAH’s dialogue responses. ANEESAH’s understanding of user requirement received high rating from 60% of the participants but 30% of the overall participants rated low for this metric. Further, ANEESAH received highest satisfaction rating (by 90% of participants) for its ability to produce information from the database. Also, 90% of Group-A participants strongly agreed to use a comparable system in the future. In response to Question 12, 70% of participants agreed to take help from ANEESAH as an alternative to a human SQL developer.

5.5.3 Experiment 1 Discussion (Group-B)

Figure 5.3 illustrates participants rating from Group-B. ANEESAH’s dialogue naturalness and level of understanding was rated high by 70% of the Group-B participants. In response to question 3, 4 and 5, most participants (approx. 70%) agreed that they found ANEESAH effective and efficient in task performing. In particular, 30% of participants from Group-B rated ANEESAH’s activeness at an average level. The same proportion of participants rated ANEESAH’s effectiveness as very high. Figure 5.3 shows that ANEESAH’s factors such as usefulness, the level of understanding (of users’ requirements) and comfort have received the rating between high and very high from 70% of participants. Further, 30% of participants ratings for ANEESAH’s usefulness, understating user requirements and comfort were recorded between low and medium. ANEESAH’s dialogue responses satisfied 70% of participants from Group-B, where 30% of participants who strongly satisfied. Further, ANEESAH’s ability to produce information from the database satisfied approx. 80% of participants (who rated at high).

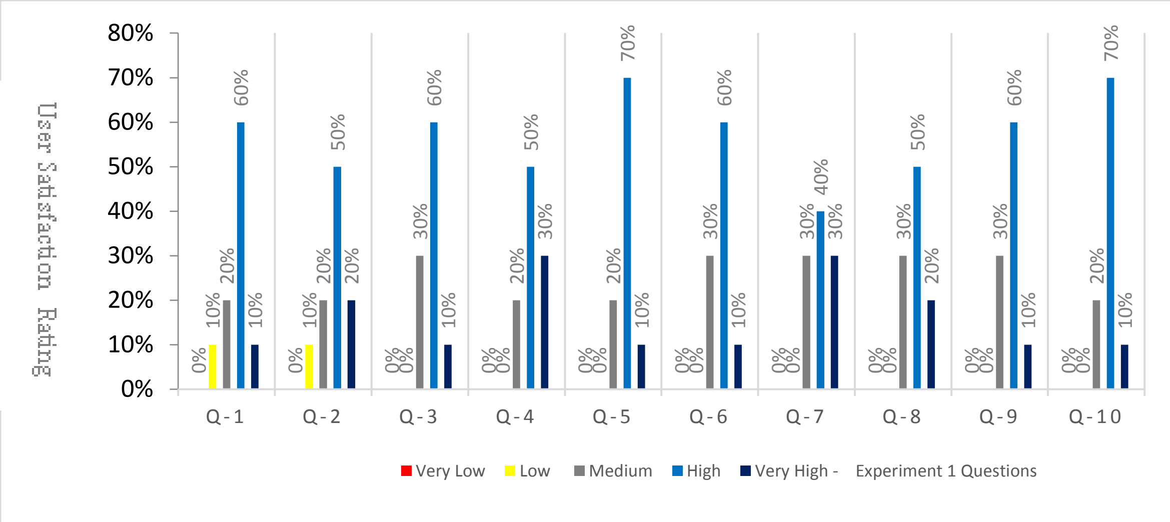


Figure 5.3: Participants Rating from Experimental Group-B

Additionally, when answering question 11, 80% participants from Group-B agreed to use a similar system in the future, and 20% of participants showed an unwillingness to use a similar system in the future. ANEESAH's acceptance as an alternative to a SQL developer agreed by 60% of participants from Group-B with 40% showing no willingness for the same.

5.6 Data Analysis and Selection of Statistical Test

Choosing the right statistical technique for data analysis is the most difficult part for any research (Pallant, 2013). The choice of statistical test is usually related to research questions being sought by the researchers and other factors such as the size of evaluated data, the number of people or groups involved evaluation, measurement scale type and distribution. Statistical tests selected for evaluation are defined in two categories namely; parametric which include an assumption of the population used for deriving the sample data from, and non-parametric type often referred as distribution-free tests, which does not take into consideration the population. Parametric statistics are performed based on the assumption of the population and require numerical values. Non-parametric statistics or inferential statistical analyses are conducted to analyse situations where the data is non abnormally distributed. A non-parametric test can be performed on ordinal and categorical data. However, descriptive statistics are conducted by visual inspection of histograms, which can help in analysing the normal distribution of data during evaluation (Doane and Seward, 2011; Gravetter and Wallnau, 1999).

5.6.1 Inferential Statistics (Mood's Median Test)

Inferential statistics have been used to determine whether the difference between two groups (Group-A & Group-B) is significantly different and not just due to chance. The Mood's median was adopted to compares the medians of Group-A and Group-B participants. The significance test between both groups (Group-A & Group-B) is analysed with the help of Mood's test-driven significance values (Exact Sig.). The null hypothesis will be accepted if significance value is recorded above 0.05. Table 5.3 highlights Mood's median test values recorded for each survey question.

		Mood's median test					
		Number	Median	Exact Sig.	Ratings	Groups	
	Test Scenarios					A	B
1	Interface and Level of dialogue naturalness during conversation.	20	4.0000	1.000	> Median	1	1
					<= Median	9	9
2	It was easy to understand and use the system.	20	4.0000	1.000	> Median	1	2
					<= Median	9	8
3	I can effectively complete my work using this system	20	4.0000	1.000	> Median	2	1
					<= Median	8	9
4	I am able to complete my work actively using this system.	20	4.0000	0.582	> Median	1	3
					<= Median	9	7
5	I am able to complete my work quickly using this system.	20	4.0000	1.000	> Median	1	1
					<= Median	9	9
6	I found this system to be useful	20	4.0000	1.000	> Median	0	1
					<= Median	10	9
7	ANEESAH's level of understanding your requirement	20	4.0000	0.582	> Median	1	3
					<= Median	9	7
8	I feel comfortable using this system	20	4.0000	1.000	> Median	2	2
					<= Median	8	8
9	Are you satisfied with ANEESAH's dialogue responses?	20	4.0000	1.000	> Median	1	1
					<= Median	9	9
10	Are you satisfied with information produced from domain Database?	20	4.0000	1.000	> Median	1	1
					<= Median	9	9

Table 5.3: Mood's median test results

The significance value recorded for **Question 1** (*Interface and Level of dialogue naturalness during conversation*) is greater than 0.05 threshold, therefore shows no significant difference between Group-A and Group-B. The null hypothesis can be accepted as the ratings from both groups do not differ significantly.

The Mood's median test reveals no significant difference for survey **Question 2** (*It was easy to understand and use the system*), as the significance value recorded for both groups is higher than 0.05. Therefore, it is not a considerable difference between the two groups, and null hypothesis can be accepted.

The significance value recorded for **Question 3** (*I can effectively complete my work using this system*) does not represent a significance difference between both participants groups. Therefore, the null hypothesis is accepted for question 3.

For **Question 4** (*I am able to complete my work actively using this system*), significance value is recorded higher than 0.05. Therefore, there is no considerable difference between the both groups' rating.

Further, Mood's median test revealed no significant difference for **Question 5** (*I am able to complete my work quickly using this system*). The significance value is measured greater than 0.05; hence it is maintained that rating given by both groups reflected no significant differences. Therefore, null hypothesis will be accepted on this occasion.

The significance value recorded for **Question 6** (*I found this system to be useful*) does not reflect a significant difference between Group-A and Group-B members rating. Therefore, the null hypothesis will be accepted for **Question 6**.

Both groups ratings for the **Question 7** represent slight but not significant difference (*ANEESAH's level of understanding your requirements*), as the value (Exact Sig.) was measured higher than 0.05.

The significance value for **Question 8** (*I feel comfortable using this system*) showed no significant difference. Therefore, distribution of ratings from both groups does not differ significantly, and the null hypothesis is accepted on this occasion.

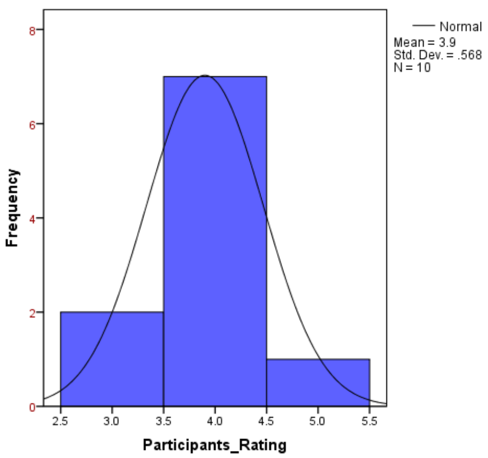
The significance value for **Question 9** (*Are you satisfied with ANEESAH's dialogue responses?*) shows that participants rating from both groups reflects no significant difference and significance value is recorded higher than the threshold value (0.05). The null hypotheses will is assumed for **Question 9**.

The rating given by both participant groups for **Question 10** (*Are you satisfied with information produced from domain Database?*) were analysed to measure significance difference, which showed that the significance value recorded was higher than 0.05.

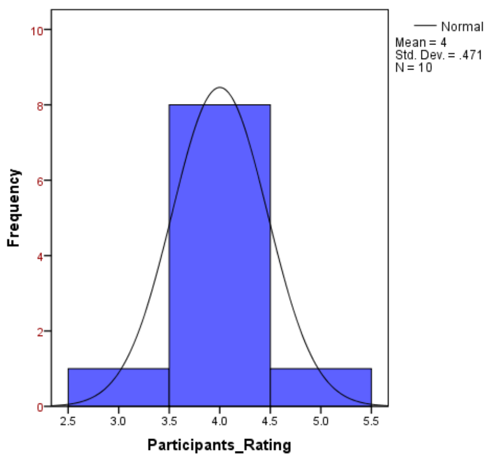
The above test has established data normality concerning data distribution. Further, in the following sections, descriptive statistical analysis techniques have been utilised to investigate the difference of data normality between participant groups.

5.6.2 Descriptive Statistics (Test of Normality)

The descriptive statistics test has been used due to small sample size. The histograms and paired means test are used to calculate and evaluate normality of data distribution over both participant groups (Group-A & Group-B). The visual inspection of the histogram (see Appendix C) for both groups (Group-A & Group-B) reveals the approximate shape of the normal curve. Therefore, it is assumed that data was normally distributed. Figure 5.4 is an example normality histogram values (for one question) for both groups.



Are you satisfied with information produced from domain Database? (Group-A)



Are you satisfied with information produced from domain Database? (Group-B)

Figure 5.4: Normality Histograms

The above test has established data normality in distribution. The descriptive and inferential test results highlight the normality between tested metrics in contrast with the perception of participants from both groups. The next section (5.7) will discussed experiment 2.

5.7 Experiment 2

The aim of Experiment 2 was designed to perform analysis such as robustness and accuracy (evaluation metrics 11 to 17 from Figure 5.1). This used information captured from ANEESAH’s log file with records such as occurrences of dialogues between participants and the prototype system during evaluation. The following sections will provide detail on objective aspects of ANEESAH's evaluation such as interactive sessions, robustness and information accuracy.

5.7.1 Interactive Sessions

In this evaluation, ANEESAH handled 485 dialogues from twenty participants, an average 24.5 utterances per participant. Table 5.4 illustrates the distribution of log file recorded session dialogues from the seven scenarios, allocated to each participant during Experiment 2. The ANEESAH’s log file was configured to record many variables such as dialogues/utterances between participants and prototype system, rejected statements, attribute and characteristics, key figures, context, SQL queries, etc. The number of utterances shown in Table 5.4 represents ANEESAH's ability to interact with end users.

Table 5.4 Headings:

Utterances – Exchange of dialogues between users and ANEESAH
Correct Results – ANEESAH produced query results
Inadequate Results – ANEESAH query results contained excessive records
Incorrect Results/Failure – Incorrect query results or system failed/crashed.

Scenario Number	Utterances	Correct Results	Inadequate Results	Incorrect Results – System Failure	Utterances	Correct Results	Inadequate Results	Incorrect Results – System Failure	Total Dialogues
	Group A				Group B				
Scenario 1	38	9	0	1	30	9	0	0	68
Scenario 2	31	10	0	0	27	10	0	0	58
Scenario 3	43	5	1	4	44	5	0	1	87
Scenario 4	28	9	0	1	27	9	0	0	55
Scenario 5	45	10	0	0	30	10	1	0	75
Scenario 6	28	9	1	0	32	9	1	1	60
Scenario 7	52	4	4	2	30	4	1	2	82
Total	**265**	**56**	**6**	**8**	**220**	**56**	**3**	**4**	**485**

Table 5.4: Number of utterances and results for each test scenario

Table 5.4 also illustrates statistical information captured from the system log file. The number of correct results represents ANEESAH's ability to understand user requirement followed by dynamic SQL query formulation and database information retrieval. Further, utterance distribution for Group-A and Group-B followed by correct and incorrect results produced by ANEESAH for both groups. Table 5.4 also reflect inadequate system responses and its failure to understand or react to the participants' requirements during evaluation.

The information mentioned in time column of Table 5.5 shows overall tasks completion time for each scenario. The participants' interaction time with ANEESAH also reflects the level of difficulty embedded in each scenario. Scenario 3, 5, 6 and 7 relatively took more time due to embedded query structuring complexity when comparing with scenario 1, 2, and 4.

Table 5.5 Headings:

Utterances – Exchange of dialogues between users and ANEESAH
Correct Results – ANEESAH produced query results
Inadequate Results – ANEESAH query results contained excessive records
Incorrect Results/Failure – Incorrect query results or system failed/crashed.

Scenario Number	Utterances	Correct Results	Inadequate Results	Incorrect Results - System Failed	Time Per Scenario
Scenario 1	81	95%	5%	0%	11.71%
Scenario 2	73	100%	0%	0%	10.94%
Scenario 3	103	70%	25%	5%	15.18%
Scenario 4	70	95%	5%	0%	12.39%
Scenario 5	94	95%	0%	5%	14.01%
Scenario 6	80	85%	5%	10%	16.44%
Scenario 7	115	55%	20%	25%	19.28%

Table 5.5: Experiment results for Group-A & Group-B

The test scenarios for ANEESAH's evaluation were devised with embedded difficulty and complexity. Figure 5.5 gives an understanding of the level of difficulty (designed in task/scenario completion) for each scenario based on success results for each scenario. Figure 5.5 also shows that Scenario 1, 2, and 5 were easily handled. Scenario 2 was completed successfully by all participants from both groups. Scenario 3, 6 and 7 proved difficult for ANEESAH. Scenario 1 and 4 had been completed successfully by all participants except for one participant where ANEESAH produced results with excessive information. Scenario 6 was better achieved by participants when comparing with Scenario 3 results. While completing scenario (Scenario 3), 25% of participants were presented with excessive information by the prototype system. Scenario 7 is visible as most difficult (with 25% of overall participants failed to complete the task (either query results were incorrect, query failed, or system crashed) and further 20% participants received query results containing excessive information (information in excess of what was originally requested by the participants)

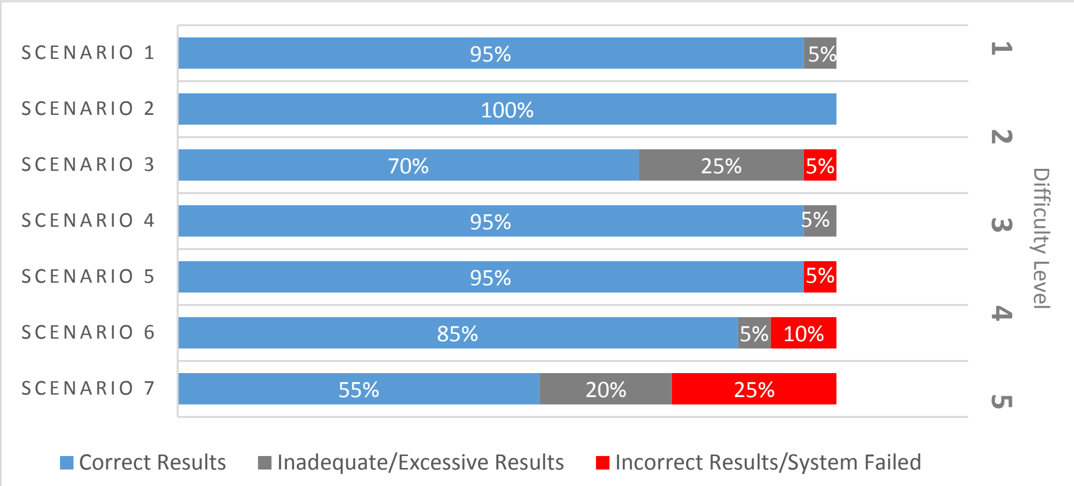


Figure 5.5: Level of difficulty for each scenario

5.7.2 ANEESAH's Dialogue Responses

This section will discuss ANEESAH's responses and user interaction abilities noted during the evaluation phase. This section also comprises few example dialogues collected from the prototype systems' log file. Figure 5.6 highlights one participant's attempt to complete Scenario 7. ANEESAH effectively guided and managed the conversation by engaging participants in disambiguating between duplicate records. The discussion shows ANEESAH stayed on topic and focused on the user desired information "country_region" following the query formulation path to extract database information.

Scenario – 7
ANEESAH: *How can I assist you with sales information stored in company database?* **Participant-1:** "how many mouse pads were sold in asia?" **ANEESAH:** *Search have found duplicate records for ' asia '* *1 country_subregion asia* *2 country_region asia* *Please make appropriate selection, you can write below:* **Participant-1:** "delete 1 please" **ANEESAH:** *In response to your request I have following records:*

```
In response to your request I have found the following results  >>>>
=============================================================
¦ COUNTRY_REGION               ¦ COUNT(PRODUCTS.PROD_NAME)  ¦
=============================================================
¦ Asia                         ¦ 2323                       ¦
-------------------------------------------------------------
You can entre your request to find records related to Company Database.
```

Figure 5.6: Example dialogues between ANEESAH and user

Figure 5.7 highlights that ANEESAH did not only perform validation of participants' inputs but also led conversation during experiments. Figure also 5.7 reveals that ANEESAH promptly recognised and reacted to participant's use of the abusive word in his/her input. ANEESAH responded with a warning message and urged the participant to enter a valid input. Subsequently, the participant entered some random characters "slk;dfalsd", which also attracted an appropriate system response "*Sorry I didn't understand what you said. Please try again*". Further, as soon as participant entered (rather less structured) but meaningful/knowledge base relevant input, ANEESAH promptly recognised it and offered database available information.

Scenario - 2

ANEESAH: *You can enter your request to find records related to Company Database.*
Participant-6: "stupid system"
ANEESAH: *You have used a bad word. Please enter a valid requirement.*
Participant-6: "slk;dfalsd"
ANEESAH: *Sorry I didn't understand what you said. Please try again.*
Participant-6: "barcelona's customer first and last name, email, contact number"
ANEESAH: *Do you want the system to display found records related to "Barcelona customer email, customer first name, customer last name, customer number"?*
Participant-6:"yes"
ANEESAH : *In response to your request I have following records:*

CUST_CITY	CUST_EMAIL	CUST_MAIN_PHONE_NUMBER	CUST_FIRST_NAME	CUST_LAST_NAME	CUST_ID
Barcelona	RossellettCcompany.com	110 204 8560	Abner	Rossellett	16562
Barcelona	Roland@company.com	571-357-4517	Abraham	Roland	16563

Figure 5.7: Example dialogues between ANEESAH and user

Figure 5.8 shows (Scenario 6) ANEESAH's ability to understand and converse with end users to confirm exclusion of irrelevant information from their requests. ANEESAH offered available results and truncated nonexistence information from the participant's input. The prototype system also offered the participant the chance to restrict query produced results. The participant's desire to view top five query results understood and displayed by the prototype system.

Scenario - 6

ANEESAH: *How can I help you with sales information?*
Participant-20: "Can I have a look please on top fve bestselling products during 1999 in France and Pakistan?"
ANEESAH: *The system have found records related to "highest sold products, France, 1999," But could not find any records relevant to "Pakistan"? Would you like to ignore "Pakistan" and see available records?*
Participant-20: "Yes**"**
ANEESAH: *The number of records returned against your request are* ***781****. You can limit number of rows to appear on screen by Entering "a number" below:*
Participant-20: "5"
ANEESAH: *Please make a valid selection you can enter selection i.e. first 5, last 5 etc. Please try again.*
Participant-20: "FIRST 5"
ANEESAH : *Following is the information in response to your request:*

CALENDAR_YEAR	COUNTRY_NAME	PROD_NAME	AMOUNT_SOLD	SUM(PROFITS.QUANTITY_SOLD)
1999	France	Keyboard Wrist Rest	13.66	103
1999	France	Mouse Pad	11.38	94
1999	France	1.44MB External 3.5" Di...	10.1	94
1999	France	Mouse Pad	11.22	91
1999	France	Mouse Pad	10.15	89

Figure 5.8: Example dialogues between ANEESAH and user

Further, when attempted to complete Scenario 5 (as shown in Figure 5.9), ANEESAH partially understood participant's request. The participant's request did not contain intention and the word "countres" was incorrectly spelt. Therefore, in the absence of a valid purpose, it offered to show partially understood/available results. The participant realised his/her mistake and asked to include "countries" in query results. Subsequently, ANEESAH then validated participant's requirement followed by production of query based response.

Scenario - 5

ANEESAH: *What would you like to know about the company's sales today?*
Participant-20: "total profit from different countres"
ANEESAH: *Would you like see records maintained for "', total profit (i.e. net company gain from business) ' "?*
Participant-20: "add countries in it"
ANEESAH: *Are you looking for information related to 'countries, total profit (i.e. net company gain from business) '?*
Participant-20: "yeah"
ANEESAH: *Below is the found information against your request:*

COUNTRY_TOTAL	COUNTRY_NAME	PROFIT
World total	Spain	432444.59
World total	United Kingdom	1322567.68
World total	Canada	552488.97

Figure 5.9: Example dialogues between ANEESAH and user

5.7.3 Precision, Recall and Accuracy

This section will describe ANEESAH's efficiency and robustness, noted in this evaluation. The recall and accuracy measures (discussed in chapter 2 section 2.2.5) have been adopted to determine these factors. The information captured in the log file was analysed to determine the information accuracy.

The recall value for Group-A participants, having familiarity with structured query language and databases, was recorded at 90% and the information accuracy is noted as 80%. The recall value for Group-B participants with no knowledge of structured query language and database recorded at 95% and ANEESAH’s overall accuracy for database information produced during experimentation is calculated at 90%.

As mentioned in section 2.2.5 of chapter 2, unfortunately, there is no uniformity on what constitutes as a correct results query. Historically, in some NLIDB evaluations, queries with excessive results are often counted as “correct queries”, while in other NLIDB evaluations queries with excessive information are not considered as correct (Pazos R. et al., 2013). For ANEESAH's evaluation, queries with excessive information (i.e. results with additional database fields, excessive records etc.) are not considered as correct. The overall accuracy of ANEESAH for query produced results (for experimental Group-A and Group-B) is recorded as 85%. Further, the harmonic mean or F-measure (discussed in section 2.2.5 chapter 2) is used to measure test of accuracy by combining accuracy and precision. This measure provides approximately the average of the two (accuracy and precision) and has been recorded as 87.42%.

5.8 Discussion

An ideal or more robust NILDB system should be conversationally strong and be able to guide users to ensure goal achievement. The experiments conducted on the prototype system and collected results suggest and validate the robustness and accuracy of ANEESAH's framework as a conversational NLIDB. The evaluation results have shown that ANEESAH can engage users in conversation, provide conflict resolution and perform dynamic query formulation to extract database information. Therefore, the null hypothesis (**H0**) is rejected, and the alternative hypothesis (**H1**) is assumed that a general user can interact with a Natural Language Interface to Database to formulate a query to retrieve desired information from a relational database. The feedback and lessons learnt from these experiments will be dealt in the future work.

The end user evaluation brought to light weakness in ANEESAH's architecture, mainly the number of unrecognised utterances and incorrect query responses. ANEESAH failed to recognise some utterances from the participants (7% approx.). Upon further analysis of the log file it was found that some of these unrecognised utterances were due to minor spelling mistakes in the user/participants utterances, gaps in the knowledge base. The other weaknesses were identified as failure of SQL engine, responses perceived as machine-like and command line interface.

The spelling mistakes from the users inputs led to misunderstanding and repetition of responses from ANEESAH. ANEESAH is programmed to ask the user to repeat or rephrase his/her input "*Sorry I didn't understand what you said. Please try again*". However, if the spelling mistake is not corrected, the prototype system will again fail to understand that would eventually lead to the termination of chat session. Figure 5.7 and 5.8 show that although ANEESAH failed to recognise spelling mistakes, it showed the tenancy to support and recover the conversation towards users desired goals. The text in the system's log file (reflected in Table 5.4 and 5.5) reveals that where ANEESAH failed to recognise spelling mistakes, it offered available response for the matched portion of the user utterance. However, this issue will require further research to develop new approach, which will reduce the impact of these language unique issues on the CA's performance.

As mentioned above in this section, one of the other causes for the unrecognised utterances was due some gaps revealed in the knowledge base during end user evaluation, but these gaps are easily addressable, simply by further scripting to the knowledge base. The survey questionnaire revealed that system's interface was very basic, and its responses were perceived machine-like. This will require further investigation and development to address these points.

5.9 Chapter Summary

The initial evaluation showed some key information with respective to the effectiveness, robustness and functionality of ANEESAH. The main findings of the evaluation are summarised below:

- ANEESAH can mimic as a human query assistant, which can lead/guide conversation with the users to achieve their desired goals
- ANEESAH is able to understand, analyse and perform translation of user utterance to formulate the appropriate response.
- It incorporates a novel CA developed using pattern matching, scripting language and implemented string similarity algorithm.
- ANEESAH employs a novel SQL engine to perform dynamic query formulation to retrieve database stored information.

The initial evaluation and testing of the ANEESAH prototype system revealed weaknesses in specific components of its architecture. Further research will be required to achieve all research objectives and to address the points highlighted through the initial end user evaluation. The following section details components for further development and enhancements.

1. Development of existing ANEESAH prototype, to further its abilities to handle more complex user requirements and address linguistic problems noted during preliminary evaluation.
2. Implement a graphic interface by replacing existing command line interface to add to improve user system experience, as noted during initial evaluation.
3. Further development to increase ANEESAH's conversational abilities to sustain dialogues and engage users in smart conversation to refine information
4. Research and development of implemented SQL engine to perform query refinement/querying the query scenarios, which would allow users to carry out drill down and drill across information analysis
5. Research, develop and enhance the implemented pattern matching engine, improve the knowledge base and other components of the prototype system highlighted to address points found during end user evaluation.

These weakness and further refinements and enhancements will be addressed by further research and development which is detailed in the next chapter.

Chapter 6 - ANEESAH NLIDB (PROTOTYPE TWO) WITH INFORMATION REFINEMENT

6.1 Introduction

Chapter 6 of presents the results of an evaluation of the first prototype of ANEESAH NLIDB. The initial evaluation (chapter 5) showed several weaknesses in ANEESAH's architecture such as failure recognised user utterances, incorrect query responses. Some weaknesses revealed during initial evaluation were due to morphological nature and limitation of conversational features such as the inability to handle spelling mistakes, which affected the overall effectiveness and robustness of ANEESAH. Additionally, survey questionnaire results highlighted that end users perceived ANEESAH's responses as machine-like and command line interface as very basic/unfriendly. Following is the list of major issues and weaknesses noted during end user evaluation:

- The initial evaluation revealed ANEESAH failed to understand users' requirements/inputs.
- ANEESAH's inability to recognise spelling mistakes led to incorrect/inadequate query responses.
- ANEESAH's query formulation and execution abilities showed weaknesses, which led to query/system failures.
- ANEESAH's command line interface was criticised and described as very basic/generic and unfriendly.
- The initial evaluation also revealed that participants perception about ANEESAH's dialogue naturalness was low (e.g. responses were machine-like).

These issues were individually researched and investigated that led to the enhancement of existing components and development of new components for the existing architecture. Additionally, the further development was also aimed to answer all research questions (that have not been answered during the initial evaluation e.g. Can a NLIDB allow users to engage in sustained dialogues to refine query produced information from a database?). In addition to addressing these weaknesses the following novel enhancements made to the existing architecture of ANEESAH.

- ANEESAH's conversational abilities and domain specific knowledge were analysed, and the knowledge base was expanded with more domain specific information to improve its abilities to understand user requirements (discussed in Section 6.5).
- A spelling correction feature has been introduced with the help of a language dictionary to address the negative impact spelling errors had on the overall performance of ANEESAH (discussed in Section 6.3.3).
- The query formulation components were enhanced and further strengthened to address the issues highlight during the initial evaluation. In addition, novel information refinement abilities have implemented to allow users to sustain dialogues and refine query produced information with the ability to perform querying the query operations (discussed in Section 6.4).
- A graphical user interface was developed to improve end user interaction and to address system engagement related issues such as usability, user understanding of ANEESAH's responses (discussed in Section 6.6).
- The knowledge base was further developed with scripts to aid end user conversation to be perceived more natural, casual and friendlier (discussed in Section 6.5).

This chapter will highlight further investigation of existing components of the ANEESAH's architecture as well as the development and addition of new features. The next section will show the updated architecture of ANEESAH followed by section (6.2) explaining what research and development decisions were made to overcome these issues.

6.2 Revised Architecture of ANEESAH NLIDB

Figure 6.1 shows the updated architecture of ANEESAH. Figure 4 also highlights new components of ANEESAH's architecture and their integration/interaction with other components to address issues noted through end user evaluation. The new and modified features have been explained the following sections of this chapter.

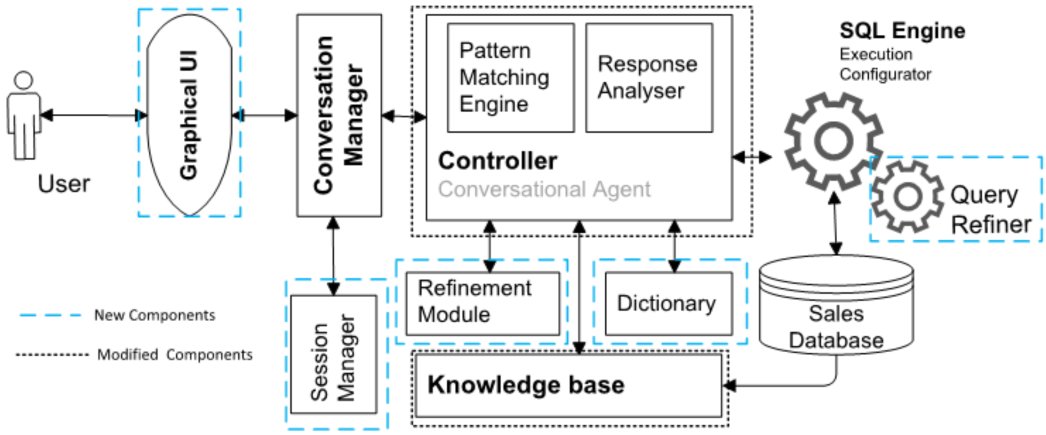


Figure 6.1: Improved Architecture of ANEESAH NLIDB

6.3 Further Development of ANEESAH NLIDB (Overview)

Further development work was carried out by following the originally proposed development methodology (discussed in chapter 3) to address all issues noted during end user evaluation as well as extend ANEESAH' abilities with additional features such as sustain dialogues to refine query produced database information. The further development combines improvements to the existing architecture and addition of new features to improve ANEESAH's ability to mimic as a human structured query language expert. The research and development decisions made to address noted issues, weaknesses and addition of new features as follows:

- **Extended Features of ANEESAH's CA**

The pattern matching (PM) engine was improved to recognise and deal with challenges and common mistakes in user inputs such as spelling mistakes, the ambiguity of information. This was achieved by improvement and addition of new features such as improvement in the algorithm (section 6.4.3), the introduction of a language dictionary to handle spellings and unrecognised user inputs related issues.

- **Information Refinement Features**

ANEESAH's architecture has been further developed to support sustained dialogues (section 6.4) with end users to perform information refinement operations and querying the query operations (e.g. user's ability to add/remove/update more information/records in existing query produced information/results etc.). This was achieved by modification and construction of features such as pattern matching engine components, development of query refining algorithm and SQL refiner module, etc.

- **Knowledge base expansion**

Through further knowledge engineering, ANEESAH's knowledge base was extended to provide improved user experience and make the conversation more natural. The knowledge base was extended with domain specific and other conversation topics (e.g. Frequently Asked Questions (FAQ) and General Chat (GC)) to strengthen ANEESAH's understanding of user requirements.

- **New User Interface**

The old command based user interface was replaced with a new Graphical User Interface (GUI) to improve the end user experience, make information more presentable. The new interface has been implemented to also provide ease in clarification and disambiguation to make interaction experience more natural and improve user satisfaction.

- **Database base management tool**

At run time, a part of ANEESAH's knowledge base is filled (discussed in chapter 4 section 4.4.6) with selective/current domain database information such as schema, master data records. The management tool has been added to control and select domain database information for including in the knowledge base, which can be used to converse with users for the formulation of queries.

These components have contributed to ANEESAH's overall improvement, effectiveness and information accuracy. Each component has been discussed in the following section.

6.4 Extended Features of ANEESAH's CA

The first evaluation results revealed positive findings for the ANEESAH's CA with respect to its ability to recognise and process user inputs. However, there were few points highlighted through the end user evaluation that required enhancement and further development to improve ANEESAH's robustness and effectiveness. The evaluation also revealed that ANEESAH's responses were perceived more machine-like, failed to handle common spelling mistakes made by users, failed to recognise user requests due to lack of scripts in the knowledge base. In addition to the improvements highlighted through end user evaluation, additional development was also required to achieve sustained dialogues and query refinement features. Therefore, the ANEESAH's CA was improved and further developed with new components to address issues raised through end user evaluation. The pattern matching (PM) engine has been developed to improve utterance matching process. The PM engine's ability to detect and handle user intention to switch topic or context has been improved. The response analyser component of CA is further developed to improve user input analyses. The next sections (6.4.1, 6.4.2, 6.4.3) will illustrate the new components added to the CA including date matching and user response agreement.

6.4.1 Date/Time Matching Feature

In real-life environments, time has a significant importance in information analyses. The segmentation of information over different time spans (such as weekly, monthly, quarterly, yearly) is very common. The database records are always maintained with date/time stamps, which can be used to perform historical analyses. The date matching feature of ANEESAH has been further developed to allow users to ask time driven database information with slice and dice approach to divide information into different parts or time. The date and time matching feature enables users to ask time driven information by either using an actual database table maintained time records (such as January, 1999) or by using informal expressions such as last week, last quarter. The ANEESAH' knowledge base was extended to allow user inputs to match against different time related scripts.

For example, consider the following user utterances:
Example user input 1: “I want to see last quarter sales from Spain”
Example user input 2: “I want to see sales for last quarter of 1998 from Spain”

Example user input 1: I want to see last quarter sales from Spain **Time Pattern-1:** ^(?=.*\b((?:last?))\b)(?=.*\b((quarter?))\b).*$ **Pattern Category:** Time **Rule Id:** 3.1 **Response:** System Date and (System Date - 90 days) **Example user input 2:** I want to see sales for last quarter of 1998 from Spain **Time Pattern-2:** ^(?=.*\b((?:last?))\b)(?=.*\b((quarter?))\b).*$ **Time Pattern-3:** ^(?=.*\b((?:year?))\b).*$ **Pattern Category:** Time **Rule Id:** 8.2 **Response:** 1.10.1998 and 31.12.1998

Table 6.1 – Example of user input match against time pattern

The example user input 1 contains “last quarter” words corresponding to a single Time Pattern-1 (scripted with regular expression discussed in chapter 4 section 4.3.5.1), which matched in the knowledge base against appropriate response (as shown in Table 6.1). The response query will retrieve database results with date restriction from system date and 90 days before. The time matching feature can detect and react with an appropriate response when users’ requests contain variation or selection of different date or specific period (as shown in example user input 2). In the example user input 2, the user input contains “last quarter” and “1998” words corresponding to two matched patterns (e.g. Time Pattern-2 and Time Pattern-3). In this case, using the developed time logic, the response query will retrieve database results for the three months of 1998.This feature has enabled ANEESAH to produce time driven information from the domain database.

6.4.2 User Response Agreement

The ANEESAH prototype system employs a new feature to check the user agreement on query produced responses. The user agreed responses are stored in the system’s log file. This feature was added to help ANEESAH to engage in conversation with confidence. Further to a query produced response, ANEESAH asks the user “*ANEESAH: I have discovered information reflected in 'ResultsView' window relevant to your input. Is this* *what you were looking to find out from the database?*". The user agreement leads to a link response "*ANEESAH: Thanks, let's continue with sales discussion*". Following up from the user agreement/confirmation, ANEESAH initiates the query refinement feature, which allows users to refine the existing query produced information/results (discussed in 6.5), or enter a new information request. If the user disagrees with a produced response, then ANEESAH allows the user to renew his/her request or continue to with ongoing request to correct it. This feature was also introduced to improve user understanding of ANEESAH and offer flexibility in achieving their desired information from multiple dialogues.

6.4.3 English Language Dictionary

The end user evaluation and log file analysis revealed that user inputs contained spelling errors, which resulted in incorrect responses or system failure. The literature review showed that text-based dialogue systems with text correction, and validation feature can help improve performance and effectiveness. Therefore, it was decided to implement a spelling correction feature into the architecture of ANEESAH to address the negative impact spelling errors has on the utterance matching and query responses. This feature works by utilising an English language dictionary to detect and analyse user inputs for spelling related errors. The English language dictionary utilised for this purpose is known as Hunspell spell checker (Hunspell Dictionary, 2017).

Once the CA receives a user utterance, the spelling correction feature is initialised. The user utterances are evaluated and validated for spelling errors before complete utterance process is commenced. The user input is broken into words and held in a temporary short-term memory. Firstly, each word is called by spelling correction feature to match across database scripts in the knowledge base. If a word match is found in the knowledge base/recognised as named entity such as customer name, place name, product name, then that word is not proceeded for spell checking purpose. The spelling correction feature moves onto the next word in user input. If a spelling error is detected, the CA communicates it back to the user to clarify. The dictionary evaluation process makes few suggestions to predict as to what user meant to write as part of his/her input. The dictionary suggested words are further analysed through Dice Coefficient similarity measure (refer to chapter 4 section 4.3.4) to determine highest matched word(s) before displaying it to the user. The similarity match feature helps the CA in the selection of highest matched (with match strength of 1.0 that translates as 100%) word, which doesn't require user confirmation. For example, a user input from ANEESAH's log file "*Ruqayya: can you show the average sale for Mouse Pad for the Octuber, 1998 in fiscal period?*" contained a spelling error "Octuber". The spelling correction feature adjusted this input as "*can you show the average sale for Mouse Pad for the October, 1998 in fiscal period?*" and then forwarded for the matching process.

However, if string matching process fails to achieve the highest match or match strength of 100%, then a highest ranked word from matched words is suggested to the user. For example, a user input "*how many customers do we have in Japa*" is communicated by to the user for clarification i.e. "You have mentioned 'Japa' in your input, I couldn't find any matching record in the database. Did you mean to say 'Japan'?". The user can accept ANEESAH's suggestion, in which case user input is adjusted with correct word and forwarded for further processing. If the user ignores the suggestion, the user input is not altered and transferred for further processing.

This feature has been implemented to address the issue of unrecognised user inputs, noted during the initial evaluation, which resulted in the pattern matching engine failing to recognise specific part(s) of the user input.

6.5 Information Refinement Feature

ANEESAH's architecture was further developed with a novel information refinement feature to allow users to sustain dialogues and refine query produced information. The information refinement feature has enabled ANEESAH to handle refinement requests such as add, update or remove information from query produce results. The information refinement service is activated and is made available by the system followed by successful execution (e.g. a query produced results displayed and user agreement received) of a query response. This required examining of modern business and reporting applications with drill down/drill across (filter) into different aspects of reports and further interviews with end users to understand information refinement scenarios such as common query modification/enhancement requests raised by end users. Further review into existing framework and development approaches was also carried out to evaluate the most appropriate method that can be adopted to develop information refinement feature. This required modification of existing components and development of the new components that by working together as part of the architecture extend ANEESAH's abilities to support information refinement or querying the query operations. The next section (6.5.1) will illustrate how refinement requests are detected by ANEESAH's CA, followed by a section (6.5.2) on the SQL Query Refiner detailing refinement techniques and steps adopted in the process.

6.5.1 Refinement Request Detection

The ANEESAH prototype has been enabled to identify the user desire to continue the discussion about an ongoing query response. Each user input can lead to a different requirement or response from the ANEESAH system. The refinement requests can be entered in different ways and at any stage during a conversation between end users and ANEESAH. The users will perceive ANEESAH to recognise and distinguish between refinement related and other (non-refinement) requests. This was achieved by further development of ANEESAH's CA and expansion of knowledge base to include refinement scripts. The ANEESAH's CA has been modified to detect user refinement requests. Firstly, for the CA to detect a user refinement request, a successful query response must be executed. The syntax and objects used to formulate the executed query are temporarily stored in the short-term memory of ANEESAH (discussed in chapter 4). The CA evaluates the new user input against refinement scripts stored in the knowledge base. If the CA detects a link between new user input and previous query response, then it initiates the SQL query refiner module for further processing (see section 6.5.2). In the case when the CA cannot detect refinement and user input contains less information than otherwise would require to qualify for a new query-based response fully. At that point, the user is asked to clarify if his/her new request is in relation the previous response i.e. "Is this in connection with your previous request?". If the user clarifies that his/her request in not related to the previous request/query produced results, then the information stored in the temporary memory is reset and user input is treated as a new request. Following is the list of refinement types that can be used by end users to sustain dialogues to refine query produced information on a continual basis:

- AddInformation –
  If user wants to add more information into existing query results
- RemoveInformation –
  If user wants to remove information from existing query results.
- ReplaceInformation –
  Where user wants to replace information in existing query results
- AddFunction –
  Where user wants to introduce function into existing query results
- RestrictInformation –
  Where user wants to limit query produced results

The next section will provide detail on the SQL query refiner module and each refinement type mentioned in above list.

6.5.2 SQL Query Refiner Module

The SQL query refiner module has been implemented to update and adjust query information (e.g. syntax collection) to enable information refinement. The SQL query refiner module is initiated when a query response is successfully executed. Following up from a query execution, the session manager module temporarily preserves the query information (e.g. query syntax collection) in its short-term memory. Each refinement request is analysed by the CA to understand its types to determine whether if it's related to new information addition or deletion etc. Once the nature of refinement request is identified, then former query syntax is released for the refiner module to apply adjustments. The SQL query refiner module performs the following refinement features.

6.5.2.1 Add Information

The SQL query refiner has been equipped undertake refinement request to add further database information into query produced results. For example, a user input "*Show me our last quarter total sales in the UK and France*" can be refined by asking system such as "*add more countries*", "*adding order quantities*", etc. Each user input, regardless of refinement in nature, follows the complete utterance matching process, and query relevant syntax is extracted and stored in short-memory section of the session manager module (discussed in Section 6.8) along with database objects i.e. tables, column names, attributes, etc. This information is stored distinctly in the temporary memory and in addition to the existing collection of syntax/objects used to formulate the last query. Once refinement type is finalised, query related information stored in the short-memory is released, and query refiner module combines the newly requested information into the existing collection of query syntax and objects follow a sequence of steps illustrated (in Table 6.2) below.

```
If (QueryProducedResponseExists = True) Then
//A query produced response must be executed before refinement
    If (NextUserRequest = AddInformation()) Then
    //Refinement request relate to adding information
        If (RequestedInformationExistInDatabase = True ) Then
        //Number of additional entities are found from database
          Foreach If (RequestedInformationIsNotInExistingResponse = True) Then
              //Existing query produced response doesn't include request entities
                  AddInformationInExistingSyntaxCollection()
                  //Include requested entities into existing match collection
                      If (ResponseAnalyserCheck = True)
                      //Analyse response for minimum query condition
                          ReformulateQueryWithNewInformation()
                          //Reformulate query with additional syntax
                          ExecuteQueryResponse()
                          //Execute query and analyse response
                      Else (Tell user response can't be formed/rollback)
              Else (Information is already included in report)
              //Tell user information requested is already present in existing response
        Else (AskUserForClarity)
        //Ask user to rephrase his request
    Else (Do next)
    //Check for matching against other refinement functions
Else (ConsiderUserRequestAsNew)
//Take user request forward and treat it as fresh/new user request
```

Table 6.2: Algorithm for addition of information in query refinement scenario

As illustrated in Table 6.2, query refinement process is governed by algorithmic rules, which involves various steps to evaluate and ensure that information is refined correctly and in desired order. The next section will provide details on how users can ask ANEESAH to remove information from query produced results.

6.5.2.2 Remove Information

The users can also ask ANEESAH to remove information (such as column name, attributes) from query produced results. For example, a user input "*Show me our total sales from UK and France for the last quarter of 1999*" can be refined such as "*can you remove country name*", "*can you remove last quarter period*", etc. This information is stored distinctly in short-memory and in addition to the existing collection (present in the short-term memory) of syntax and objects used to formulate the last query. Once refinement type is finalised, query related information (stored in the short-memory) is released, and query refiner module finds and removes the request information from existing collection. This achieved by following a sequence of steps illustrated (in Table 6.3) below.

```
If (QueryProducedResponseExists = True) Then
//A query produced response must be executed before refinement
    If (NextUserRequest = RemoveInformation()) Then
                //Refinement request relate to remove information
        Foreach If (InformationExistInQuery = True) Then
                //Information exists in query produced results and less than 3 at a time
                    RemoveInformationFromExistingSyntaxCollection()
                    //remove requested entities from existing match collection
                        If (ResponseAnalyserCheck = True)
                        //Check if query can still be formed with minimum condition
                                ReformulateQueryWithRemainingInformation()
                                //Reformulate query with remaining syntax
                                ExecuteQueryResponse()
                                //Execute query and analyse response
                        Else (Tell user response can't be formed/rollback)
                Else (Information is not present in report/or excessive)
                //Tell user information requested is already present in existing response
    Else (Do next)
//Check for matching against other refinement functions
```

Else (ConsiderUserRequestAsNew) //Take user request forward and treat it as fresh/new user request

Table 6.3: Algorithm for deletion of information in query refinement scenario

In the case, when a user is requests to remove information that is not present in the existing query syntax collection held in short-term memory, then the user is informed and offered to renew his/her request. The next will illustrate how ANEESAH can replace information in query produced results.

6.5.2.3 Replace Information

Following a successful query response, (unless refinement is explicitly mentioned in user input that is successfully detected by ANEESAH) the user is asked to clarify if his/her new input is in relation to previously executed response. The ANEESAH's refinement features also include its ability to replace information (i.e. column name, attributes, etc.) in query produced results. For example, a user input "*Show me our total sales from UK and France for the last quarter of 1999*", can be refined by asking ANEESAH such as "replace the country name with Italy", "*can you remove the year 1999 from results?*" with different information from domain database. The user input "replace the country name with Italy" containing replacement information is processed through utterance matching process. ANEESAH ensures that database information is present in the existing query syntax collection. Once discovered the SQL query refiner performs the replacement information in user desired order. Table 6.4 shows a high-level example of the algorithm developed to replace information from query produced results:

```
If (QueryProducedResponseExists = True) Then
//A query produced response must be executed before refinement
        If (NextUserRequest = ReplaceInformation()) Then
        //Refinement request relate to replace information
                Foreach
                    If (InformationExistInQuery = True
                     && RequestedInformationExistInDatabase = True
                     && ReplacementObjects =1) Then
                      //Replacement object exists in query produced results,
                      //and new object exists in knowledgebase, and 1 at a time
                          ReplaceInformationInExistingSyntaxCollection()
                          //Replace requested entities from existing match collection
                              if (ResponseAnalyserCheck = True)
                              //Check if query can still be formed with minimum condition
                                  ReformulateQueryWithRemainingInformation()
                                  //Reformulate query with remaining syntax
                                  ExecuteQueryResponse()
                                  //Execute query and analyse response
                              Else (Tell user response can't be formed/rollback)
                     Else (Information is not present in report/or excessive)
                      //Tell user either new information doesn't exist in database or
                      //replacement object not present in existing query response or
                      // Display appropriate message
         Else (Do next)
         //Check for matching against other refinement functions
Else (ConsiderUserRequestAsNew)
//Take user request forward and treat it as fresh/new user request
```

Table 6.4: Algorithm to replace information in query refinement scenario

If the new information requested for replacement is not matched in the existing collection of last query syntax (stored in short-term memory), ANEESAH adds new information to the collection. The next section will provide detail on how users can interact with ANEESAH to add/include aggregation functions to query produced results.

6.5.2.4 Aggregation Function

The information refinement feature has enabled ANEESAH to detect and introduce aggregation functions (i.e. profit, sum, average, etc.) in query produced results. For example, a user input "*Show me product orders from UK and France for the last quarter* *of 1999*", can be refined conversationally to introduce functions such as "*give me a sum of orders*", "*what were average orders during this period*". After the user input matching process, information from the short-memory module is combined and the SQL query refiner performs the subsequent steps to adjusts query syntax collection with a new function. Table 6.5 gives a high-level overview of the algorithm used in introducing functions in query produced results:

```
If (QueryProducedResponseExists = True) Then
//A query produced response must be executed before refinement
        If (NextUserRequest = AddFuntion()) Then
                //Refinement request relate to adding function
                    If (FunctionIsNotInExistingCollection = True  //Function doesn't exist in results
                    && FunctionIdentifiedCorrectly = True               //Function requested available
                    && FunctionRelevanceToResultIdentified = True //Function relevance to results
                    && FunctionPossiblityChecked = True      //Check if function can be introduced
                    && MaxAdditionOfFunctionIs = 1) Then          //Only one function is requested
                            AddFunctionInExistingSyntaxCollection()
                            //Replace requested entities from existing match collection
                                    if (ResponseAnalyserCheck = True)
                                    //Check if query can still be formed with minimum condition
                                            ReformulateQueryWithRemainingInformation()
                                            //Reformulate query with remaining syntax
                                            ExecuteQueryResponse()
                                            //Execute query and analyse response
                                    Else (Tell user response can't be formed/rollback)
                      Else (Inform user about the problem)
                       // Display appropriate message
            Else (Do next)
 //Check for matching against other refinement functions
Else (ConsiderUserRequestAsNew)
//Take user request forward and treat it as fresh/new user request
```

Table 6.5: Algorithm to add function in query refinement scenario

In line with the query formulation abilities of the ANEESAH prototype, this feature is limited to include only one function in each query response. However, with further modifications and development, ANEESAH can be equipped to produce a query results with multiple functions applied. If the user attempts to include more than one function in a query response or request to include a different function in a pre-executed query (that already contained a function) produced results, then ANEESAH reacts to inform user about duplicate functions and offer to make selection.

6.5.2.5 Restrict Information

Another refinement type added to the prototype system is to allow users to conversationally limit query produced results. The knowledge base has been scripted to help ANEESAH to recognise user desire to limit query produced results (i.e. the number of rows, attributes, etc.). For example, a user input "*Show me our total monthly sales from UK and France for the year 1999*", can be refined in a number of ways such as "show me only just top five", "*last ten rows*". The user input is matched against knowledge base scripts to extract row limit instructions in its short-term memory. Additionally, the user desire to limit query results is evaluated to ensure that restrictions can be applied i.e. query rows/records exist, there is no existing restriction in place, etc. The query syntax is adjusted with the addition of results filter instructions before further processing. Table 6.6 shows the algorithm applied for the restriction of query results:

```
If (QueryProducedResponseExists = True) Then
//A query produced response must be executed before refinement
        If (NextUserRequest = RestrictInformation()) Then
                //Refinement request relate to restricting information
                        If (InformationIsRestrictable = True) Then
                        //Information can be restricted i.e. rows/records exists before restriction etc.
                                RestrictQueryProductResult()
                                //Restrict query result as per user input if possible
                                        if (ResponseAnalyserCheck = True)
                                        //Check if query can still be formed with minimum condition
                                                ReformulateQueryWithRemainingInformation()
                                                //Reformulate query with remaining syntax
                                                ExecuteQueryResponse()
                                                //Execute query and analyse response
                                        Else (Tell user response can't be restricted/rollback)
                        Else (result or information cannot be restricted)
        Else (Do next)
        //Take user input forward to subsequent steps
Else (ConsiderUserRequestAsNew)
//Take user request forward and treat it as fresh/new user request
```

Table 6.6: Algorithm to restrict information in query refinement scenario

When there are no query results maintained in domain database, the user is informed of an appropriate response. Further, the ANEESAH prototype has been equipped to recognise and deal with complex refinement scenarios such as a user requesting multiple refinements in a single input. The user is presented with an appropriate response "*Can you please ask only single refinement request next time. As your request contains complex refinement processes, I am renewing this session please only ask simple and single refinement request next time*". The information refinement abilities of ANEESAH allow users to engage in sustained dialogues to discover and manipulate database information with ease and at will. The next session will provide detail on complete algorithm for user utterance processing including refinement features discussed above.

6.5.3 ANEESAH NLIDB with Information Refinement

The ANEESAH's information refinement approach is considerably more complex than a single transaction query, thus requiring an algorithm for sustaining dialogues, conflict resolution and querying the querying operations. ANEESAH's architecture and algorithm were developed by further research and investigation with the intention to improve the overall effectiveness and user experience of ANEESAH and to achieve research objectives in fullness. Further development of ANEESAH's algorithm (illustrated in Table 6.7) was based on the original algorithm developed for initial prototype.

```
1    > START
2       Update Knowledgebase
3       Get user input
4        IF (User input valid = TRUE)
5            Take input forward to controller for processing (GO Step 9)
6        ELSE (Ask user for a valid/relevant input) //Allow user to make three attempts for a valid input
7              IF (Valid input violated > 3)
8                  END Session
9           Match input across domain contexts – (Database | FAQ | General)
10             IF (Input matched/Response found = TRUE)
11                    ELSE (Ask user to enter a relevant input)
12             IF (Default response found (Non-Query) = TRUE && Match Strength = TRUE)
13                   Execute Rule-based Response (Reset System (0))
14            ELSE IF (Default response found (Non-Query) = TRUE && Match Strength = FALSE)
15                IF (Match found is what was requested) // Check with user
16                    Execute Rule-based Response (Reset System (0))
17            ELSE IF (Input matched database = TRUE) // (Query-based response = TRUE)
18                         List matched query syntax   // (Keyfields, Attributes, Functions, Filer etc.)
19                               <Analyse query syntax>
20                                        IF (User Intention Exists = TRUE) // Select Statement
21                                              ELSE (Ask user to rephrase/or further clarification)
22                                        IF (Database Keyfields Exist = TRUE || Database Attributes Exist = TRUE)
                                       // Table, Column names, cell level information etc.
23                                             ELSE (Ask user to rephrase/select/clarify database information)
24                                        IF (Function Exist(s) = TRUE) //Aggregation function or sub-function etc.
25                                                   <Analyse excessive use of syntax/function in input>
26                                                          IF (Excessive Syntax Used = TRUE)
27                                                                 {Ask user to make selection}
28                                                          ELSE (Move to the next step) (GO Step 30)
29                                        ELSE (Move to the next step)
30                         IF (Minimum Query Formulation Condition Met = FALSE)
31                                {Ask user to provide missing information to meet minimum condition}
32                         ELSE (Minimum Query Formulation Condition Met = TRUE)
```

```
33                SWITCH (SQL Query Generator)
                   // Query Types engineered based on complexity levels
                  // of query syntax matched/collected from user input
34                CASE: Query Type 1
35                CASE: Query Type 2
36                CASE: ……………………
37                CASE: Query Type n
38                CASE:  Query Type
39                    {Collection of query related tables
40                     Formalise database Keyfields and Attribute
41                     Formalise functions and sub-functions
42                     Formalise appropriate joining & result filter}
44              EXECUTE (SQL Query); **Algorithm for refinement**
45          IF (SQL Query Execution/Results = TRUE)
46             Display (Display (0))
47               IF (User Agreed on Displayed Results = TRUE && Query Refinement = FALSE)
48          Reset Session (0)          // Treat next input as new request
49               ELSE IF Check User (Agreement on Displayed Results = FALSE)
50                {Ask user to rephrase request} // Try again with rephrased input
51                    (Reset Session (0))
52               ELSE IF Query Refinement Detected (Amend/Refine Query Results = TRUE)
53                <ANALYSE QUERY REFINEMENT REQUEST>
                   // i.e. Add/Remove/Replace/Restrict/function
54                  IF (User Intention Exists = TRUE) // Refinement request from user
55                       ELSE (Ask user to rephrase/or further clarification)
56                  IF (Database Keyfields Exist = TRUE || Database Attributes Exist = TRUE)
57                       ELSE (Move to the next step)
58                 IF (Function Exist(s) = TRUE)      // Aggregation function
59                       ELSE (Move to the next step)
60                 DO (APPLY ADJUSTMENT TO THE EXISTING/SYNTAX COLLECTION)
61                  IF (Query/Syntax Adjustment Successful = TRUE)
62                       ELSE (Resolve/clarify with the user)
63                       <Analyse excessive use of syntax/function in input>
64                       IF (Minimum Query Formulation Condition Met = FALSE)
65                           ELSE (Ask user to rephrase/or further clarification)
66                      IF (Minimum Query Formulation Condition Met = TRUE)
67                         {SWITCH (SQL Query Generator)
                           // See Algorithm-1: Go to Step - 33}
68                      ELSE (Ask user to rephrase/or provide clarification)
69          ELSE Reset Session (0)       //Treat new user input as new request
70     END;
```

Table 6.7: Algorithm for sustained dialogue and query refinement

6.6 Knowledge base expansion

Following up from end user evaluation, ANEESAH’s knowledge base was further expanded to address related findings and issues such failure recognised user utterances, agent's understanding of user requirements, dialogue naturalness. The findings were collected from user evaluation and system's log file, which revealed that participants perceived their conversation with ANEESAH to be low in naturalness and more formal, which can be translated that dialogues were seemed as "machinelike". The knowledge base was reconfigured and further expanded with scripts to aid end user conversation to be perceived as more human-like, casual and friendlier. This was achieved in following ways:

Firstly, domain specific knowledge was analysed and expanded with more domain specific knowledge. This process involved a further review of CA and NLIDB applications, enterprise reporting systems and further interviews with structured query language (SQL) experts to understand information requirements in a real-life environment to strengthen the knowledge base. The ANEESAH's knowledge base was expanded to provide dialogue/conversation coverage for ANEESAH to engage users and support information refinement operations. The unrecognised utterances recorded during end user evaluation, due to knowledge base weakness, were also added as new patterns in ANEESAH's knowledge base.

In addition to the domain specific knowledge, the knowledge base was also reconfigured to include scripts for other contexts namely; frequently asked questions (FAQ) to allow users to ask domain related questions, and general chat (GC) to detect and respond to non-domain related questions (see chapter 4 section 4.4.4 and 4.4.5). Finally, more domain relevant responses (including previously failed user utterances recorded in the log file, noted during initial evaluation) were scripted into the knowledge base to handle and respond to users' requests with more variety and making ANEESAH to be perceived more natural in conversations with end users.

6.7 Graphical User Interface (GUI)

The command line interface of ANEESAH prototype was replaced with a Graphical User Interface. The end user evaluation from ANEESAH prototype one revealed negative perception for the command line interface. The user comments noted during initial evaluation showed end user understanding of command line interface as plain and lacked in engagement (see chapter 4 and section 4.3.13) for command line interface). Therefore, a conventional graphical user interface was developed to improve end user interaction and to address system engagement problems noted through initial evaluation such as generic command line interface, satisfaction, effectiveness, user understanding of ANEESAH's responses. The aspects and problems related to usability problems have been considered (Nielsen and Molich, 1990). The graphical user interfaces have been evaluated as preferred interfaces for real-life environments as opposed to the use of other interface techniques (Minock, 2010; Revuelta-Martínez et al., 2013).

Figure 6.2 highlights the new graphical user interface for ANEESAH, which employs two major sections namely; ChatView showing chat history between the end user and ANEESAH, and ResultsView window for displaying query results. The system's responses (text based) are displayed in ChatView window, and query produced results are displayed ResultsView window of the interface. At start up, the system introduces itself as ANEESAH and ask the user to provide his/her name followed by greeting user and asking user "*how can I help you with sales information?*" to initialise the conversation.

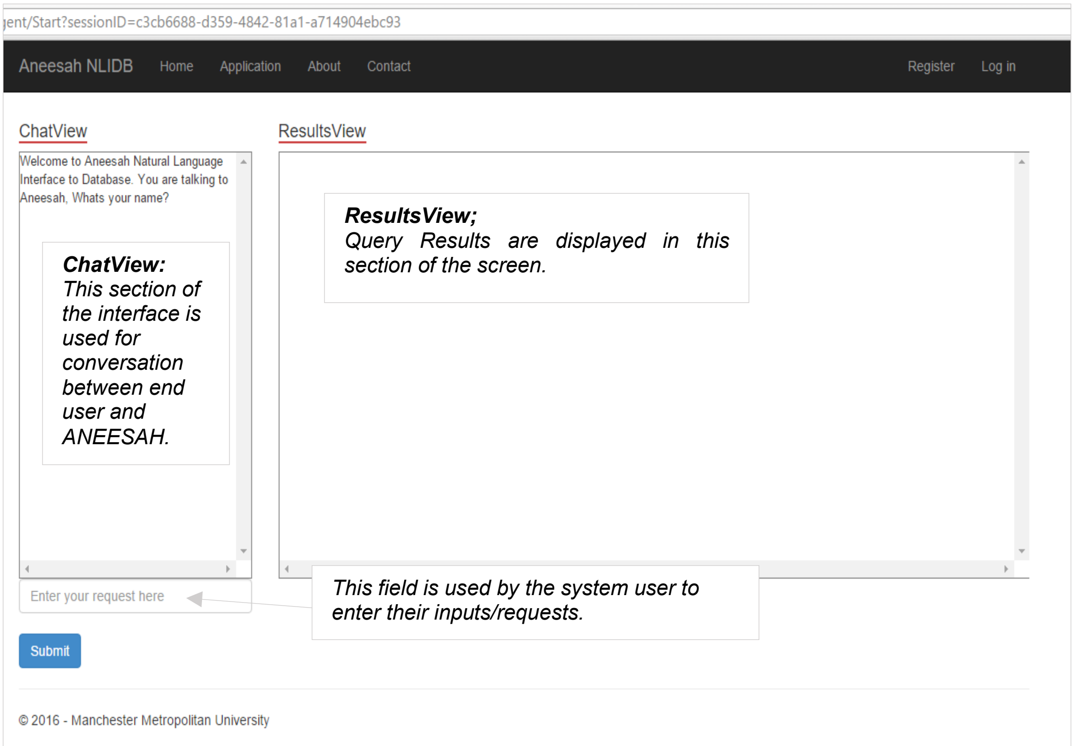


Figure 6.2: Graphical User Interface of ANEESAH prototype two

The user can use input field highlighted with text "*Enter your request here*" to enter his/her input. The both windows of new GUI are interchangeably used by ANEESAH depending upon the type of responses.

6.8 Session Manager Module

The ANEESAH's session manager module has been equipped with short-term memory feature to strengthen naturalness of the discussion and extend its abilities to support information refinement operations. In order to engage with end users through dialogues and support information refinement requires a staging memory that can be referenced, updated or removed. The incorporation of memory feature in CA's design has been emphasised, which can be used to enable agents to simulate more intelligent and human-like dialogues (O'Shea et al., 2011). The use of memory in agents is important and necessary to perform various tasks such as remembering the stage of the conversation, giving information, remembering the course of interaction and objects, and referring to old/previous tasks or topic. The important concept to consider for agents is believability in agents that can be achieved when it can imitate like a human (Brom and Lukavský, 2009b; Brom and Lukavský, 2009a).

The end user evaluation revealed that participants' general perception about ANEESAH with respect to naturalness was low. The feedback from majority participants showed that conversation with ANEESAH was machine-like and repetitive. In addition to participants' feedback, to allow sustained dialogues and perform querying the query operations required a memory feature that will enable ANEESAH to remember dialogues, ongoing conversation, database related information. Therefore, in order to address these issues, a short-term memory technique feature was developed to allow ANEESAH to remember executed responses. This feature holds query related information and the ability to relate the previous discussion with new input by asking the user "*Is this in relation to your previous request*", and perform query refinement operations. Additionally, the knowledge base was scripted to help ANEESAH clarify and deal with repetitive dialogue/input situations. For example, it helps ANEESAH to respond, when a user utterance fails to match against any context. If the same user request was entered repetitively, ANEESAH utilises short-term memory to recognise this and react more intelligently by responding with more human-like responses such as "*Sorry, I still didn't understand what you are trying to say*". Further, if the repetition happens more than permitted times (governed by three attempts rule), the conversation is terminated. Once a final response is executed, ANEESAH offers the end users to download a guiding document for the system detailing on how to use it. An example of this highlight in the following Table 6.8.

Short term memory – unrecognised utterance behaviour	
User Iteration	**ANEESAH Response**
1st	"ANEESAH: Sorry I couldn't understand what you are trying to ask. I am a sales information assistant for a computer store. You can ask information related to sales, products, profit, products etc."
2nd	"I couldn't relate your request to records in previous response. Can you please try again to ask something relevant? If you have a new requirement, please ask to renew request."
Final	"ANEESAH: I am only a computer program and my knowledge is not developed as yours. You can download helpful information about using this system by clicking here: *Download Information Document*. This is your last chance to ask sales related information."

Table 6.8 – ANEESAH's responses based on short-term memory

After the "final" response is executed, ANEESAH waits for the user to enter a valid request that it can recognise, then the conversation continues to towards the built purpose of the system. This enables ANEESAH to behave in more intelligent manner when responding to unrecognised and repeated requests. The introduction of short-term memory feature in session manager module allows ANEESAH to interact with users in more natural and intelligent style by referring to previous knowledge related to the conversation, therefore, making the conversation and responses to be perceived more natural and human-like.

6.9 Database Information Selection Tool

An information selection tool has been implemented to control the selection of database information updated in the knowledge base upon start up (discussed in chapter 4 section 4.4.6). The selection tool is implemented as an administration tool to provide control over what database information is dynamically loaded into the system's knowledge base. A section of the knowledge base is updated at system's runtime with fresh database information, which is used as part of the overall knowledge base to support utterance matching process. In real life business environments, organisations are required to alter, update or enhance databases on a regular basis to procure business needs. The database modifications are often related to changes in database schema structures such as addition or deletion of tables, or columns (Harrington, 2016). Therefore, database information selection tool has been developed to keep system's knowledge base updated with the latest information.

This tool allows information selection as to what tables and columns should be used to update at knowledge base at runtime. This tool works by calling and reading the domain database structure at runtime. The information available from the domain database tables and columns can be selected/unselected to pull the knowledge base, which then can be used as part of the overall knowledge base for the matching process. This is also useful to exclude any non-functional or database backend tables such as system tables.

6.10 Conclusion

This chapter highlighted additional research, development and techniques adopted to not only address the weaknesses revealed through during the initial evaluation but also to introduce such as sustained dialogue and querying the query operations. The architecture components enhanced and developed at this stage of the research are believed to bring improvement in the robustness and effectiveness of ANEESAH. Improvements made to the CA along with the development of new features (such as date matching, user response agreement and spelling correction) have improved the overall robustness and accuracy of ANEESAH prototype system. The introduction of information refinement feature to allow users to sustain dialogues for query refinement on a continual basis has strengthened and improved the user experience. Other supplementary components such as short-term memory, new graphical user interface and database information selection tool have been implemented to with the intention to improve overall effectiveness and user interaction experience with ANEESAH. Most significant contribution at this stage of the research as follows:

- An undated architecture to develop a conversational NLIDB with improved components and features.
- ANEESAH mimics as a human query assistant and conversationally allows users to sustain dialogues to extract and refine query produced information stored in the domain database.
- ANEESAH has been enabled to offer information refinement features such as addition new records, replacing new records, removing records from query produced information.
- Improved/Enhanced SQL engine abilities and related components to offer not only offer dynamic formulation of single transaction query but perform querying the query operations on continual basis.
- Improved CA with new features (such as date matching, user response agreement and spelling correction).
- Knowledge base improvements with extended conversational abilities to improve user interaction experience and provide wider coverage of conversational topics.
- A new graphical user interface to improve end user interaction.

The new updated architecture of ANEESAH will undergo end user evaluation to evaluate if the new development and enhancement have any positive impact on the success and effectiveness of ANEESAH compare to the first prototype. The second end user evaluation methodology and results are explained in the following chapter (7 and 8).

Chapter 7 - ANEESAH 2 Evaluation Results and Discussion (Phase Two)

7.1 Introduction

The first phase evaluation was aimed to validate Conversational Agent (CA) enabled Natural Language Interface to Database (NLIDB) framework methodology and implemented CA enabled NLIDB ANEESAH. During first phase evaluation, implemented architecture revealed weaknesses in specific components, and improvement with further development points was highlighted. The improvements were carried out with the addition of several new features and furthering the development of prototype one to achieve a complete set of research objectives.

Formulation of evaluation metrics was used to analyse different components of implemented prototype during the first phase. For phase two, formulation of evaluation metrics was based on the original selection of metrics that were used for initial evaluation. This will help in determining the success of further development and improvements made to the ANEESAH's architecture. The evaluation at phase two will also highlight the overall effectiveness of ANEESAH. The evaluation metrics selected for this purpose individually map to different features of ANEESAH, which can be used to detect and evaluate the contribution of individual components. In addition, carrying first phase evaluation metrics forward as a base for phase two evaluation will also serve as a benchmark and bring to light any significant improvements between the two prototypes.

7.2 Experimental Design

The data collection for phase two evaluation was carried out by experiments, which required test participants to interact with the system to perform an objective analysis of logs followed by completing survey questionnaires for subjective analysis. The purpose of the evaluation is primarily to gauge the success of further developed components and improvements made to the ANEESAH's framework. This will also help in examining if further development and enhancement have improved different aspects of ANEESAH's framework such as conversational abilities, query refinement, weaknesses highlighted during first phase evaluation. The data collection during experiments will aid in concluding the main research questions.

7.3 Hypothesis

The main aim of this research is highlighted through main research question as follows:

Research Question -

Can a Natural Language Interface to Database allow users to access desired database information and sustain dialogue for further refinement of information?

The research hypothesis (**H0**) with subsidiary research hypothesises mentioned below are to be evaluated through phase two evaluation by way of conducting experiments on updated architecture of ANEESAH system.

H0-A. A NLIDB cannot allow users to retrieve desired information from a database interactively.

H0-B. A NLIDB cannot allow users to perform multidimensional information analysis and further refine information produced from the database.

H0-C. A Pattern Matching approach cannot be used to build a conversational NLID successfully, capable of automating complex query formulation process.

H0-D. A conversational NLIDB cannot generate comparable results to those produced conventionally by a database expert.

The research hypothesises refer to the subjective and objective features of the updated architecture of ANEESAH. The original null hypothesis (**H0)** will be accepted or rejected based on the results gathered for subsidiary hypothesis (A, B, C, D and E).

7.4 Experiments

In line with the first phase evaluation and to collect evidence for H0, two experiments were designed to evaluate prototype two. The prototype two was expected to perform better than prototype one in phase two experiments. Primarily two participant groups namely; Group A and Group B partook in phase two experiments. There was a total of 32 participants in both groups. The Group A participants were selected on the basis of their structured query language and database knowledge. The Group B participants possessed no structured query language (SQL) and database knowledge. The Group B participants can be further divided into three categories namely; Group B.1 test users who create their reports based on queries written by other people, Group B.2 test users who use reports and queries developed by other people or applications, and Group B.3 users who have never used a database before. The intention behind using participants from diverse backgrounds was to put the system through firm testing, and their contribution will help in taking an insight into system usage patterns.

The experiments involved participants to conversely interact with ANEESAH to complete a set of test scenarios. The interaction between ANEESAH and test users is recorded in the log file that will provide data required to evaluate objective metrics. The log file was configured to capture discussions between the system and test users. The log file was also configured to store database queries that will be used to measure the success of enhanced and further developed components of ANEESAH's architecture.

The information necessary to evaluate subjective metrics was collected through experiment two, which involved participants' groups in completing survey questionnaires (see Appendix A) after interacting with ANEESAH. The survey questionnaire was amended to include questions in line with further developed features to measure users' perceptions such as their view of the query refinement feature etc. The data collected from the log file and survey questionnaires are compiled to extract subjective and objective data that is used to examine and measure the effectiveness of enhanced architecture statistically.

7.5 Participant interaction

The test participants were invited through email, and later prototype two was made available to them. The participants were informed that developed system is a prototype only and it can only produce responses related to a domain in specific Sales History sample database. The participants were briefed that scenarios are only open-ended instructions to stipulate individual tasks that ANEESAH can perform. They were told to interact with the system freely and as they felt suitable for example use of language when interacting to complete individual scenarios assigned to them. All participants were briefed on how to use the system to complete appropriate scenarios. The test participants were presented with a list of scenarios with subtasks requiring users to find database information.

The native language for selected participants varied, however, they were fluent in the English language, but their knowledge of SQL and database varied from expert to complete no knowledge of database. The scenarios instructions given to the participants had no predetermined limit or boundary. Each participant was required to read and translate scenarios questions as per his/her understanding when using the ANEESAH system. Thus, avoiding the introduction of any bias in questions the participants will ask the system during test sessions.

7.6 Evaluation Metrics Formulation

During the first phase of ANEESAH's evaluation, subjective and objective metrics were derived using goal, question, metrics (GQM) methodology. The next section (7.7) will provide detail on the selected metrics for evaluation.

7.7 Evaluation Metrics

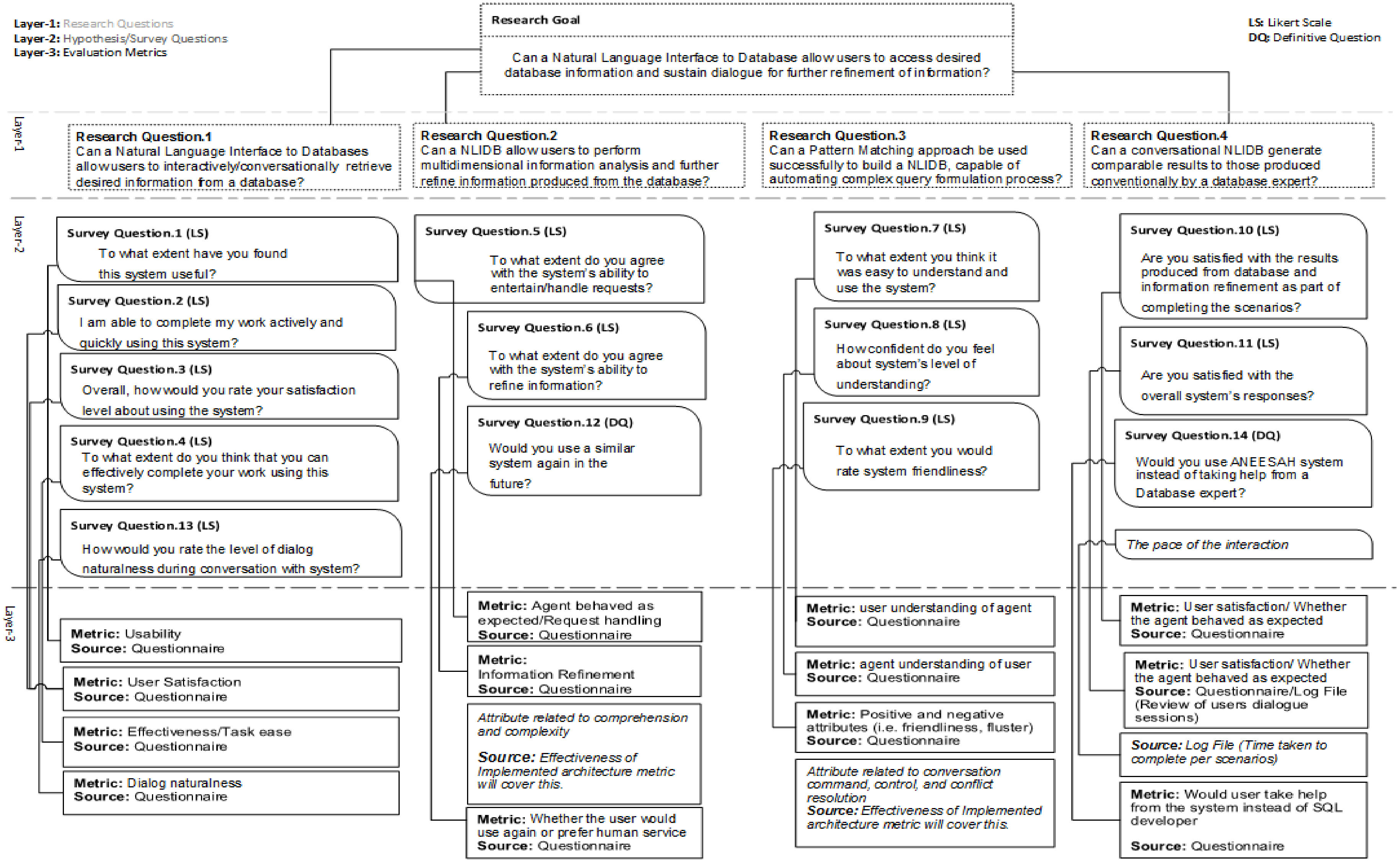
The data collected against these metrics (in phase two) will be contrasted to the set of data collected during first stage evaluation to yield any statistical difference or improvement between the two data samples. Table 7.1 illustrates "*objective metrics*" sought for evaluation, "*source*" of information and what agent "*characteristics*" are measured.

Obj. Metrics	Source	Characteristics Measured
Dialogue/ Conversation Length	Log File	• Time taken to get desired info or complete a test scenario • Time required to produce results
Count of dialogue turns	Log File PR-F	• Number of iterations/dialogues required per one scenario or all test scenarios
Various measures of success at utterance or task completion level	Log File PR-F	• Number of times correct answers/information produced by ANEESAH-2? • SQL queries executed with correct results. • SQL queries enhanced/reformulated to provide refinement
Various counts of correct actions by the agent (e.g. answering questions)		
Various counts of errors, corrections or percentage error rates	Log File PR-F	• Number of times system crashed during testing?
		• Number of times incorrect answers / info produced.
		• Number of times ANEESAH did not recognise user questions?
Precision Recall F-Measure	Log File PR-F	• Other input recognition / accuracy measures
		• If implemented architecture and query generation engine is effective?
		• Handling of test scenarios and information refinement by way of dynamic query formulation

Table 7.1: List of objective metrics

Table 7.2 reflects how research hypothesises questions are mapped to subjective metrics selected for ANEESAH's evaluation such as usability, effectiveness, user satisfaction.

Table 7.2: Goal, questions, metric model for phase two evaluation



7.8 Data Collection

The data to evaluate subjective and objective metrics have been collected as follows.

7.8.1 Subjective Data Collection

The data required to evaluate subjective metrics has been collected by way of survey questionnaire filled by participants. The structure of questionnaire has been updated to cover research questions that were not part of the first phase evaluation scope. The scope of phase two evaluation is to provide answers to all research questions.

7.8.2 Objective Data Collection

The data required to examine objective metrics will be collected from the log file, produced during participants' interactive sessions with ANEESAH. The information such as dialogues, responses, database queries is captured in the log file, which will be utilised to derive statistical analysis.

7.9 Data Analysis

The data collected during phase two evaluation will be examined and contrasted with the data collected at first phase evaluation of prototype one. Doing this will help in analysing which if any of the further developed and enhanced/improved components has a significant impact on the effectiveness and overall performance of ANEESAH. The evaluation data collected during phase two will be compiled and transformed to apply different statistical analysis techniques. The statistical analysis techniques will aid in answering the research hypothesis questions. Following up from the phase one, the phase two evaluation will also include participants with two main backgrounds namely; participants with SQL knowledge (Group A) and participants with no SQL (Group B). Table 7.3 highlights the differences between the two prototypes (one and two) of ANEESAH. In addition, this will also reflect if the participants knowledge of SQL/database have any significant difference in how both participants' groups interact with the prototype system and whether or not these variables have any impact on the performance of prototype system.

New Data	Participants with SQL Knowledge	Participants with No SQL Knowledge	Between Groups
Old Data	Participants with SQL Knowledge	Participants with No SQL Knowledge	

Table 7.3: Participants groups for data analysis

7.10 Scenarios

The prototype two was also evaluated through test scenarios (Appendix B). For phase two evaluation, the system's knowledge base has been further developed through knowledge engineering that has increased the system's abilities to include sustained dialogues and query refinement features. The scenarios devised for experiments were based on example queries and review of existing NLIDBs (discussed in chapter 2) and business applications. This process also involved taking an insight into how database information is requested in real life environments. There were seven test scenarios developed in total ranging (1-7) from simple to complex in nature. The scenarios were different from one another in terms goal achievement and included sufficient information necessary to complete the set-out tasks.

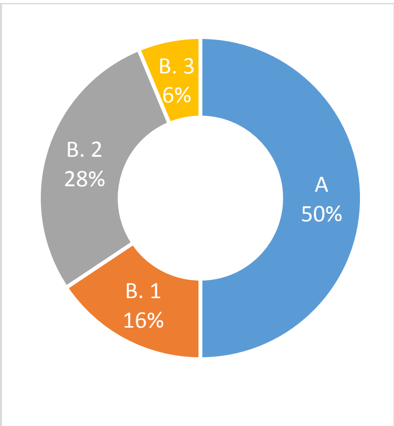
The scenarios were embedded with query formulation and refinement difficulties. For example, scenario 1 required a simple/single transaction query and scenario 7 was designed to involve complex queries and multiple refinements. Each participant from both groups (Group A and Group B) was required to complete seven scenarios. In addition, the test scenarios were developed to (or "intending to") giving participants the flexibility to select specific database information of their choice when attempting scenario 1 and scenario 2. Their selected information was later to be used by participants in completing the subsequent scenarios (3-7). This was done so that information given on scenarios sheet was not hard coded but down to participant's choice leading to different results as output. The participants' choice of information selection does not alter the outcome/nature of queries expected against each scenario. Therefore, the variations will still be comparable and gathered results would be usable for statistical analysis.

7.11 Participants Sample

The number of participants involved in first phase evaluation was 20, but for phase two evaluation this number was increased to 32. This will help in gathering more data that will lead to more decisive and conclusive results. A sample size of 32 has been found to give meaningful results in other work on CAs and NLP in the past (O'Shea et al., 2011; Pazos R et al., 2013). The sample has been divided into groups (Group A and Group B) to analyse whether their knowledge about databases has any impact on the effectiveness of ANEESAH. During the evaluation, experiment data was filtered to include fully completed test sessions only. The test data for participants who completed full experiments followed by completion of survey questionnaire was carried forward for analysis. The following section illustrates test sample distribution.

7.11.1 Sample Distribution by SQL Knowledge

Figure 7.1 shows participants' distribution chart based on different level of knowledge for structured query language and use of database information.



A - Participants who can write their database queries.

B.1 – Who create reports with queries written by others.

B.2 – Who use database reports & queries developed by others.

B.3 – Participants who never used a database.

Figure 7.1: Pie chart of sample distribution by SQL knowledge

The total sample of 32 participants can be grouped into two main categories. Group A participants with SQL/database knowledge or Group B with no knowledge of SQL/database that is further divided into subsidiary groups. Figure 7.1 shows that

50% of participants included in experiments possessed structured query language and database knowledge. These participants had this knowledge because of educational or professional background. This was followed by 28% of the original sample who use database information based reports or queries developed by experts. Furthermore, 16% of the original sample included participants who author their reports based on queries written by database experts. This proportion of the sample was drawn from a professional environment. Following this, 6% of the sample included participants who have never used database before. None of the participants involved in the evaluation will have any previous experience using ANEESAH, and the participants were not paid for their participation in the evaluation study they all volunteered to participate for altruistic reasons.

7.12 Experiments Results

The following section will detail analysis of evaluation from experiment 1 and experiment 2.

7.12.1 Experiment 1

Experiment 1 was designed to evaluate feedback from both participant groups following up from their interaction with prototype two. In order to determine the conversational abilities and interaction experience of further developed and enhanced prototype two, all test participants were presented with an evaluation questionnaire. The evaluation questionnaire was designed with a Likert scale based questions to enable participants to rate their experience between one to five points. The evaluation questionnaire also included two definitive questions answerable in yes or no. There was an open-ended question to allow users to write any comments or feedback about prototype two system as shown in Table 7.4. The evaluation metrics were translated into individual questions on survey questionnaire to evaluate different aspects of prototype two. The questions formed in below questionnaire have been used effectively to evaluate similar systems.

Participant No:

- ☐ I write my own database queries.
- ☐ I create my own reports based on queries written by other people.
- ☐ I use database reports and queries developed by other people/applications.
- ☐ To the best of my knowledge I have never used a database

"Please rate the degree to which you agree with the following on a scale from 1 to 5 where 1 means strongly/very negative and 5 means very positive.*"*

			1	2	3	4	5	
1	I found this system to be useful?	strongly disagree	☐	☐	☐	☐	☐	strongly agree
2	I am able to complete my work actively and quickly using this system?	strongly disagree	☐	☐	☐	☐	☐	strongly agree
3	Overall, how would you rate your satisfaction level about using the system?	strongly disagree	☐	☐	☐	☐	☐	strongly agree
4	I think that I can effectively complete my work using this system.	strongly disagree	☐	☐	☐	☐	☐	strongly agree
5	To what extent do you agree with the system's ability to entertain/handle requests?	strongly disagree	☐	☐	☐	☐	☐	strongly agree
6	To what extent do you agree with the system's ability to refine information?	strongly disagree	☐	☐	☐	☐	☐	strongly agree
7	I think it was easy to understand and use the system.	strongly disagree	☐	☐	☐	☐	☐	strongly agree
8	I am confident about the system's level of understanding my inputs.	strongly disagree	☐	☐	☐	☐	☐	strongly agree
9	I found this system to be user friendly.	strongly disagree	☐	☐	☐	☐	☐	strongly agree
10	I am satisfied with the results produced from database and information refinement as part of completing the scenarios.	strongly disagree	☐	☐	☐	☐	☐	strongly agree
11	I am satisfied with the overall system's responses.	strongly disagree	☐	☐	☐	☐	☐	strongly agree
12	The system's dialogue during the conversation was natural.	strongly disagree	☐	☐	☐	☐	☐	strongly agree

13. Would you use a similar system again in the future?

YES ☐ NO ☐

14. Would you use ANEESAH system instead of taking help from a Database expert??

YES ☐ NO ☐

Any further comments you may have:

Table 7.4: Questionnaire for phase two evaluation

7.12.2 Experiment 1 Results

This section will discuss questionnaire results from both participant groups. The questionnaires' data from all participants was compiled to prepare Table 7.5. Table 7.5 shows that ANEESAH prototype two was well received by test participants. The system usability question was rated at 94% where participants agreed or strongly agreed with its usefulness. Approximately, 6% participants represent neutral rating for system's usefulness. In response to question two (satisfaction) and three (task ease), prototype two received a rating at 82% and 81% where participants agreed or strongly agreed that they are satisfied with the use of the system. Overall 19% neutral rating was received for system's satisfaction and its task easing feature. In response to question four (system effectiveness), prototype two received an overall rating of 91% between agreed and strongly agreed with a neutral rating at 9%. For question five, request handling ability of the system was rated at 97% where users agreed or strongly agreed that was followed by 3% neutral rating for the same. The system's information refinement ability (question six) has been rated 88% where users agreed or strongly agreed with 13% of participants giving a neutral rating for the same. The user understanding of the system and agent understanding of user has been rated 87% and 91% between agreed and strongly agreed, respectively.

Overall 13% users gave a neutral rating for question number seven followed 9% users who gave a neutral rating for question number eight, respectively. The question number nine has been rated at 84% where users agreed or strongly agreed to ANEESAH's friendliness. There were 16% participants who gave a neutral rating for system's user friendliness. For user satisfaction on system produced results, it received 84% agreement or strong agreement of test participants with 16% showing neutral rating. Subsequently, agent behaviour as expected, and dialogue naturalness received a rating of 84% and 79%. For these metrics, 16% and 19% of participants gave a neutral rating. Finally, 91% of overall participants agreed to use ANEESAH in future and same rating for their wiliness on using a similar system with only 9% of overall participants disagreeing on either taking help from ANEESAH or using a similar system in the future.

#	Questions		1	2	3	4	5	
1	I found this system to be useful?	strongly disagree	0%	0%	6%	44%	50%	strongly agree
2	I am able to complete my work actively and quickly using this system?	strongly disagree	0%	0%	19%	38%	44%	strongly agree
3	Overall, how would you rate your satisfaction level about using the system?	strongly disagree	0%	0%	19%	56%	25%	strongly agree
4	I think that I can effectively complete my work using this system.	strongly disagree	0%	0%	9%	47%	44%	strongly agree
5	To what extent do you agree with the system's ability to entertain/handle requests?	strongly disagree	0%	0%	3%	44%	53%	strongly agree
6	To what extent do you agree with the system s ability to refine information?	strongly disagree	0%	0%	13%	47%	41%	strongly agree
7	I think it was easy to understand and use the system.	strongly disagree	0%	0%	13%	53%	34%	strongly agree
8	I am confident about the system's level of understanding my inputs.	strongly disagree	0%	0%	9%	63%	28%	strongly agree
9	I found this system to be user friendly.	strongly disagree	0%	0%	16%	28%	56%	strongly agree
10	I am satisfied with the results produced from database and information refinement as part of completing the scenarios.	strongly disagree	0%	0%	16%	53%	31%	strongly agree
11	I am satisfied with the overall system's responses.	strongly disagree	0%	0%	19%	34%	47%	strongly agree
12	The system's dialogue during the conversation was natural.	strongly disagree	0%	0%	22%	31%	47%	strongly agree
			Yes	**No**				
13	Would you use ANEESAH system instead of taking help from a Database expert?		91%	9%				
14	Would you use a similar system again in the future?		91%	9%				

Table 7.5: Questionnaire results from both participant groups

7.12.3 Experiment 1 Discussion (Group-A)

Figure 7.2 shows questionnaire results from Group A participants during phase two evaluation. Overall 93% participants have rated the system high (out of which 37.5% rated it very high) for its perceived usefulness with 6% rated at medium. The participants (87%) from Group A have rated their perceived satisfaction level for the system between high and very high with 13% (approx.) rating their satisfaction at medium level. The perceived system effectiveness and task ease (question 4) have received a high rating from 93% of the sample (out of which 43% approx. rating it at very high). The system's ability to handle user requests (question 7) has received 62.5% high rating from overall Group A sample and 37.5% rated this feature as very high. The system's ability to refine query information has received a high rating from 43% (approx.) participants with further 37% rated it as very high. However, 18% of overall sample rated this at medium level. Most participants (68% approx.) highly agreed with user understanding of the system (question 7 and 8) and further 25% rated this as very high. Only 6% of overall sample rated user understanding of the system at medium level. The system's friendliness received 81% rating between high to very high.

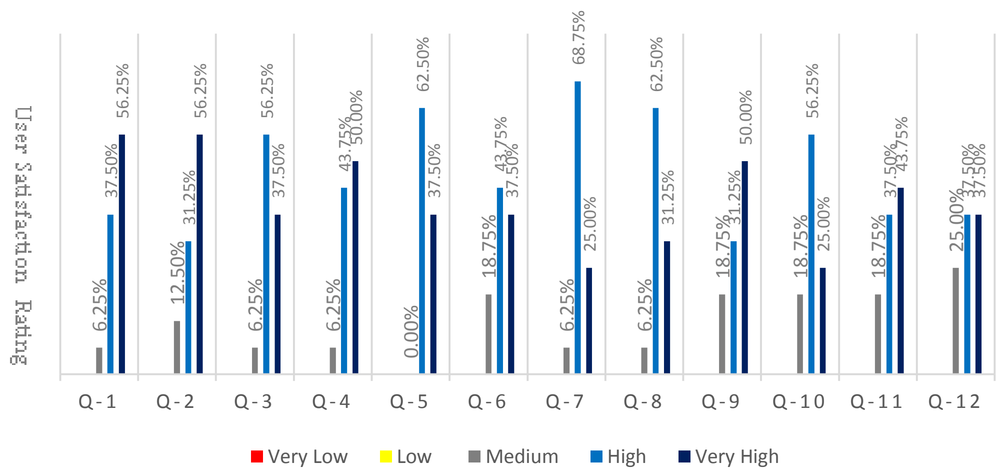


Figure 7.2: Participants rating from experimental Group A

The participant's satisfaction on ANEESAH's ability to produce database information (question 11) received 56% (approx.) high rating and 25% very high satisfied followed by 18% (approx.) showing their satisfaction level medium. The system highly satisfied 81% (approx.) participants with its responses. The system's dialogue naturalness has been rated between high and very high by 75% of Group A participants with 25% rating this measure at medium level. There were 93% participants who agreed to use ANEESAH as an alternative system instead of writing structure queries to extract information from a database. The same number of participants also agreed to use similar systems in the future.

7.12.4 Experiment 1 Discussion (Group-B)

The overall results from Figure 7.3 reflect Group B participants have well received the prototype two. The participants (50%) from Group B have rated system's usability high. There 43% participants (approx.) found system's usefulness as very high with only 6% who rated this at medium level. The system was successful in satisfying participants from this group by receiving 56% high satisfaction and 31% rating their satisfaction as very highly. The system's effectiveness has received 50% high rating from the overall participants in Group B and further 37% who have rated effectiveness at very high. The system's ability to handle the request (received 68% high and 25% very high rating) and refine information (received 50% high and 43% very high rating) has been strongly agreed. The users understanding of agent and their confidence has been rated high or above by 85% of the participants. The system's friendliness is strongly agreed by 25% of participants rating where 62% very strongly agreed to this measure.

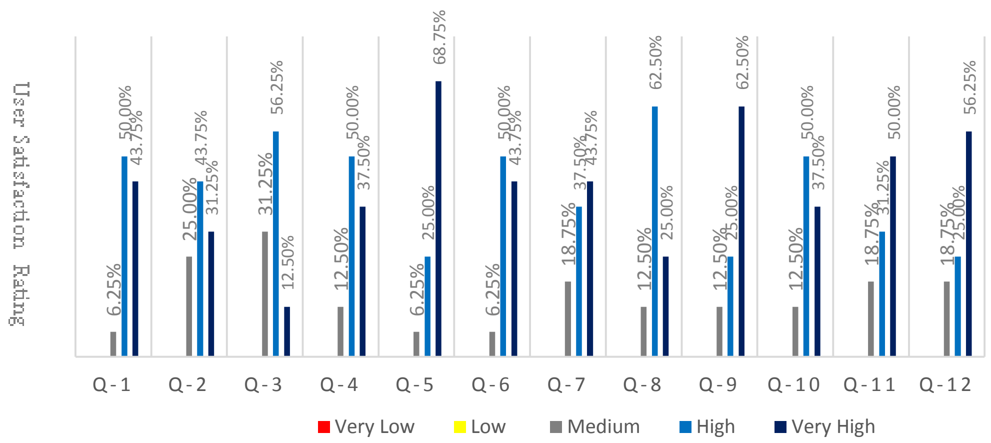


Figure 7.3: Participants rating from experimental Group B

ANEESAH's ability to produce database information also strongly satisfied 87% of participants from Group B. ANEESAH's overall responses and dialogue naturalness have been rated high or above by 80% (approx.) participants. Overall 91% of participants from Group B agreed to use ANEESAH as an alternative system than taking help from a structured query language expert and showed a willingness to use similar systems in the future.

7.12.5 Descriptive Statistics (Test of Normality)

The data collected from phase two evaluation has been used to prepare histograms (Appendix C) for visual inspection. The visual inspection of histograms will reveal the distribution of evaluation data and highlight any abnormality in distribution. The null hypothesis has been assumed that evaluation data has normally been distributed. The data from both experiment groups is analysed (Figure 7.4 is an example histogram values for question 9) through the curve of normality, which will show lead to a conclusion on acceptance of the null hypothesis.

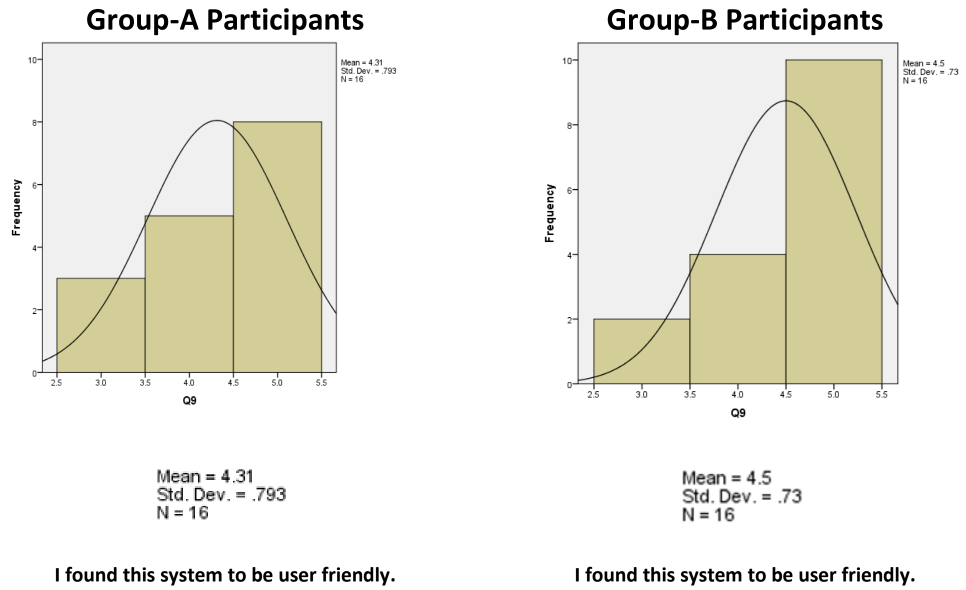


Figure 7.4: Phase two evaluation data histograms

The visual inspection reveals that the test calculated normal curve for all questions in the questionnaire used for phase two evaluation. The histograms also show there is no significant difference in how both groups rated the system. For **question 9** ("*I found this system to be user friendly.*"), the histogram shows slight abnormality in curve between Group A and Group B. This represents that the participants from Group B (participants with no SQL and database knowledge) found the prototype two more friendlier than the participants in Group A (participants with SQL and database knowledge). The test of normality shows that participants for both groups similarly perceived the system during phase two evaluation (Latham et al., 2011).

7.12.6 Selection of Statistical Test

As discussed in chapter 2 that, there exists no universally accepted benchmark evaluation methodology that can be used to evaluate conversational NLIDB. In addition, selection of a statistical test is also determined based on nature and type of research. There are two types of statistical tests namely; parametric and nonparametric. The parametric test involves the assumption of participants involved in the test to derive the data from. Non-parametric tests usually known to evaluate data that doesn't take a number of participants into consideration. Non-parametric analyses are performed to evaluate situation or occurrence of abnormal distributions, and parametric (also known as descriptive) tests are carried out with numerical values and assumption of the population. Parametric statistics can be carried out by visual inspection of histograms that can help in analysing the distribution of data, and non-parametric tests work with ordinal and categorical data. Reviewing the ratio of skewness and kurtosis in relation to standard error values can help in performing the test of normality (Doane and Seward, 2011; Gravetter and Wallnau, 1999).

7.12.7 Inferential Statistics (Mood's Median Test)

The inferential statistical analyses can assist to determine if test data is normally distributed and not based on a set of assumptions about the participants (Nolan and Heinzen, 2011). The Mood's Median test has been used (because of the sample size) to comparing the difference in ratings between two participants' groups (Group A and Group B) and highlight reason behind significant difference (if any) that can be otherwise assumed as a difference by chance. The Mood's median values for Group A and Group B are reflected in Table 7.6. Table 7.6 highlights "*Questionnaire Questions*" asked to test the participants, "*Number*" of test participants, overall "*Median*" for each question for the two groups combined, "*Exact Sig.*" significance value between both participants' groups, and "*ratings*" showing less than, equal to overall median or greater than overall median values. The significant difference between Group A and Group B will be determined based on significant values also known as “Exact Sig.” recorded when using Mood’s test for comparison. For this purpose, the H1 hypothesis will be accepted with the difference value between both groups is recorded less than 0.05. Table 7.6 reflects statistical values for both groups.

Mood's Median Test							
#	**Questionnaire Questions**	**Number**	**Median**	**Exact Sig.**	**Ratings**	**Groups A**	**Groups B**
1	I found this system to be useful?	32	4.5	0.479	<= Median	7	9
					> Median	9	7
2	I am able to complete my work actively and quickly using this system?	32	4	0.154	<= Median	7	11
					> Median	9	5
3	Overall, how would you rate your satisfaction level about using the system?	32	4	0.07	<= Median	1	5
					> Median	15	11
4	I think that I can effectively complete my work using this system.	32	4	0.476	<= Median	8	10
					> Median	8	6
5	To what extent do you agree with the system’s ability to entertain/handle requests?	32	5	0.076	<= Median	10	5
					> Median	6	11
6	To what extent do you agree with the system s ability to refine information?	32	4	0.718	<= Median	10	9
					> Median	6	7
7	I think it was easy to understand and use the system.	32	4	0.264	<= Median	12	9
					> Median	4	7
8	I am confident about the system’s level of understanding my inputs.	32	4	0.544	<= Median	1	2
					> Median	15	14
9	I found this system to be user friendly.	32	5	0.476	<= Median	8	6
					> Median	8	10
10	I am satisfied with the results produced from database and information refinement as part of completing the scenarios.	32	4	0.445	<= Median	12	10
					> Median	4	6
11	I am satisfied with the overall system’s responses.	32	4	0.723	<= Median	9	8
					> Median	7	8
12	The system’s dialogue during the conversation was natural.	32	4	0.287	<= Median	10	7
					> Median	6	9
13	Would you use a similar system again in the future?	32	1	0.544	<= Median	1	2
					> Median	15	14
14	Would you use ANEESAH system instead of taking help from a Database expert?	32	1	0.544	<= Median	1	2
					> Median	15	14

Table 7.6: Mood’ median test results

For **question 1** (“*I found this system to be useful?*”), the significant value recorded is greater than 0.05 limit. Therefore, the H1 hypothesis can be rejected as the rating from both groups does not show significant difference.

The significant value recorded for **question 2** (“*I am able to complete my work actively and quickly using this system?*”) is greater than 0.05. Therefore the H1 hypothesis can be rejected as there is no significant difference between both participants’ groups ratings.

The test showed no significant difference between the rating of Group A and Group B for **question 3** (“*Overall, how would you rate your satisfaction level with using the system*?”). The significant value recorded for both groups is somewhat different but greater than 0.05. Therefore, we can reject the alternative H1 hypothesis.

In the case of **question 4** (“*I think that I can effectively complete my work using this system.*”), Mood’s test revealed no significant difference between ratings of both groups and the significant value recorded is greater than 0.05. For question 4, the H1 hypothesis can be rejected as the value recorded is greater than the threshold limit.

The significant value recorded for **question 5** (“*To what extent do you agree with the system’s ability to entertain/handle requests?*”) near threshold limit but over a threshold value. Therefore, the alternative H1 hypothesis can be rejected, as the difference recorded is not below threshold limit of 0.05.

The **question 6** (“*To what extent do you agree with the system’s ability to refine information?*”) has a recorded value of 0.718 that is over 0.05 that shows no significant difference between both participants’ ratings. Therefore, for question 6 the H1 hypothesis can be rejected as significance value from both groups is recorded over threshold limit.

Mood’s test showed no significant difference for **question 7** (“*I think it was easy to understand and use the system*.”), and a significant value was measured higher than 0.05, therefore H1 hypothesis can be rejected for this question.

The distribution of rating from both participant groups doesn't differ significantly for **question 8** ("*I am confident about the system's level of understanding my inputs.*") as value determined by Mood's test is greater than 0.05. The H1 hypothesis can be rejected for this question in the absence of significant difference between both groups ratings.

The significant value noted for **question 9** ("*I found this system to be user friendly.*") represents that participants from both groups have relative ratings for the prototype two. The H1 hypothesis can be rejected for question 9 as significant value is recorded above the threshold limit.

Ratings given for **question 10** ("*I am satisfied with the results produced from database and information refinement as part of completing the scenarios*.") from both participants' groups show no significant difference as recorded value was above the threshold limit. The H1 hypothesis can be rejected for question 10 as the recorded value is above the minimum threshold value of 0.05.

Both groups' ratings for **question 11** ("*I am satisfied with the overall system's responses*.") does not significantly different as test value is greater than 0.05. Therefore H1 hypothesis can be rejected for this question.

The participants from both groups' ratings for **question 12** ("*The system's dialogue during the conversation was natural*.") relatively similar as the significant value was recorded above the threshold limit. The H1 hypothesis can be rejected for this question as there is not a significant difference.

For **question 13** ("*Would you use ANEESAH system instead of taking help from a Database expert?*") and **question 14** ("*Would you use a similar system again in the future?*"), the test recorded values are above threshold limit (0.05), therefore the H1 hypothesis can be rejected as there is not significant difference between both group's rating for these questions.

The above test analysis shows that ANEESAH has sufficient usability that there is no significant difference between the user experience of an expert and a non-expert, also as the scores are generally positive, it is a suitable system for non-expert users.

7.12.8 Analysis of Questionnaire Results for Prototype (one and two)

The questionnaire was divided into two parts, the first part was based on Likert scales based questions and second part consisted definitive questions (Yes/No). The both parts were designed to evaluate participants ratings from subjective aspects based on their interaction with ANEESAH (see Appendix D). The analysis of questionnaire based collected data will help in testing and concluding hypothesis questions.

ANEESAH Prototype Two Likert Scale Questions								
#	**Questionnaire Questions**		**1**	**2**	**3**	**4**	**5**	
1	I found this system to be useful?	strongly disagree	0%	0%	6%	44%	50%	strongly agree
2	I am able to complete my work actively and quickly using this system?	strongly disagree	0%	0%	19%	38%	44%	strongly agree
3	Overall, how would you rate your satisfaction level about using the system?	strongly disagree	0%	0%	19%	56%	25%	strongly agree
4	I think that I can effectively complete my work using this system.	strongly disagree	0%	0%	9%	47%	44%	strongly agree
5	To what extent do you agree with the system's ability to entertain/handle requests?	strongly disagree	0%	0%	3%	44%	53%	strongly agree
6	To what extent do you agree with the system s ability to refine information?	strongly disagree	0%	0%	13%	47%	41%	strongly agree
7	I think it was easy to understand and use the system.	strongly disagree	0%	0%	13%	53%	34%	strongly agree
8	I am confident about the system's level of understanding my inputs.	strongly disagree	0%	0%	9%	63%	28%	strongly agree
9	I found this system to be user friendly.	strongly disagree	0%	0%	16%	28%	56%	strongly agree
10	I am satisfied with the results produced from database and information refinement as part of completing the scenarios.	strongly disagree	0%	0%	16%	53%	31%	strongly agree
11	I am satisfied with the overall system's responses.	strongly disagree	0%	0%	19%	34%	47%	strongly agree
12	The system's dialogue during the conversation was natural.	strongly disagree	0%	0%	22%	31%	47%	strongly agree
			Yes	**No**				
13	Would you use ANEESAH system instead of taking help from a Database expert?		91%	9%				
14	Would you use a similar system again in the future?		91%	9%				

Table 7.7: Questionnaire results from both participant groups for Prototype-2

Although first phase evaluation metrics have been used in designing the evaluation methodology of phase two, first phase formulation of metrics didn't cover all aspects because they were either not in scope or present at the time. The phase two evaluation carried forward those metrics in addition to the new metrics included to provide conclusions for all research hypotheses. Therefore, the survey questionnaire for second evaluation combined relevant question (common) from survey questionnaire used during the first phase evaluation.

ANEESAH Prototype One Likert Scale Questions								
#	**Questionnaire Questions**		**1**	**2**	**3**	**4**	**5**	
6 (1)	I found this system to be useful	strongly disagree	0%	0%	25%	70%	5%	strongly agree
4 (2)	I am able to complete my work actively and quickly using this system.	strongly disagree	0%	0%	30%	60%	10%	strongly agree
3 (4)	I can effectively complete my work using this system	strongly disagree	0%	10%	10%	50%	30%	strongly agree
7 (5)	ANEESAH 's level of understanding your requirement	Very Low	5%	5%	25%	45%	20%	Very High
2 (7)	It was easy to understand and use the system.	strongly disagree	0%	10%	20%	50%	15%	strongly agree
5	I am able to complete my work quickly using this system.	strongly disagree	0%	5%	15%	60%	20%	strongly agree
8	I feel comfortable using this system	strongly disagree	0%	10%	25%	50%	15%	strongly agree
10 (10)	Are you satisfied with information produced from domain Database?	strongly disagree	0%	5%	20%	65%	10%	strongly agree
9 (11)	Are you satisfied with ANEESAH's dialogue responses?	Very Low	0%	0%	25%	60%	15%	Very High
1 (12)	Interface and Level of dialogue naturalness during conversation.	Very Low	0%	5%	20%	60%	15%	Very High
				Yes	**No**			
11 (13)	Would you use these kind of systems in the future?			90%	10%			
12 (14)	Would you use ANEESAH system instead of taking help from a SQL expert?			65%	35%			

Table 7.8: Questionnaire results from both participant groups for Prototype-1

Table 7.7 shows the findings of the questionnaire survey from the second evaluation of ANEESAH prototype two with enhanced and further developed architecture. Table

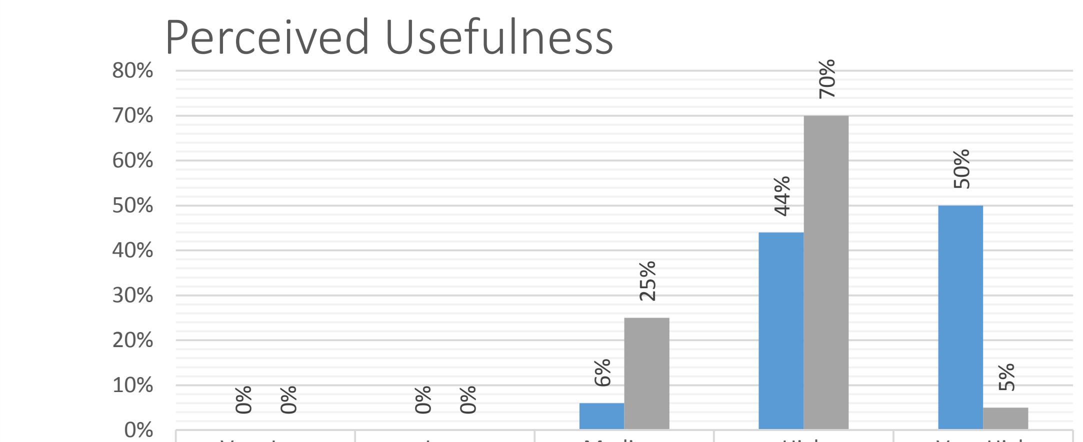
7.8 shows findings of questionnaire-based evaluation (first phase) data for prototype one.

The Mann-Whitney test is a version of the two-sample t-test for samples which don't meet the requirements of the standard t-test (e.g. non-parametric data, ordinal scale or better, etc.) The Mann Whitney U test technique is used to determine differences between two independent groups. The Mann-Whitney U is the non-parametric alternative to the t-test for independent samples. Instead of comparing means of the two groups, as in the case of the t-test, the Mann-Whitney U test compares medians. Scores are converted on the continuous variable to ranks, across the two groups and then evaluation to observe whether the ranks of the two groups differ significantly. If the calculated p-value is below the usually agreed alpha risk of 5 percent (0.05), the H1 hypothesis can be accepted, and at least one significant difference can be assumed. As the scores are converted to ranks, the actual distribution of the scores does not matter (Pallant, 2013). Table 7.9 provides Mann Whitney test statistics performed based on data collected from evaluation questionnaire.

	Q1	Q2	Q4	Q5	Q7	Q10	Q11	Q12	Q13	Q14
	Perceived Usefulness	Time Taken	Effectiveness	Request Handling	Ease of Use	Satisfaction	System Responses	Dialogue Naturalness	Willingness to Use Again	Instead of Human
Mann-Whitney U	155.00	212.00	453.500	168.5	225.00	239.50	226.50	231.50	319.50	222.00
Z-ratio	3.09406	2.02195	1.15675	2.84014	1.77744	1.50471	1.74922	1.65518	0.29096	1.83386
p-value. (2-tailed)	.002	.043	.246	.004	.075	.133	.080	.096	.771	.067

Table 7.9: Mann-Whitney u test statistics

The question 1 in evaluation questionnaire was designed to ascertain participants' perception on the usefulness of ANEESAH. Figure 7.5 illustrates the questionnaire results for the question about system's usefulness. Figure 7.5 shows that majority of test participants rated the system either "high" or "very high" during phase two evaluation. The comparison of this rating, when compared to phase first evaluation, reveals that participants perceived the prototype two better than prototype one. In addition, corroboration of these results by the Mann Whitney test in Table 7.10 shows that difference in perceptions between the two evaluations is significant (with p value = .002). The result is significant at $p < .05$.



	Very Low	Low	Medium	High	Very High
ANEESAH 2	0%	0%	6%	44%	50%
ANEESAH 1	0%	0%	25%	70%	5%

Figure 7.5: Comparison of results for question 1

The comparative analysis of participants' responses related to question about the system's activeness show (in Figure 7.6) that prototype two received overall 78% (approx.) rating as "high" or "very high. In Figure 7.6, comparison of prototype two results with prototype one for the same question reveals that 34% more participants perceived system's time taken for task completion as "very high" during phase two evaluation. Furthermore, the Mann Whitney test results for this questions in Table 7.9 show that participants' general perceptions about both prototypes are statistically significant (with p value = .043). Therefore, the H1 hypothesis can be accepted. Test participants have well received the prototype two.

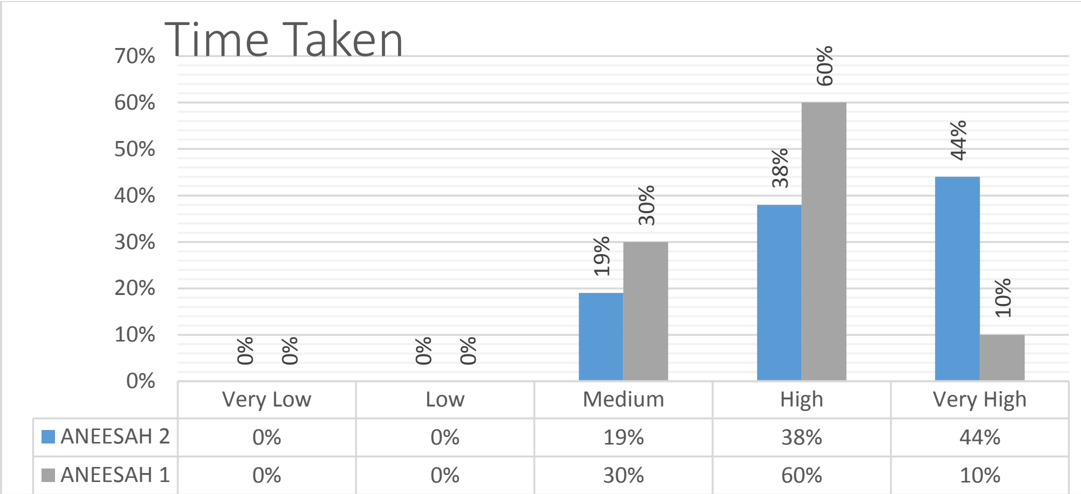


Figure 7.6: Comparison of results for question 2

Figure 7.7 illustrates that prototype two received over 91% rating “high” or “very high” for perceived effectiveness from test participants, during phase two evaluation. Figure 7.7 also represents a comparison of evaluation data for each evaluation. The comparison shows that prototype two performed well, and it was perceived better by test participants. The prototype two received 24% increased rating as “very high” during phase two evaluation based on the first phase collected results for system’s perceived effectiveness.

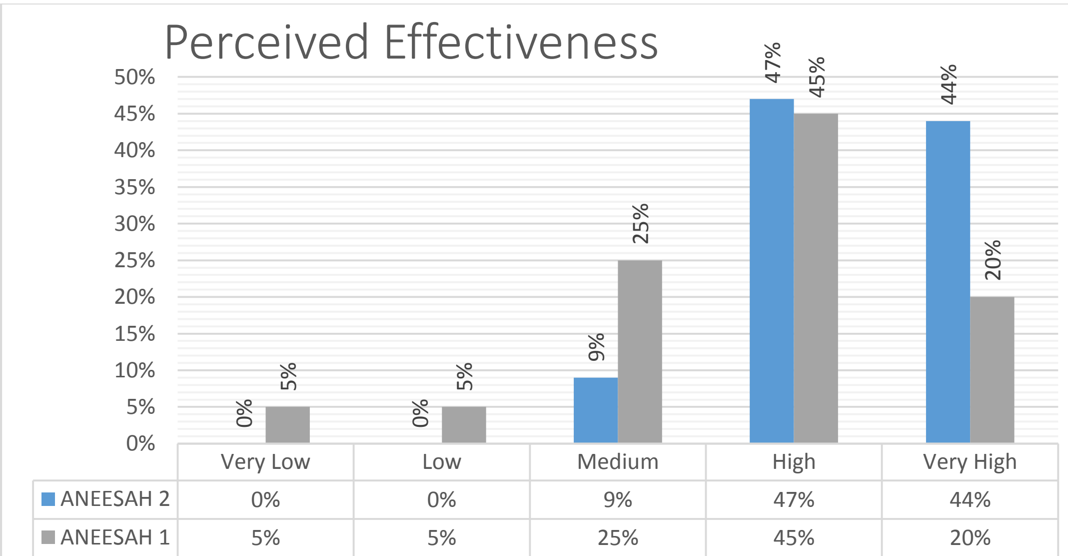
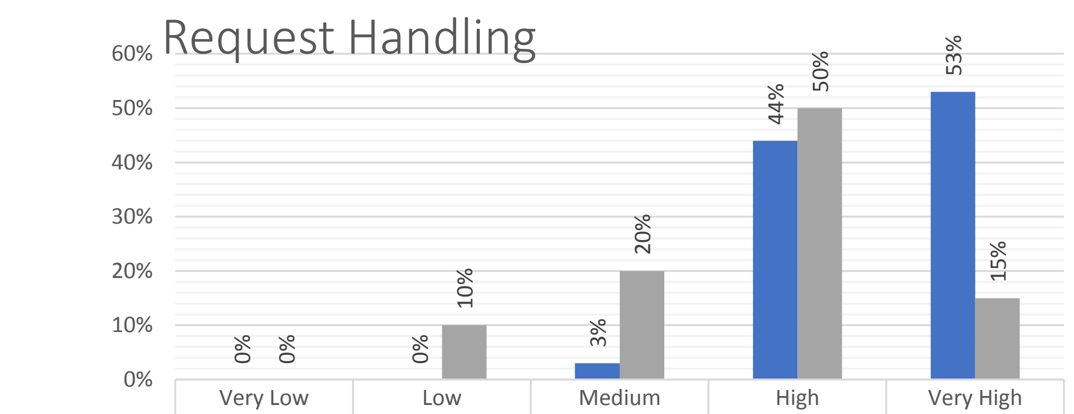


Figure 7.7: Comparison of question 4 results

Figure 7.8 highlights that the prototype two received 97% overall rating either “high” or “very high” for its request handling abilities, during phase two evaluation. The prototype two received 38% more improved rating (as “very high”) for its request handling abilities than phase first evaluation. The Mann Whitney test results show prototype two being perceived better with the difference in participants’ perceptions that are statistically significant (with a p value = .004).



	Very Low	Low	Medium	High	Very High
ANEESAH 2	0%	0%	3%	44%	53%
ANEESAH 1	0%	10%	20%	50%	15%

Figure 7.8: Comparison results for question 5

The participants from both groups gave prototype two overall rating of 87% as “high” or “very high” for its perceived ease of use, during phase two evaluation, as shown in Figure 7.9. The comparison of ratings for ANEESAH’ s ease is reflected in Figure 7.9, that shows that the prototype two was received better by test participants than prototype one. The participants rating increased 14% for the system from “high” to “very high” for its perceived ease of use. The Mann Whiney test results for this question in Table 7.9 show a significance value close to the threshold limit (p value = .075).

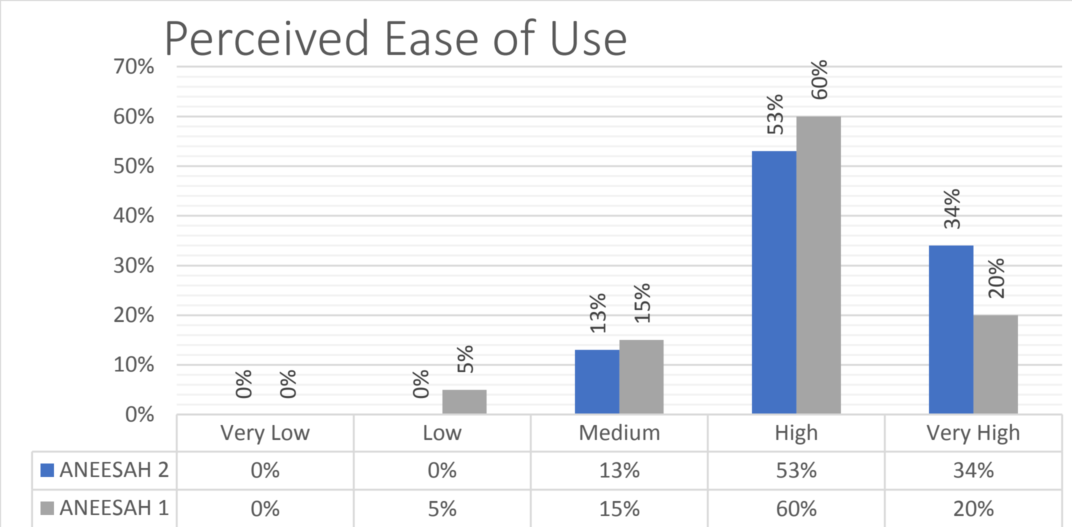


Figure 7.9: Comparison of results for question 7

The evaluation questionnaire for first and second phase evaluations was designed to examine users' perceived level of satisfaction with database information produced by the system. Figure 7.10 shows that prototype two received an improved rating from participants about their perceived satisfaction with system produced information. The comparison of both evaluation questionnaire (phase one and phase two) shows that prototype two received 21% increased (perceived) satisfaction rating from "high" to "very high" on query produced information.

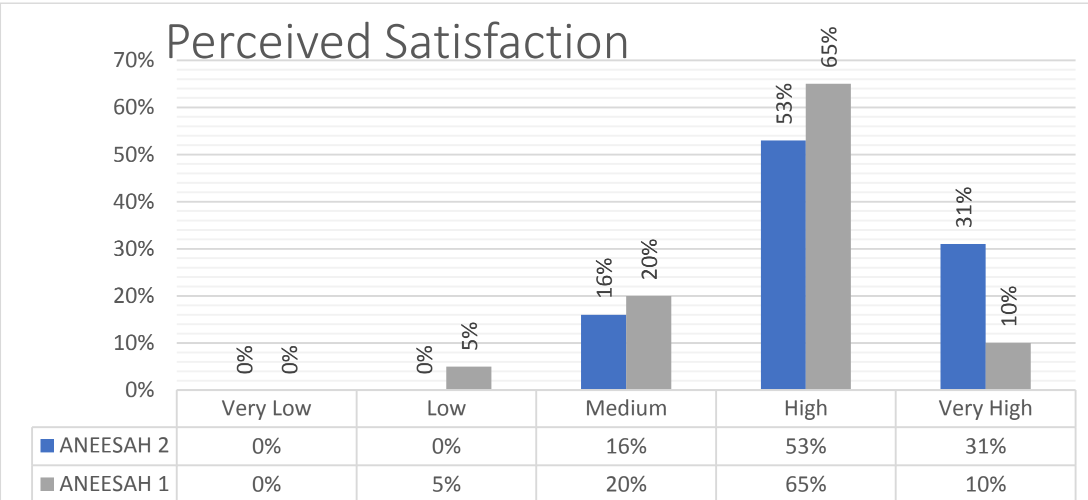


Figure 7.10: Comparison results for question 10

Following up from users interaction experience, test participants were asked to rate system's responses. Figure 7.11 shows that overall 80% of test participants rated the prototype two between “high” and “very high”. Figure 7.11 also gives a comparison between participants rating during first and second phase evaluation for the same. The comparison highlights that 32% participants have rated the prototype two “very high” when compared to the prototype one. The Mann Whitney test reveals a significance value closer to the threshold of .05 (recorded p value = .080).

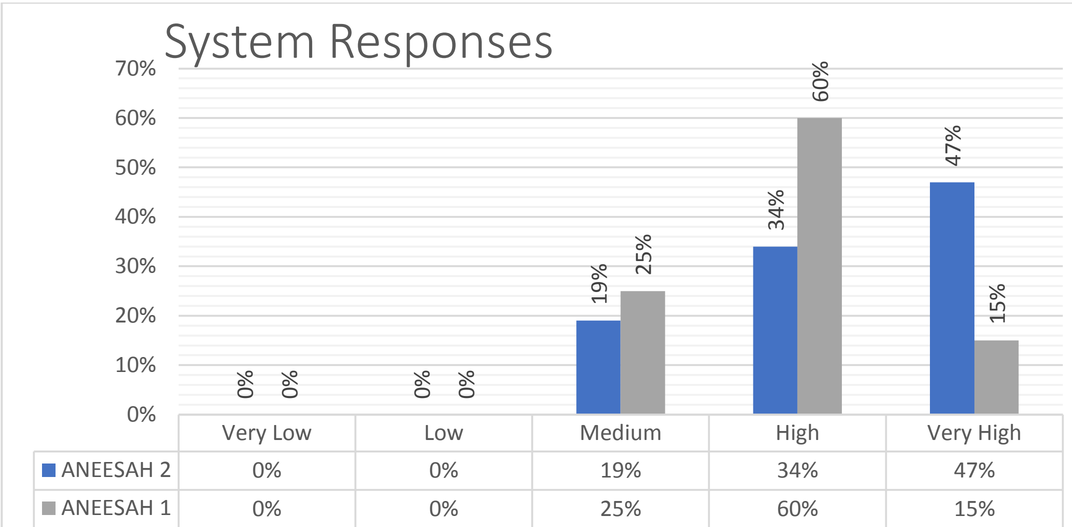


Figure 7.11: Comparison of results for question 11

Figure 7.12 shows that prototype two received an accumulative rating of 78% between “high” and “very high” for its dialogue naturalness. The datasets comparison between first and second phase evaluation reveal that there is a significant increase in participants’ rating for dialogue naturalness.

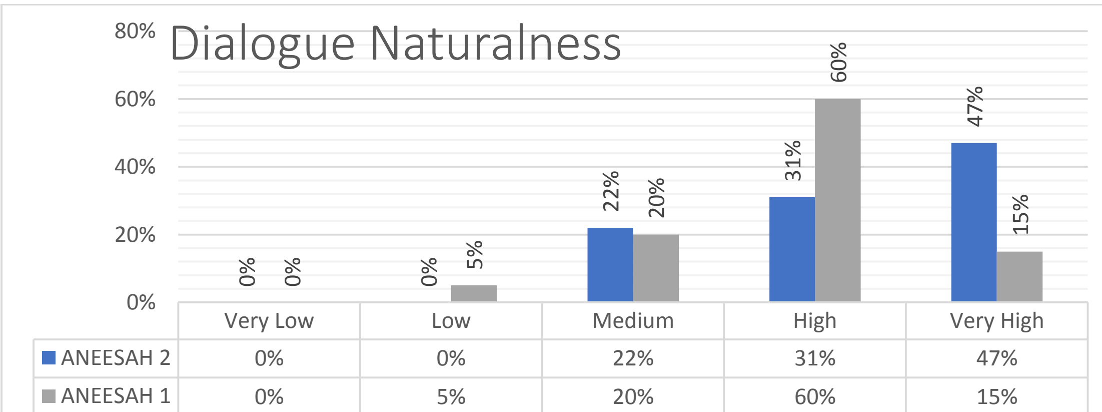


Figure 7.12: Comparison of results for question 12

The prototype two received 32% increased rating as “very high” during phase two evaluation than phase first evaluation. The Mann Whitney test in Table 7.9 recorded significance difference above the threshold value (p value = .096).

The participants rating for definitive questions (*Would you use a similar system again in the future?*) and (“*Would you use ANEESAH system instead of taking help from a SQL expert?* “) during the first and second phase, has also changed. There were 91% participants who willingness to use a similar system in the future during phase two evaluation.

This rating has not changed significantly when compared to the rating received for prototype one at 90%. The question on users’ willingness to use ANEESAH instead of using a structured query language received 91% acceptance rating, whereas during first phase evaluation participants’ acceptance rating of just 65%. This shows that the prototype two has been perceived well, which resulted in attracting 26% more participants to use ANEESAH instead of a SQL expert. The Mann-Whitney test shows the difference significance value .771 for willingness to use the same system again (question 11) in the future, and for question 12 significance value of .067 (for using ANEESAH as an alternative to a human) was recorded closer to the threshold.

The Mann-Whitney test shows that enhancements and further development of ANEESAH (prototype two) have marked impact on the overall performance. Prototype two was well received, and participants overall perception about certain aspects (such as usefulness, dialogue naturalness) has changed positively. The following section explores and analyses the questionnaire data that was gathered in order to gauge participants perceptions related to the subjective metrics.

7.13 Experiment 2

The experiment 2 is focused at analysing ANEESAH’s robustness and accuracy in producing database information as well as overall performance when interacting with end users. The following section will detail ANEESAH’s interacting abilities during phase two evaluation followed by examination of measures of accuracy and robustness.

7.13.1 Interactive Sessions

The raw data was collected from the system's log file and transformed for analysis. The prototype two handled 2296 dialogues from thirty-two participants, an average 71 of utterances per participant. Table 7.10 illustrates high-level overview of different agent "*Characteristics*", "*Description of Characteristics*" and total "*Count*". Table 7.10 has been prepared with system's log file recorded variables such as dialogues between participants and system, rejected utterances, key figures, syntax, database queries were used as a source of information for object data evaluation. The number of correct results shown in Table 7.10; represent the system's ability to interact with the participants. The summarised data shows that prototype two system perform well during end user evaluation comparing to the prototype one. The data is further transformed for examination to determine if phase two evaluation results are significantly different from phase first evaluation.

Characteristics	Description of Characteristics	Count
Utterances	Count of total user utterances during experiments	1140
Responses	Count of total system's Reponses during experiments	1158
User Utterance Failure	Number of times system failed to recognise user utterance	7
System's Understanding	Number of times system partially understood user utterance	*18*
Incorrect System Responses	Number of times system incorrectly responded	6
Completed Scenario Tasks	Number of scenarios based tasks completed successfully	210
Incomplete Scenario Tasks	Number of test scenarios failed/unfinished by the system	14
System Reset	Number of times users requested the system to reset	4
System Error/Crash	Number of times system failed due to fatal error/crashed	2
	Number of times system failed because of query refinement	1
Discussion times	Average time per experiment (mins)	12

Table 7.10: Log file analysis of data collected during phase two evaluation

Table 7.11 shows log file captured information and statistics for each scenario. The system behaviour during experiments can be seen for participants from both groups. The system's failure (showing overall effectiveness of ANEESAH) to understand user utterances is also reflected against each scenario. The test scenarios were developed on scale basic to complex. The difficulty or complexity was engineered around different factors such as utterance recognition, syntax, database information, query formulation, refinement, etc.

Table 7.11 shows that Scenario 1 and 2 were handled easily with average 2 (approx.) utterances per participant from each group (i.e. participants completed the scenario 1 and 2 with an average of 2.25 utterances per paticipants etc). Scenario 3 required average of 5 utterances per participant from both each group. Scenario 4 and Scenario 5 required average 7 and 6 utterances per participants from each group. Table 7.11 shows that prototype two system on seven occasions failed to understand (either fully or partially) user utterances when engaged in task completion for scenario 4. Scenario 6 was handled with average 4 utterances per participant from each group followed by scenario 7 ranking at an average of 9 utterances per participant from each group.

Scenario Number	Utterances	Failed to Fully/Partially recognise Inputs	Incorrect System Responses	Utterances	Failed to Fully/Partially recognise Inputs	Incorrect System Responses	Total
	Group A			*Group B*			
Scenario 1	36	0	0	40	1	0	76
Scenario 2	33	0	0	40	2	2	73
Scenario 3	75	0	0	77	0	0	152
Scenario 4	119	5	2	112	2	0	231
Scenario 5	103	0	0	91	3	0	194
Scenario 6	61	1	1	66	0	0	127
Scenario 7	138	*7*	0	149	*3*	1	287
Total	565	6	3	575	9	3	1140

Table 7.11: Number of utterances and results for both groups for each scenario

7.13.2 Utterance Distribution

Figure 7.13 reflects utterance distribution for both participant group for each scenario. The gap between both (grey and blue) lines highlights the difference between one participant group's interactive experience from the other. The plots in Figure 7.13 don't show significant differences in counts of utterances for both groups (Group A and Group B) except for scenario 5 where there is an utterance count difference of 12. Some difference in utterance distribution between both participant groups can be accounted for the flexibility in information selection by the participants.

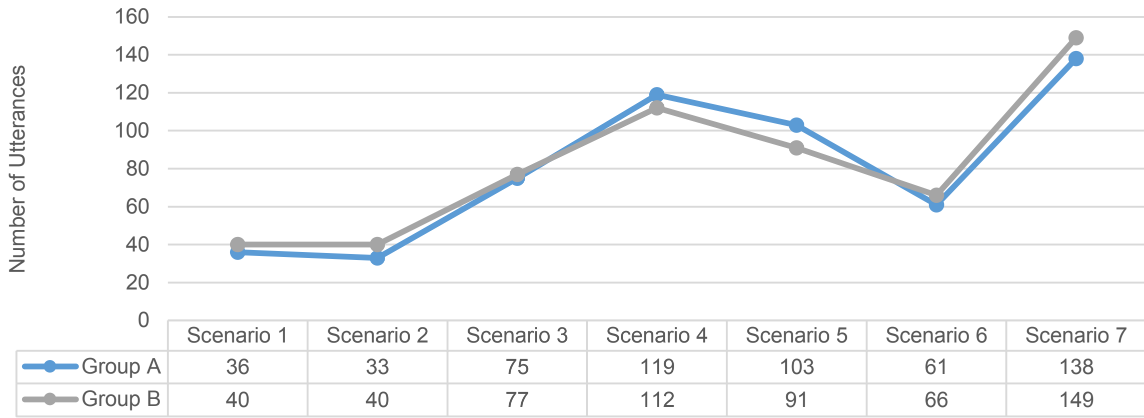


Figure 7.13: Utterance Distribution for Each Participant Group

7.13.3 AEESAH's Responses

During phase two evaluation, experiment data was recorded in the system's log file. The log file data has been analyzed, and some example dialogues between prototype two and test users have been used to discuss ANEESAH's conversational and query formulation abilities in this section. Table 7.12 shows that system behaved as expected in terms of understanding, greeting and calling user with her name throughout the experiment session. In Table 7.12, the column "*Actor*" has been used to differentiate system's responses from user inputs. The column "*Utterances/Responses*" contains actual response text between ANEESAH and test user. The column "*Attributes*" highlights background processes and actions at different levels. The system initiated the conversation by welcoming the user and asking his/her name, and once it was established that user name had been provided; the system preceded the conversation by greeting her and offered to help about sales information. The user entered her input that was matched in all context (i.e. Database, FAQ, General Chat) to find an appropriate response. In line with the Scenario 1 requirement, when asked for by the user, the query produced database information was displayed in the system interface.

Scenario 1 (Participant 1)		
Actor	**Utterances/Responses**	**Attributes**
Aneesah:	*Welcome to Aneesah NLIDB. My name is Aneesah, what is your name?*	Welcome Message
User	my name is Rebecca	User Name Provided

Aneesah:	*Nice to meet you Rebecca, what would you like to know about the sales?*	Greetings
Rebecca:	can you tell me what products you sell?	Searching Context/Syntax
Aneesah:	*I have discovered information reflected in 'ResultsView' window relevant to your input.* *Is this what you were looking to find out from the database?*	Database Context
Query – 1 ***Initial Results***	ResultsView: PROD_NAME 5MP Telephoto Digital Camera	(more rows but clipped)

Table 7.12: Example dialogue during phase two evaluation

During first phase evaluation, it was noted that the system's inability to deal with spelling mistakes led to incorrect queries or incomplete results. The prototype two has been further developed to detect and deal with spelling mistakes. Table 7.13 shows that the user mistyped spelling of "October" month as "Octuber", which was detected and replaced by the system before user utterance was further processed for query formulation. Initially, the system asked the user if her input was related to the previous query result. The user confirmed her input as a new request. The system processed the user's input as new, furthered by producing database query results and displaying them in the user interface. The second part of test Scenario 5 required results refinement for completeness. The user further asked the system to refine query produced results by providing instructions in the form of text input "*can you add Envoy External Keyboard into this*". The subsequent input from the user was processed by the system and new database matched information was added to existing query syntax for its reformulation to produce refined results.

Scenario 5 (Participant 3)		
Actor	**Utterances/Responses**	**Attributes**
Ruqayya:	can you show the average sale for Mouse Pad for the Octuber, 1998?	User utterance
Aneesah:	*Is this in connection with your previous request?*	Refinement Check
Ruqayya:	no	Searching Context/Syntax

<table>
<tr><td>Aneesah:</td><td>I have found duplicate records for October in database:
1. October in calendar_month_name
(One of the twelve divisions of the calendar year)
2. October in fiscal_month_name
(One of the twelve divisions of accounting or tax period of the company)
Please select which record did you mean or enter record number to select appropriate record?</td><td>Duplicate Database Entity/Records Detected

Clarify</td></tr>
<tr><td>Ruqayya:</td><td>2</td><td>Refine Collection</td></tr>
<tr><td>Aneesah:</td><td>Aneesah: I have found duplicate records for 1998 in database:
1. 1998 in calendar_year
(A period of time containing 365 (or 366) days or period combining 12 calendar months)
2. 1998 in fiscal_year
(Accounting or Tax period of 12 months of the company)
Please select which record did you mean or enter record number to select appropriate record?</td><td>Duplicate Database Entity/Records Detected

Clarify</td></tr>
<tr><td>Ruqayya:</td><td>2</td><td></td></tr>
<tr><td>Aneesah:</td><td>Aneesah: I have discovered information reflected in 'ResultsView' window relevant to your input.
Is this what you were looking to find out from the database?</td><td>Database Context</td></tr>
<tr><td>Query -1 Initial Results</td><td colspan="2">ResultsView:
<table>
<tr><th>PROD_NAME</th><th>FISCAL_MONTH_NAME</th><th>FISCAL_YEAR</th><th>AVG(PROFITS.AMOUNT_SOLD)</th></tr>
<tr><td>Mouse Pad</td><td>October</td><td>1998</td><td>10.85</td></tr>
</table></td></tr>
<tr><td>Ruqayya:</td><td>can you add Envoy External Keyboard into this</td><td>Refinement Check (True)</td></tr>
<tr><td>Aneesah:</td><td>Aneesah: I have discovered information reflected in 'ResultsView' window relevant to your input.
Is this what you were looking to find out from the database?</td><td>Database Context
(User Database Results Selection Remembered)</td></tr>
<tr><td>Query – 2 Refined Results</td><td colspan="2">ResultsView:
<table>
<tr><th>PROD_NAME</th><th>FISCAL_MONTH_NAME</th><th>FISCAL_YEAR</th><th>AVG(PROFITS.AMOUNT_SOLD)</th></tr>
<tr><td>Mouse Pad</td><td>October</td><td>1998</td><td>10.85</td></tr>
<tr><td>Envoy External Keyboard</td><td>October</td><td>1998</td><td>26.40</td></tr>
</table></td></tr>
</table>

Table 7.13: Example dialogue during phase two evaluation

During phase two evaluation, Scenario 7 required users to conversely ask the system to perform multiple query refinements for completeness. Table 7.14 shows that the system used its conflict resolution abilities to produce definitive query results. Following up from conflict resolution, there were no records maintained in the database for first query and the user was notified by the system about his selection. Later, the user requested to replace product "*y box*" with a different product "*64mb memory car*" but made a spelling mistake, which the system did not deem suitable for correction because of its match similarity with the database records. The system included the overarching category of products to look for the same results for all products. At the outset, the user did not realise this mistake and asked the system to include region "*Americas*" name. Further, the user realised this mistake and asked the system to include the actual product "*64MB Memory Card*". ANEESAH refined the same query a fourth time to produce user-desired results. In conclusion, the system was flexible in allowing users to make mistakes and facilitate users with a recovery process to retrieve their desired results in a sustained conversing manner. During test sessions, the system also offered users to start to renew their requests or sessions. The system successfully formulated database queries, applied refinement, and guided and led users to extract their desired database information.

Scenario 7 (Participant No 29)		
Actor	**Utterances/Responses**	**Attributes**
Andy:	can you assist me with total orders received by company for y box item in 1998	New Request
Aneesah:	*Is this in connection with your previous request?*	Refinement Check
Andy:	No	Searching Context/Syntax
Aneesah:	*Aneesah: I have found duplicate records for 1998 in database:* *1. 1998 in calendar_year* *(A period of time containing 365 (or 366) days or period combining 12 calendar months)* *2. 1998 in fiscal_year* *(Accounting or Tax period of 12 months of the company)* *Please select which record did you mean or enter record number to select appropriate record?*	Duplicate Database Entity/Records Detected Clarify
Andy:	this should be calendar_year	
Aneesah:	*I do not think there are any records present in database meeting your input. Would you like to ask information about something else?*	Database Context
Query – 1 Results	ResultsView: CALENDAR_YEAR | PROD_TOTAL | SUM(PROFITS.QUANTITY_SOLD) | PROD_SUBCATEGORY_DESC	
Andy:	i see. can you replace y box with 64mb memory car	Refinement Check (True)
Aneesah:	*Aneesah: I have discovered information reflected in 'ResultsView' window relevant to your input.* *Is this what you were looking to find out from the database?*	Database Context *(User Selection Remembered)*
Query -2 Refined Results	ResultsView: CALENDAR_YEAR | PROD_TOTAL | SUM(PROFITS.QUANTITY_SOLD) | PROD_SUBCATEGORY_DESC 1998 | TOTAL | 7291 | Memory	
Andy:	can you filter this result for only Americas	Refinement Check
Aneesah:	*Is this in connection with your previous request?*	Clarify Refinement
Andy:	Yes	Refinement Confirmed
Aneesah:	*Aneesah: I have discovered information reflected in 'ResultsView' window relevant to your input.* *Is this what you were looking to find out from the database?*	Database Context *(User Selection Remembered)*
Query-3 Refined Results	ResultsView: CALENDAR_YEAR | PROD_TOTAL | SUM(PROFITS.QUANTITY_SOLD) | PROD_SUBCATEGORY_DESC | COUNTRY_REGION 1998 | TOTAL | 4205 | Memory | Americas	
Andy:	I just need for 64MB Memory Card	Refinement Check

Aneesah:	*Is this in connection with your previous request?*	Clarify Refinement
Andy:	*Yes*	Refinement Confirmed
Aneesah:	*I do not think there are any records present in database meeting your input. Would you like to ask information about something else?*	Database Context *(User Selection Remembered)*
Query-4 Refined Results	ResultsView: CALENDAR_YEAR | PROD_TOTAL | SUM(PROFITS.QUANTITY_SOLD) | PROD_SUBCATEGORY_DESC | PROD_NAME	
Andy:	excellent. Thanks	Output Agreed

Table 7.14: Example dialogue during phase two evaluation

7.13.4 Robustness and Accuracy

Table 7.15 demonstrates an overview of ANEESAH produced database queries during phase two testing. The column "*Correct Queries*" shows queries produced by the system with correct records, "*Incorrect Queries*" were either missing user desired results or led to a system error, "*Queries with Excessive Information*" included results/information in addition to user requirement. In total, prototype two produced 513 database queries during test experiments with 10 incorrect queries followed by 4 queries which were produced with excessive information. Table 7.15 shows the effectiveness of the SQL query engine and overall performance of the prototype two system. The participants from both groups were presented with database information in response to scenario tasks during experiments. Scenario 1 and 4 were accomplished with no errors by both group participants. The number of queries executed to achieve scenario 1 and scenario 4 by both participants does not significantly differ. Table 7.15 reflects that on three occasions prototype two produced queries with excessive information during Scenario 2 and Scenario 3. Scenario 5 was highest in terms of incorrect queries. In total, there were 7 incorrect queries during scenario 5 testing followed by scenario 6 and scenario 7 with a total of 6 incorrect queries for both participants' groups.

Scenarios	Correct Queries (Group A)	Incorrect Queries	Queries with Excessive Information	Correct Queries (Group B)	Incorrect Queries	Excessive Information	Total Correct Queries	Total Incorrect Queries	Total Excessive Queries
	Group A			***Group B***					

Scenario 1	16	0	0	16	0	0	32	0	0
Scenario 2	16	0	0	15	0	1	31	0	1
Scenario 3	39	0	2	37	0	1	76	0	3
Scenario 4	40	0	0	38	0	0	78	0	0
Scenario 5	40	2	0	36	3	0	76	5	0
Scenario 6	39	1	0	39	1	0	78	2	0
Scenario 7	65	1	0	63	2	0	128	3	0
Total Queries	**255**	**4**	**2**	**244**	**6**	**2**	**499**	**10**	**4**
Total Queries Executed by the System							**513**		

Table 7.15: System produced queries for each scenario

Following sections will focus on query distribution, Precision, Accuracy and F-Measure metrics to further analyse reliability and robustness of ANEESAH system.

7.13.5 Queries Distribution Between Group A and B Participants

Figure 7.14 shows distribution of ANEESAH produced database queries for both participants' groups during the second evaluation phase. The total number of queries generated by the system are shown for each completed scenario. At the outset, chart lines in Figure 7.14 reflects that Group B (participants with no SQL and database knowledge) participants have completed test scenarios with a somewhat lower (overall) number of queries than their counterparts (participants with SQL and database knowledge). The Scenario 1 was completing by both participants' groups with sixteen database queries. Scenario 2 was completed by both groups with the difference of one database query. In completion of Scenario 3 and Scenario 4, there is a difference of two database queries between both groups. The Scenario 5 shows difference of four database queries followed by Scenario 6 with equal number of queries. Scenario 7 has difference of two database queries.

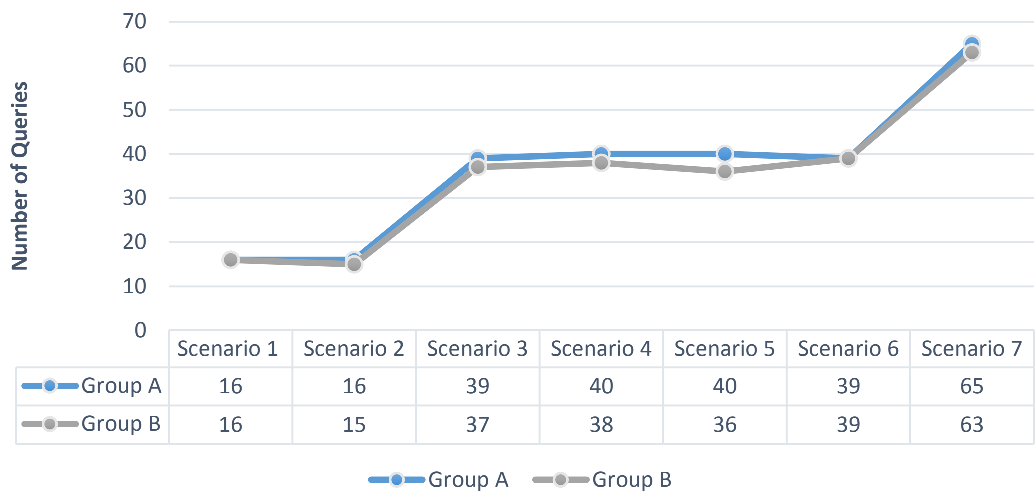


Figure 7.14 - Queries distribution for each scenario between both groups

7.13.6 Precision, Recall and F-Measure

The precision is the ratio of the number of relevant records retrieved to the total number of irrelevant and relevant records retrieved. Like precision measure (discussed in section 2.2.5 chapter 2), recall measure (discussed in section 2.2.5 chapter 2) has been used to determine the number of relevant records retrieved to the total number of relevant records in the database. Accuracy (discussed in section 2.2.5 chapter 2) has been used to determine the proportion of overall correct results.

7.13.6.1 Precision, Recall and Accuracy for Group A

The precision for Group A participants with knowledge of structured query language and database is rated at 98%. The recall for Group A participants is measured at 99%. The system accuracy for queries produced database information by ANEESAH prototype two has been rated at 96%.

7.13.6.2 Precision, Recall and Accuracy for Group B

The participants from Group B had a diverse background and possessed no structured query language or database knowledge. The log file information was analysed to determine precision, recall and accuracy for Group B participants. The precision measure for Group B participants has been rated at 97% with one percent difference when comparing to Group A participants. The Recall has been measured at 99% same as Group A participants followed by Accuracy measure at 99% again like Group A participants.

7.13.6.3 F-Measure Comparison for Prototype one and two

Finally, the F-Measure (discussed in section 2.2.5 of chapter 2) value recorded for the prototype one was calculated at 88%. The results show significant improvement made to the system with further development and enhancements with F-Measure value recorded as 98%. There is no definition for what constitutes to a correct query in the field of NLIDB. The literature shows (in chapter 2) that for the study of some NLIDBs, queries with excessive information have been considered as accurate, and for others queries with only accurate and concise information have been assumed as correct. The queries recorded in the ANEESAH's log file were analysed and only queries with correct and concise information have been considered when measured for robustness and accuracy. The queries with excessive or inadequate results were excluded from correct queries list. This rule was also adopted during the evaluation of prototype one.

7.14 Results Conclusion

The results highlight that the improvements and further development of ANEESAH's architecture (the prototype two) have a substantial impact on the effectiveness and performance when compared to prototype one. The results also show, for certain key evaluation metrics such as dialogue naturalness, precision, accuracy and recall prototype two performed better than prototype one. The results highlight that further development of new components and enhancement of existing architecture were relevant to the improvement of key evaluation metrics.

In addition, the results show that improvements, enhancements and new components added to ANEESAH's architecture have improved its conversational and query formulation abilities. Prototype two performed better than ANEESAH prototype one for key evaluation metrics from subjective and objective perspectives. The results show that new components such as query refinement, and the spell checker had the intended impact on the overall architecture of ANEESAH. The enhancements and new components come together to make ANEESAH's architecture more effective and robust when compared with the prototype one in conversation-based query formulation, and query produced information refinement.

The key findings of the results of statistical analysis reveal that prototype two of ANEESAH system has fewer unrecognised utterances when compared with prototype one. The system's ability to formulate database queries and ability to present users with information is improved. The query refinement also performed as expected as a feature added after the first evaluation.

Furthermore, the results also show improvements in subjective metric evaluation during phase two evaluation. The questionnaire comparison revealed that the prototype two not only improved its weaknesses recorded (such as failure to recognise user requirements, dialogue naturalness) during phase first evaluation but was perceived as better by test participants from various subjective perspectives such as dialogue naturalness, request handling, user interface, system understanding. In addition, all metrics that were evaluated through questionnaire during phase first evaluation were perceived as better during phase two evaluation and received an improved rating from test participants.

The evaluation data from both experiments was analysed distinctly for the participant groups (A and B) revealing an interesting (but not significant) difference in how both participant groups interacted or perceived the second prototype. For example, Group B (participants with no SQL and database knowledge) found prototype two more user friendly than their counterparts. The Group B participants also managed to complete test experiments with fewer database queries than Group A participants.

Overall the system showed improvement in test participants' perception, which is evidenced by the ratings reflected in Table 7.7 (prototype one) and Table 7.8 (prototype one), highlighting that enhancement and further development of prototype two collectively had a positive impact on ANEESAH's underlying architecture and further improved the user experience. The comments from the system participants also show that they enjoyed their interaction with ANEESAH and ratings from participants supported the system as useful tool retrieving and refining information from the database.

Chapter 8 - Conclusions, Key Findings, Contributions and Future Work

This chapter concludes the thesis by summarising the work, key findings and contributions with regards to the research hypothesis and objectives. The significance and implications of the research are given. Finally, recommendations for the direction of future research are summarised.

8.1 Overview

This research presented in this thesis has brought together the main areas of Natural Language Interfaces to Databases (NLIDBs), Conversational Agents (CAs) and language processing techniques including Natural Language Processing (NLP), Pattern Matching (PM), Sentence Similarity Measures. The main aim of this research was to contribute to the understanding of existing NLIDBs by developing a novel conversational NLIDB architecture for sustained dialogues to automate query formulation, information refinement process and perform an evaluation of the implemented prototype. The research investigations revealed several challenges in relation to the development of existing NLIBD applications such as social adaptabilities, conversational abilities, information refinement, and other language-specific issues. Based on the study (in chapter 2 section 1.2.1) into the NLIDBs development technique; the PM approach was considered as an appropriate candidate for building a conversational NLIDB due to its advantages and flexibilities. The PM approach was used to develop a novel text-based conversational NLIDB, called ANEESAH. The development of ANEESAH was completed in two parts e.g. prototype one and two. The ANEESAH's framework has been implemented with several components such as User Interface, Knowledge base, SQL engine and a CA, which comprises several sub-components such as PM engine, response analyser, query refinement module.

ANEEESAH prototype one was developed to validate the design and implementation of a novel framework to build a conversational NLIDB. Prototype one allowed users to communicate using natural language. Prototype one understood, and translated user utterances to formulate SQL queries to produce desired information stored in the dataset. Initial experiments revealed ANEESAH's ability to recognise users' requests followed by dynamic formulation of SQL queries to produce database stored information.

Further research and development decisions were made to overcome weaknesses and problems highlighted during initial experiments (chapter 5) such as failure to understand user requirements. Rectification efforts led to the investigation and development number of new components and enhancements of the existing framework. The initial evaluation of prototype one also helped in recording participants' perception about ANEESAH. Some participants' perception about ANEESAH was noted as negative e.g. their conversation lacked naturalness, ANEESAH's responses were machine-like, etc. Also, the initial evaluation results of prototype one (recorded value as 87.42%) revealed that query formulation features and information accuracy required improvement.

ANEESAH's frontend end interface also received criticism during the initial experiments. The front end of was disliked by participants and expressed as basic and unfriendly. Moreover, the scope of the initial/preliminary evaluation was not positioned to answer all research questions due to the functional limitations of prototype one. Further research and development also included an extension of features that would enable ANEESAH's framework to be evaluated/used to answer all research questions. Several new features such as information refinement, querying the query operation, spelling mistake detection were included in the existing framework. During initial experiments, the results showed that users choice of variation/variable words (to express database named entities) had a negative impact on ANEESAH's understanding. Further scripting of the knowledge base was carried out with new patterns to allow users to use variable/combination of words to express database requirement. Also, common spelling mistakes from users' utterances also showed a detrimental effect on the accuracy and robustness of ANEESAH. This issue was resolved by the implementation of a language dictionary to detect and fix spelling errors in an intelligent manner.

The second evaluation results showed that enhancements and further development (which constituted to prototype two) of the existing (ANEESAH) framework had a positive impact on overall performance and increased its accuracy, robustness and effectiveness. The second evaluation revealed that objective metrics ratings had improved significantly. Prototype two showed improved conversation experience with participants, the accuracy of information/responses has also improved, and the ability to handle refinement requests was successful/proven.

The survey questionnaire results showed that the end user's perception about the second ANEESAH prototype was improved in relation to all evaluation metrics measured. The questionnaire analysis shows that all subjective metrics have improved when compared with the questionnaire results collected for the prototype one. Improvement in the end users' perception shows that enhancement and further development of existing framework had a positive impact on addressing the weaknesses and issues highlighted through initial experiments on measured subjective metrics.

Moreover, it was shown in evaluation 2 that ANEESAH can engage users conversationally and lead the conversation to achieve their desired information from domain database. ANEESAH guides users conversationally to ensure users stay on the path pre-determined to serve its in-built purpose (e.g. to serve as a conversational NLIDB). ANEESAH can detect and deal with out of context conversations intelligently (i.e. user desire to talk about weather etc.). If the user utterance is detected as off topic, ANEESAH can steer the direction of conversation from out of context towards its goal of query formulation. ANEESAH has proven its abilities to understand user requirements followed by dynamic formulation of an SQL query to retrieve database information and further refine query produced results on a continual basis. This has only come to realisation through the novel research, investigation and development of components and algorithms that are specifically engineered to address the historical NLIDB challenges such as social adaptability, sustaining dialogues, information refinement, querying the query.

The research stimulated some novel contributions (such as the CA enabled NLIDB framework, novel query formulation and information refinement algorithms, SQL engine to dynamically formulate queries and perform querying the query operations) to address the key challenges (highlighted in chapter 2) in the field of NLIDB applications. The user utterance matching and query refinement algorithm have helped in mitigating challenges posed by the development of NLIDB applications. The developed framework can be easily applied in real-life environments with minimal scripting efforts. Features such as sustained dialogues, information refinement and querying the query operations are proof of concept that how prototype framework can adopt cross-contexts/domains with complex database structure for inexperienced users to interact and run the meaningful analysis.

Consequently, this research has resulted in the development of a conversation based NLIDB (ANEESAH) that mitigates several of the identified challenges such as social adaptabilities, conservational abilities, sustained dialogue for information refinement and dynamic querying the query operations. The research also helped in answering the main question that a general user can interact with a Natural Language Interface to Database to formulate a query to retrieve and refine desired information from a relational database. This research has provided foundational/structure for further work in the field of NLIDB to build on top.

8.2 Key Findings and Limitations

The main aim of the second evaluation was to determine if enhancements and new components added to the ANEESAH's existing framework had any impact on the effectiveness, overall performance and abilities as a conversational natural language interface to a database. The second evaluation results presented in chapter 6 reveal that the ANEESAH prototype two performed significantly better with respect to the subjective and objective metrics when compared to ANEESAH prototype one. The second prototype was well received in terms of objective task completion, the information refinement features and the querying the query operation by the test participants. The second evaluation results were used to test the following hypothesises:

The research hypothesis with subsidiary research hypotheses refers to subjective and objective features of the updated architecture of ANEESAH. The main hypothesis (H1) “a general user can interact with a NLIDB to formulate a query to retrieve and refine desired information from a relational database, successfully “is accepted or rejected based on the results of the subsidiary hypotheses (H1-A, H1-B, H1-C and H1-D).

For each subsidiary, there is a null hypothesis that there is no evidence to support the H1 variants.

H1-A. A Natural Language Interface to Database allow users to retrieve desired information from a database interactively.

H1-B. A Pattern Matching approach be used successfully to build a conversational NLIDB, capable of automating complex query formulation process.

H1-C. A Natural Language Interface to Database allow users to sustain dialogues and refine query produced information.

H1-D. A conversational Natural Language Interface to Database generate comparable results to those produced conventionally by a database expert.

The log file analysis reveals that ANEESAH’s ability to handle user requests and query formulation strengths have improved significantly. The 97% of formulated queries were correct and in line with the users' requirements towards completing the assigned test scenarios. The results analysis from Table 7.11 in chapter 7 and Table 5.4 in chapter 5 show that the number of incorrect responses produced by the second prototype have reduced to 3% from the original 7% noted during the initial evaluation. The second evaluation results suggest that enhancements made to the algorithm and existing framework had a positive impact on effectiveness and robustness of the second prototype. Analysis of the survey questionnaires also reveals that the second prototype was well received and users' interaction experience with ANEESAH (during second evaluation) has improved. The second evaluation results provide evidence to support that H1 – A can be accepted.

The development of the existing framework and new components were developed through further research and findings of the initial evaluation. The initial evaluation results revealed that the implementation of the Pattern Matching (PM) technique based engine as part of the Conversational Agent (CA) had proven its ability to understand user utterances and perform mapping onto the scripted patterns in the knowledge base to formulate query and non-query based responses. The initial prototype evaluation revealed few weaknesses such as unrecognised/partially unrecognised user utterances/inputs and ability to recognise spelling mistakes. The results reflected in Table 5.4 in (chapter 5), 7.9 and 7.10 (in chapter 7) illustrate that, during second evaluation, the PM engine handled 47.55% more user utterances when compared with the initial evaluation results. The second evaluation results show significant improvement in the framework and different components such as PM engine, SQL query engine, and knowledge base that have contributed to improve/strengthen overall conversational abilities of ANEESAH. Based on the evaluation results, there is satisfactory evidence that H1 – B can be accepted.

The enhancements and new components worked together to aid ANEESAH to perform more robustly and effectively when compared to the first prototype. The test scenarios required participants to engage with ANEESAH to produce specific database information requiring the use of multiple dialogues for refinement and queries. During second evaluation, ANEESAH two handled 1140 user utterances/inputs in total. The user utterances were divided over 32 participants (with the average of 71 utterances per participant) dealt with by the system. The log file analysis revealed (as showing in Table 7.9 and 7.10 of chapter 7) ANEESAH understood 98.70% of the overall users' utterances during second evaluation, which involved ANEESAH in handling multiple refinement requests and performing of information refinement by using querying the query operations. Based on the log file analysis and test results there is satisfactory evidence to accept H1-C.

The log file analysis presented in Table 7.10 of chapter 7 show that the SQL query engine formulated a total of 513 database queries. The log file analysis reveals that 97% of overall formulated queries attributed in steering the conversation toward successful scenario results. The 97% correct query formulation rating has improved by 13% when comparing to the first prototype results (84% noted during first evaluation). The results also show that the SQL engine abilities are strengthened with a reduction in incorrect results and excessive information to 3% from the original 16% noted for the initial prototype. The refactoring, improvement and further development led to the formulation/execution of 77.55% more database queries during the second evaluation than the first prototype's evaluation. The information accuracy metrics rating has also improved from 80% to 96.48% during the second evaluation. Based on the gathered results, there are satisfactory evidence to accept H1-D.

The comparative analysis of the second evaluation results with the first prototype results shows that the second prototype performed better in most aspects such as reduced number of unrecognised utterances and increased correct system responses. The second prototype failed to recognise 2.45% of the overall user utterances during second evaluation, that has been substantially reduced during second evaluation when compared with the original 7% unrecognised utterances recorded during the initial evaluation. The accuracy in query produced information has also improved significantly to 98% during the second evaluation when compared with 87% noted for the first prototype. Additionally, the second evaluation results show that further development and newly added components to the framework enabled ANEESAH to sustain dialogues with the end users and offered information refinement operations (querying the query) on a continual basis. Therefore, the evaluation results provide evidence that the overall null hypothesis should be rejected and the overall alternative **H1** hypothesis that "a general user can interact with a NLIDB to formulate a query to retrieve and refine desired information from a relational database, successfully" should be accepted.

8.3 Research Contributions

This main research contribution is the development of a proof of concept that a conversational NLIDB can mimic a human structured query language expert to engage users conversationally to retrieve and refine their desired information from a database.

The research and development of ANEESAH inevitably revealed several language and database specific challenges that had to be resolved so that a functional conversational NLIDB could be developed. The algorithms, development, evaluation and testing mythologies driven from this research constitute as new knowledge and contributions, which can be used as a foundation by future researchers and practitioners in the field of NLIDBs. The architecture, algorithms and methodologies developed in this research are domain independent. Therefore, this can be adapted to work with other/multiple domains. The key contributions of the research include:

8.3.1 ANEESAH - Conversational NLIDB Development

The most significant contribution of this research is the development of ANEESAH NLIDB itself, which can mimic a human structured query language expert to engage users conversationally and understand their requirements, dynamically formulate queries to retrieve their desired information stored in a database, sustain dialogues for information refinement and perform querying the query operation. Two prototypes were developed namely; prototype one (chapter 4) and prototype two (chapter 6). The development of ANEESAH steered the engineering of several novel NLIDB components such as rule-based CA that combines pattern matching and sentence similarity techniques along with newly introduced algorithms to process user utterances. Additional components of the proposed framework include the SQL engine for the dynamic formulation of queries to extract database information and perform querying the query operations to support the information refinement (discussed in chapter 3 and 5).

8.3.2 Framework and Methodology for Conversational NLIDB Development

The research and development have led to the design of a novel CA enabled NLIDB framework/methodology. The proposed methodology is generic by nature and can be adapted by future researchers and practitioners to develop similar conversational NLIDB applications. The proposed methodology provides a foundation to develop similar NLIDB systems for other domains/databases. The proposed methodology can help researchers and practitioners in overcoming historical NLIDB related development challenges such as sustained dialogues, dynamic query formulation, information refinement with querying the query operations as shown in implementation and evaluation of ANEESAH.

8.3.3 The PM engine

Among other components, ANEESAH's CA uses a PM engine. The PM engine works following pattern matching principles that incorporate novel algorithms and similarity measures to process user utterance. The PM engine has been developed to work as a rule-based response engine (discussed in chapter 3) to process user utterances in the English language. For rule-based/pre-scripted response processing, the PM engine utilises a sentence similarity technique that calculates the match strength value of a pattern against a user utterance. The implementation of sentence similarity technique has been implemented to strengthen the PM engine's matching abilities and response accuracy. The PM engine can also extract and map user utterance onto database relevant information to build a manual query response, if it fails to attract/match a rule-based/pre-scripted response. The manual response building technique extends the PM engine' conventional matching abilities to deal with some of the language specific challenges such as implicit information.

8.3.4 Scripting Language

A new scripting language has to be developed as the review on CA scripting languages (chapter 2) revealed that existing techniques did not contain features necessary to be applied to NLIDB problems such as conversational limitations, conflict resolution, querying the query operations, etc. The CA component utilises the proposed scripting language. The scripting language is developed to represent two separate sections of ANEESAH's knowledge base namely; set of rules that further comprise of scripted patterns with appropriate responses and other section contains domain scripts used to build query response manually. The proposed scripting language includes new language features such as the ability to deal with free word order user inputs. Moreover, the proposed scripting language includes variables that allow it work with implemented algorithms (discussed in chapter 3).

8.3.5 Query Formulation and Refinement Algorithm

To date, to the author's knowledge, there is not NLIDB that uses conversation with the user to refine queries. The research has led to the development of two novel algorithms. These two algorithms were demonstrated namely; first algorithm (discussed in chapter 3) implemented in prototype one and second algorithm (discussed in chapter 6) in prototype two. The first part demonstrated dynamically formulation of queries followed by their execution to extract database results (see chapter 3). The second part demonstrated an enhanced version of this algorithm by looking at refinement/querying the query operation, which meant that users could engage in conversation to refine a query which they received back from ANEESAH. The developed algorithm can be adapted by researchers and practitioners to produce similar conversational NLIDB applications for different domains.

8.3.6 The SQL Engine

The SQL engine component (chapter 5) plays a fundamental role in ANEESAH's framework to perform operations such as the dynamic formulation of database queries, information retrieval and querying the query operations. The SQL engine works in conjunction with other sub-components (i.e. SQL analyser etc.) to detect and formulate database queries based on syntax information received from ANEESAH'S CA. Before the query formulation process is executed, the syntax information is used by the SQL engine to determine its type and complexity i.e. requirement for query joins, use of any condition or filters, etc. The dynamic query formulation approach implemented in the SQL engine can be utilised by researchers and practitioners in the similar field to produce NLIDB applications with dynamic query formulation and refinement abilities (chapter 5).

8.4 Future Research

The research presented in this thesis has outlined a novel approach to developing a Conversational Agent enabled Natural Language Interface to Database with the ability to converse with users to perform dynamic formulation of queries to retrieve and refine database information. The proposed novel approach is not the definitive solution for developing conversation-based NLIDB with the ability to sustain dialogues and querying the query operation to refine database information. Rather it proves that it is possible to design and develop a conversational NLIDB. Further research that could be undertaken to enhance and strengthen ANEESAH's framework are listed below:

8.4.1 Voice Recognition

An interesting extension to this work would be to extend ANEESAH's conversational user interface to integrate speech recognition feature. The speech recognition feature will increase flexibility and widen access to the audience from diverse backgrounds i.e. people who cannot use computers, or people with disabilities, etc. The speech recognition will also help in reducing the errors resulted from mistakes user inputs such as spelling errors and other related mistakes etc. The voice recognition feature has been described useful in improving human machine experience. A similar spoken dialogue system can be developed and deployed in any information/database operating environment (Jurafsky and Martin, 2014). Furthermore, speech recognition could be accomplished for use on portable devices, thus reducing the load on other components and computational complexity that will add to its scalability and deployment on a wider scale.

8.4.2 Universal Web Service

The introduction of a universal web service from the CA component can transpose ANEESAH into a platform independent application, which means that it can access from any internet enabled device. This feature will make ANEESAH more flexible and susceptible for its deployment in cross platform real life environments. The framework can be deployed in cloud service to take advantages of modern day hosting platforms such as Platform as a Service (PaaS), Software as a Service (SaaS) etc. An application programming interface based extension would enable ANEESAH to receive and handle user requests coming through cross-platform devices and send responses encoded with relevant information (i.e. query/non-query based information, etc.) using platform independent data-interchange format such JavaScript Object Notation (JSON). The responses will be translated depending on the device in question, thus making ANEESAH usage less processor intensive on the client side.

8.4.3 Dynamic Knowledgebase for Link Responses/Analysis

Further work can be directed to extend ANEESAH's abilities to entertain complex user requests that are currently not possible for example as a user request asking to formulate query with multiple aggregation functions or refinements operations. Moreover, following up from a query produce response, ANEESAH seeks user agreement to determine if the information displayed is correct. The user agreement on the query produced information is used as a trigger to store both user utterance and query in the knowledge base and is used for future matching. Further work can be carried to enable ANEESAH to use machine learning techniques to predict link responses based on historical chat sessions and query patterns. The information stored in the knowledge base combined with the log file can be used to predict most commonly asked questions/queries and then fire link responses based on usage pattern/behaviour. This feature will enable ANEESAH to behave reactively in conversation and be suggestive about analysis and information that might be of interest to its intended users.

8.4.4 Graphical Representation of Query Results

Graphical representation of information is vastly used in every type of data or report, which makes it easier for the reader to understand and it has its advantages. Another interesting extension to the ANEESAH's framework would be for it to represent query produced information in the graphical format (such as “show me regional sales for all products in a bar chart format”) i.e. pie, charts, etc. Graphical representation of query produces information will help in several scenarios such as comparative analysis, decision making, less effort and time to review information. The users of ANEESAH can conversationally choose/switch to view query produced results in a table view or graphical format. The existing framework can provide the foundation work for this feature to further engineered with minimal efforts.

8.4.5 Knowledge base expansion

Another interesting direction that future research could take is to expand the knowledge base for wider coverage of topics and business areas. The knowledge base is modular by nature, therefore would support knowledge scripting of other/different domains. For evaluation, ANEESAH's framework was developed with a database containing sample Sales records for an imaginary computer/electronics business. Further enhancements, configuration and importing of knowledge from of a different business can make ANEESAH a single point of entry to access all business information or queries such as Customer Relationship Management (CRM), Human Resources (HR), etc.

8.4.6 Evaluation Framework for Conversational NLIDB

There exists no universally accepted benchmark or standard for the evaluation of NLIDB applications. Therefore, the proposed evaluation framework was developed, which combines evaluation of CA enabled/conversational NLIDB from subjective and objective aspects to determine the overall performance of developed systems. The framework employs Goal Question Metric (GQM) software evaluation methodology in conjunction with the selection of evaluated metrics to determine the expected outcome/performance from a developed system. The framework enables evaluation of similar NLIDB systems to be uniquely/individually tested. For example, development of two similar natural language systems aimed to achieve different goals and objectives would require combinations of different/relevant subjective and objective metrics for appropriate evaluations. The evaluation framework offers the flexibility and is adaptable to suit development specific goals such as the development of a similar system. The proposed evaluation framework can be used as a benchmark by future researchers and practitioners to evaluate the development of similar conversational NLIDB systems. These research contributions are expected to be of value to researchers and practitioners in the fields of CAs and conversational NLIDBs.

8.4.7 Cross-database searching

There are significant challenges involved in searching knowledge fused from multiple independent databases. Conversational features of ANEESAH could assist in instructing the SQL Engine in combining the databases.

8.5 Take Home Message

The main challenge of previous work on NLIDB has been in dynamically and automatically formulating a query in response to natural language input, as opposed to selecting a pre-written template from a library (or failing to respond where no match is available). This thesis provides evidence that it is possible for an automated system to produce a bespoke query in response to natural language and to refine or correct such a query through further dialogue. Consequently, it makes a novel and valuable contribution to the NLIDB field.

References

AKULA, A., SANGAL, R. & MAMIDI, R. A Novel Approach Towards Incorporating Context Processing Capabilities in NLIDB System. IJCNLP, 2013. 1216-1222.

ALGHAMDI, A., OWDA, M. & CROCKETT, K. 2017. Natural Language Interface to Relational Database (NLI-RDB) Through Object Relational Mapping (ORM). *Advances in Computational Intelligence Systems.* Springer.

ALOBAIDI, O. G., CROCKETT, K. A., O'SHEA, J. D. & JARAD, T. M. Abdullah: An intelligent arabic conversational tutoring system for modern islamic education. Proceedings of the World Congress on Engineering, 2013.

ANAND, P. and FAROOQUI, Z., 2017. Rule based Domain Specific Semantic Analysis for Natural Language Interface for Database. International Journal of Computer Applications, 164(11).

ANDROUTSOPOULOS, I., RITCHIE, G. D. & THANISCH, P. 1995. Natural language interfaces to databases–an introduction. *Natural language engineering,* 1**,** 29-81.

BATES, M. Rapid porting of the parlance natural language interface. Proceedings of the workshop on Speech and Natural Language, 1989. Association for Computational Linguistics, 83-88.

BECKER, C., KOPP, S. & WACHSMUTH, I. 2007. Why emotions should be integrated into conversational agents. *Conversational informatics: an engineering approach***,** 49-68.

BINOT, J., DEBILLE, L., SEDLOCK, D. & VANDECAPELLE, B. 1991. Natural language interfaces: a new philosophy. *SunExpert Magazine,* 2**,** 67-73.

BROM, C. & LUKAVSKÝ, J. Towards more human-like episodic memory for more human-like agents. International Workshop on Intelligent Virtual Agents, 2009a. Springer, 484-485.

BROM, C. & LUKAVSKÝ, J. 2009b. Towards More Human-Like Episodic Memory for More Human-Like Agents. *In:* RUTTKAY, Z., KIPP, M., NIJHOLT, A. & VILHJÁLMSSON, H. H. (eds.) *Intelligent Virtual Agents: 9th International Conference, IVA 2009 Amsterdam, The Netherlands, September 14-16, 2009 Proceedings.* Berlin, Heidelberg: Springer Berlin Heidelberg.

CAI, G., WANG, H., MACEACHREN, A. M. & FUHRMANN, S. 2005. Natural conversational interfaces to geospatial databases. *Transactions in GIS,* 9**,** 199-221.

CARBONNELL, J. G., JOSHI, A., KAPLAN, J., WALTZ, D. L. & PETRICK, S. 1982. Gary G. Hendrix, Chairperson. *American Journal of Computational Linguistics,* 8**,** 57.

CASSELL, J. 2000. *Embodied conversational agents*, MIT press.

CASTILLO, O., MELIN, P., PEDRYCZ, W. & KACPRZYK, J. 2014. *Recent Advances on Hybrid Approaches for Designing Intelligent Systems*, Springer.

CHU, H.-C., HWANG, G.-J., HUANG, S.-X. & WU, T.-T. 2008. A knowledge engineering approach to developing e-libraries for mobile learning. *The Electronic Library,* 26**,** 303-317.

CHURCH, K. & PATIL, R. 1982. Coping with syntactic ambiguity or how to put the block in the box on the table. *Computational Linguistics,* 8**,** 139-149.

COHEN, P. R. The role of natural language in a multimodal interface. Proceedings of the 5th annual ACM symposium on User interface software and technology, 1992. ACM, 143-149.

COPESTAKE, A. & JONES, K. S. 1990. Natural language interfaces to databases. *The Knowledge Engineering Review,* 5**,** 225-249.

CROCKETT, K., BANDAR, Z., O'SHEA, J. & MCLEAN, D. 2009. Bullying and debt: Developing novel applications of dialogue systems. *Knowledge and Reasoning in Practical Dialogue Systems (IJCAI)***,** 1-9.

CROCKETT, K., LATHAM, A., MCLEAN, D. & O'SHEA, J. A fuzzy model for predicting learning styles using behavioral cues in an conversational intelligent tutoring system. Fuzzy Systems (FUZZ), 2013 IEEE International Conference on, 2013. IEEE, 1-8.

CROCKETT, K., O'SHEA, J. & BANDAR, Z. 2011. Goal Orientated Conversational Agents: Applications to Benefit Society. *In:* O'SHEA, J., NGUYEN, N. T., CROCKETT, K., HOWLETT, R. J. & JAIN, L. C. (eds.) *Agent and Multi-Agent Systems: Technologies and Applications: 5th KES International Conference, KES-AMSTA 2011, Manchester, UK, June 29 – July 1, 2011. Proceedings.* Berlin, Heidelberg: Springer Berlin Heidelberg.

CUI, H., KAN, M.-Y. & CHUA, T.-S. 2007. Soft pattern matching models for definitional question answering. *ACM Transactions on Information Systems (TOIS),* 25**,** 8.

DAMLJANOVIC, D., AGATONOVIC, M. & CUNNINGHAM, H. FREyA: An interactive way of querying Linked Data using natural language. Extended Semantic Web Conference, 2011. Springer, 125-138.

DOANE, D. P. & SEWARD, L. E. 2011. Measuring skewness: a forgotten statistic. *Journal of Statistics Education,* 19**,** 1-18.

ELMASRI, R. & NAVATHE, S. 2010. Database systems: models, languages, design, and application programming. 6th Global Education. Pearson Education (US), New Jersey.

FADER, A., ZETTLEMOYER, L. S. & ETZIONI, O. Paraphrase-Driven Learning for Open Question Answering. ACL (1), 2013. Citeseer, 1608-1618.

FARAJ, A., ZUBI, Z. S., ALMAGROUS, A. A. & EL-FEGHI, I. S. 2009. Generic interactive natural language interface to databases (ginlidb). *International Journal of Computers,* 3.

FENTON, N. & BIEMAN, J. 2014. *Software metrics: a rigorous and practical approach*, CRC Press.

FENTON, N. E. & PFLEEGER, S. L. 1998. Software Metrics: A Rigorous and Practical Approach: Brooks. Cole.

FORBES-RILEY, K. & LITMAN, D. 2011. Designing and evaluating a wizarded uncertainty-adaptive spoken dialogue tutoring system. *Computer Speech & Language,* 25**,** 105-126.

GIORDANI, A. & MOSCHITTI, A. Semantic mapping between natural language questions and SQL queries via syntactic pairing. International Conference on Application of Natural Language to Information Systems, 2009. Springer, 207-221.

GRAVETTER, F. J. & WALLNAU, L. B. 1999. Essentials of statistics for the behavioral sciences . Pacific Grove, CA: Brooks. *Cole Publishing Company. Copyright restrictions may apply. doi,* 10**,** 0278-7393.31.

GROSZ, B. J., APPELT, D. E., MARTIN, P. A. & PEREIRA, F. C. 1987. TEAM: an experiment in the design of transportable natural-language interfaces. *Artificial Intelligence,* 32**,** 173-243.

HAMAZ, K. & BENCHIKHA, F. 2017. A novel method for providing relational databases with rich semantics and natural language processing. *Journal of Enterprise Information Management,* 30**,** 503-525.

HARRINGTON, J. L. 2016. *Relational database design and implementation*, Morgan Kaufmann.

HENDRIX, G. G., SACERDOTI, E. D., SAGALOWICZ, D. & SLOCUM, J. 1978. Developing a natural language interface to complex data. *ACM Transactions on Database Systems (TODS),* 3**,** 105-147.

HILLIARD, R. 2000. Ieee-std-1471-2000 recommended practice for architectural description of software-intensive systems. *IEEE,* http://standards. *ieee. org,* 12**,** 2000.

JOHNSON, T. 1984. Natural language computing: the commercial applications. *The Knowledge Engineering Review,* 1**,** 11-23.

JURAFSKY, D. & MARTIN, J. H. 2014. *Speech and language processing*, Pearson.

KALEEM, M., O'SHEA, J. D. & CROCKETT, K. A. Word order variation and string similarity algorithm to reduce pattern scripting in pattern matching conversational agents. Computational Intelligence (UKCI), 2014 14th UK Workshop on, 2014. IEEE, 1-8.

KARANDE, N. & PATIL, G. 2009. Natural language database interface for selection of data using grammar and parsing. *World Acad Sci Eng Tech,* 3**,** 11-26.

KAUR, A. & BHATIA, P. G. 2010. *Punjabi Language Interface to Database.*

KERRY, A., ELLIS, R. & BULL, S. 2009. Conversational agents in E-Learning. *Applications and Innovations in Intelligent Systems XVI.* Springer.

KHOURY, R., KARRAY, F. & KAMEL, M. S. 2008. Keyword extraction rules based on a part-of-speech hierarchy. *International Journal of Advanced Media and Communication,* 2**,** 138-153.

KONDRAK, G. Combining evidence in cognate identification. Conference of the Canadian Society for Computational Studies of Intelligence, 2004. Springer, 44-59.

KUMAR, R., AI, H., BEUTH, J. & ROSÉ, C. Socially capable conversational tutors can be effective in collaborative learning situations. Intelligent tutoring systems, 2010. Springer, 156-164.

LATHAM, A., CROCKETT, K. & MCLEAN, D. 2014. An adaptation algorithm for an intelligent natural language tutoring system. *Computers & Education,* 71**,** 97-110.

LATHAM, A., CROCKETT, K., MCLEAN, D. & EDMONDS, B. 2010. Predicting Learning Styles in a Conversational Intelligent Tutoring System. *In:* LUO, X., SPANIOL, M., WANG, L., LI, Q., NEJDL, W. & ZHANG, W. (eds.) *Advances in Web-Based Learning – ICWL 2010: 9th International Conference, Shanghai, China, December 8-10, 2010. Proceedings.* Berlin, Heidelberg: Springer Berlin Heidelberg.

LATHAM, A., CROCKETT, K., MCLEAN, D. & EDMONDS, B. 2012. Adaptive tutoring in an intelligent conversational agent system. *Transactions on Computational Collective Intelligence VIII.* Springer.

LI, F. & JAGADISH, H. V. NaLIR: an interactive natural language interface for querying relational databases. Proceedings of the 2014 ACM SIGMOD international conference on Management of data, 2014. ACM, 709-712.

LI, Q., NAKANO, Y., OKAMOTO, M. & NISHIDA, T. Highlighting multimodal synchronization for embodied conversational agent. the 2nd International Conference on Information Technology for Application (ICITA 2004), 2004.

LI, Y., YANG, H. & JAGADISH, H. 2007. NaLIX: A generic natural language search environment for XML data. *ACM Transactions on database systems (TODS),* 32**,** 30.

LIAPIS, C. G. 2013. Human threading search engine. Google Patents.

LIN, D. An information-theoretic definition of similarity. ICML, 1998. Citeseer, 296-304.

LLOPIS, M. & FERRÁNDEZ, A. 2013. How to make a natural language interface to query databases accessible to everyone: An example. *Computer Standards & Interfaces,* 35**,** 470-481.

LOPEZ, V., FERNÁNDEZ, M., MOTTA, E. & STIELER, N. 2012. Poweraqua: Supporting users in querying and exploring the semantic web. *Semantic Web,* 3**,** 249-265.

LOPEZ, V., UNGER, C., CIMIANO, P. & MOTTA, E. 2013. Evaluating question answering over linked data. *Web Semantics: Science, Services and Agents on the World Wide Web,* 21**,** 3-13.

MA, C., OSHERENKO, A., PRENDINGER, H. & ISHIZUKA, M. A chat system based on emotion estimation from text and embodied conversational messengers. Active Media Technology, 2005.(AMT 2005). Proceedings of the 2005 International Conference on, 2005. IEEE, 546-548.

MAIRESSE, F., WALKER, M. A., MEHL, M. R. & MOORE, R. K. 2007. Using linguistic cues for the automatic recognition of personality in conversation and text. *Journal of artificial intelligence research,* 30**,** 457-500.

MARIETTO, M. D. G. B., DE AGUIAR, R. V., BARBOSA, G. D. O., BOTELHO, W. T., PIMENTEL, E., FRANÇA, R. D. S. & DA SILVA, V. L. 2013. Artificial intelligence markup language: A brief tutorial. *arXiv preprint arXiv:1307.3091*.

MARTINEZ, F. F., BLÁZQUEZ, J., FERREIROS, J., BARRA, R., MACIAS-GUARASA, J. & LUCAS-CUESTA, J. M. Evaluation of a spoken dialogue system for controlling a hifi audio system. Spoken Language Technology Workshop, 2008. SLT 2008. IEEE, 2008. IEEE, 137-140.

MICHIE, D. & SAMMUT, C. 2001. Infochat scripter's manual. *Technical Repoort, Convagent Ltd, Manchester, UK*.

MINOCK, M. 2010. C-Phrase: A system for building robust natural language interfaces to databases. *Data & Knowledge Engineering,* 69**,** 290-302.

MOLLER, S., ENGELBRECHT, K.-P., KUHNEL, C., WECHSUNG, I. & WEISS, B. A taxonomy of quality of service and quality of experience of multimodal human-machine interaction. Quality of Multimedia Experience, 2009. QoMEx 2009. International Workshop on, 2009. IEEE, 7-12.

NIELSEN, J. & MOLICH, R. Heuristic evaluation of user interfaces. Proceedings of the SIGCHI conference on Human factors in computing systems, 1990. ACM, 249-256.

NOLAN, S. A. & HEINZEN, T. 2011. *Statistics for the behavioral sciences*, Macmillan.

O'SHEA, K., BANDAR, Z. & CROCKETT, K. A semantic-based conversational agent framework. Internet Technology and Secured Transactions, 2009. ICITST 2009. International Conference for, 2009. IEEE, 1-8.

O'SHEA, J., BANDAR, Z. & CROCKETT, K. 2011. Systems Engineering and Conversational Agents. *In:* TOLK, A. & JAIN, L. C. (eds.) *Intelligence-Based Systems Engineering.* Berlin, Heidelberg: Springer Berlin Heidelberg.

O'SHEA, J., BANDAR, Z., CROCKETT, K. & MCLEAN, D. 2008. A comparative study of two short text semantic similarity measures. *Agent and Multi-Agent Systems: Technologies and Applications*, 172-181.

O'SHEA, K., BANDAR, Z. & CROCKETT, K. 2010. A conversational agent framework using semantic analysis. *International Journal of Intelligent Computing Research (IJICR),* 1.

O'SHEA, K., CROCKETT, K., BANDAR, Z. & O'SHEA, J. 2014. Erratum to: An approach to conversational agent design using semantic sentence similarity. *Applied Intelligence,* 40**,** 199-199.

OTT, N. 1992. Aspects of the automatic generation of SQL statements in a natural language query interface. *Information Systems,* 17**,** 147-159.

OWDA, M., BANDAR, Z. & CROCKETT, K. Conversation-based natural language interface to relational databases. Proceedings of the 2007 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology-Workshops, 2007. IEEE Computer Society, 363-367.

OWDA, M., BANDAR, Z. & CROCKETT, K. Information extraction for SQL Query generation in the conversation-based interfaces to relational databases (C-BIRD). KES International Symposium on Agent and Multi-Agent Systems: Technologies and Applications, 2011. Springer, 44-53.

PALLANT, J. 2013. *SPSS survival manual*, McGraw-Hill Education (UK).

PAZOS R, R., GONZÁLEZ B, J., AGUIRRE L, M., MARTÍNEZ F, J. & FRAIRE H, H. 2013. Natural language interfaces to databases: an analysis of the state of the art. *Recent Advances on Hybrid Intelligent Systems*, 463-480.

PAZOS R., R. A., GONZÁLEZ B., J. J., AGUIRRE L., M. A., MARTÍNEZ F., J. A. & FRAIRE H., H. J. 2013. Natural Language Interfaces to Databases: An Analysis of the State of the Art. *In:* CASTILLO, O., MELIN, P. & KACPRZYK, J. (eds.) *Recent Advances on Hybrid Intelligent Systems.* Berlin, Heidelberg: Springer Berlin Heidelberg.

PEREIRA, M.J. & COHEUR, L., 2013. Just. Chat-a platform for processing information to be used in chatbots.

POPESCU, A.-M., ARMANASU, A., ETZIONI, O., KO, D. & YATES, A. 2004. Modern natural language interfaces to databases: composing statistical parsing with semantic tractability. *Proceedings of the 20th international conference on Computational Linguistics.* Geneva, Switzerland: Association for Computational Linguistics.

RAMASUBRAMANIAN, P. & KANNAN, A. 2004. Temporal Event Matching Approach based Natural Language Query Processing in Temporal Databases. *International Journal of Information Technology,* 10.

RAO, G., AGARWAL, C., CHAUDHRY, S., KULKARNI, N. & PATIL, D. S. 2010. Natural language query processing using semantic grammar. *International journal on computer science and engineering,* 2**,** 219-223.

RAUSCHENBERGER, M., SCHREPP, M., COTA, M. P., OLSCHNER, S. & THOMASCHEWSKI, J. R. 2013. Efficient measurement of the user experience of interactive products. How to use the user experience questionnaire (ueq). example: spanish language version. *IJIMAI,* 2**,** 39-45.

REIS, P., MATIAS, J. & MAMEDE, N. 1997. Edite - A Natural Language Interface to Databases A new dimension for an old approach. *In:* TJOA, A. M. (ed.) *Information and Communication Technologies in Tourism 1997: Proceedings of the International Conference in Edinburgh, Scotland, 1997.* Vienna: Springer Vienna.

REVUELTA-MARTÍNEZ, A., RODRÍGUEZ, L., GARCÍA-VAREA, I. & MONTERO, F. 2013. Multimodal interaction for information retrieval using natural language. *Computer Standards & Interfaces,* 35**,** 428-441.

ROY, B. & GRAHAM, T. N. 2008. Methods for evaluating software architecture: A survey. *School of Computing TR,* 545**,** 82.

RUSSELL, S., NORVIG, P. & INTELLIGENCE, A. 1995. A modern approach. *Artificial Intelligence. Prentice-Hall, Egnlewood Cliffs,* 25**,** 27.

RZEPKA, R. & ARAKI, K. 2015. ELIZA Fifty Years Later: An Automatic Therapist Using Bottom-Up and Top-Down Approaches. *In:* VAN RYSEWYK, S. P. & PONTIER, M. (eds.) *Machine Medical Ethics.* Cham: Springer International Publishing.

SADEK, D. Design considerations on dialogue systems: From theory to technology-the case of artimis. ESCA Tutorial and Research Workshop (ETRW) on Interactive Dialogue in Multi-Modal Systems, 1999.

SAHA, D., FLORATOU, A., SANKARANARAYANAN, K., MINHAS, U. F., MITTAL, A. R. & ÖZCAN, F. 2016. ATHENA: an ontology-driven system for natural language querying over relational data stores. *Proceedings of the VLDB Endowment,* 9**,** 1209-1220.

SAMMUT, C. 2001. Managing context in a conversational agent. *Linkoping Electronic Articles in Computer & Information Science,* 3.

SHAALAN, K., MONEM, A. A., RAFEA, A. & BARAKA, H. Syntactic Generation of Arabic in Interlingua-based Machine Translation Framework. Third workshop on Computational Approaches to Arabic Script-based Languages (CAASL3), Machine Translation Summit XII, 2009. ACL.

SHABAZ, K., O'SHEA, J. D., CROCKETT, K. A. & LATHAM, A. Aneesah: A Conversational Natural Language Interface to Databases. Proceedings of the World Congress on Engineering, 2015.

SIJTSMA, W. & ZWEEKHORST, O. Comparison and review of commercial natural language interfaces. Proceedings, TWLT5, 1993. 43-58.

SILVERVARG, A. & JÖNSSON, A. Subjective and objective evaluation of conversational agents in learning environments for young teenagers. Proceedings of the 7th IJCAI Workshop on Knowledge and Reasoning in Practical Dialogue Systems, 2011.

SKANTZE, G. & HJALMARSSON, A. 2013. Towards incremental speech generation in conversational systems. *Computer Speech & Language,* 27**,** 243-262.

SRIRAMPUR, S., CHANDIBHAMAR, R., PALAKURTHI, A. & MAMIDI, R. Concepts identification of an NL query in NLIDB systems. Asian Language Processing (IALP), 2014 International Conference on, 2014. IEEE, 230-233.

STRATICA, N., KOSSEIM, L. & DESAI, B. C. 2005a. Using semantic templates for a natural language interface to the CINDI virtual library. *Data & Knowledge Engineering,* 55**,** 4-19.

STRATICA, N., KOSSEIM, L. & DESAI, B. C. 2005b. Using semantic templates for a natural language interface to the CINDI virtual library. *Data Knowl. Eng.,* 55**,** 4-19.

SUJATHA, B. & RAJU, S. V. 2016. Natural Language Query Parser using First Order Logic for Querying Relational Databases. *International Journal of Computer Applications,* 134**,** 43-48.

TANG, L. R. & MOONEY, R. J. 2001. Using Multiple Clause Constructors in Inductive Logic Programming for Semantic Parsing. *In:* DE RAEDT, L. & FLACH, P. (eds.) *Machine Learning: ECML 2001: 12th European Conference on Machine Learning Freiburg, Germany, September 5–7, 2001 Proceedings.* Berlin, Heidelberg: Springer Berlin Heidelberg.

TENNANT, H. R., ROSS, K. M., SAENZ, R. M., THOMPSON, C. W. & MILLER, J. R. Menu-based natural language understanding. Proceedings of the 21st annual meeting on Association for Computational Linguistics, 1983. Association for Computational Linguistics, 151-158.

THOMPSON, B. H. & THOMPSON, F. B. Introducing ask, a simple knowledgeable system. Proceedings of the first conference on Applied natural language processing, 1983. Association for Computational Linguistics, 17-24.

THOMPSON, B. H. & THOMPSON, F. B. 1985. Ask is transportable in half a dozen ways. *ACM Transactions on Information Systems (TOIS),* 3**,** 185-203.

TURING, A. M. 1950. Computing machinery and intelligence. *Mind,* 59**,** 433-460.

TURUNEN, M., HAKULINEN, J. & KAINULAINEN, A. Evaluation of a spoken dialogue system with usability tests and long-term pilot studies: similarities and differences. INTERSPEECH, 2006.

VAN SOLINGEN, R., BASILI, V., CALDIERA, G. & ROMBACH, D. 2002. Goal Question Metric (GQM) Approach, Encyclopedia of Software Engineering (Marciniak, JJ ed.), online version@ Wiley Interscience. John Wiley & Sons.

WALKER, M. A., LITMAN, D. J., KAMM, C. A. & ABELLA, A. PARADISE: A framework for evaluating spoken dialogue agents. Proceedings of the eighth conference on European chapter of the Association for Computational Linguistics, 1997. Association for Computational Linguistics, 271-280.

WALLACE, R. S. 2008. ALICE: Artificial Intelligence Foundation Inc. *Received from:* http://www. *alicebot. org*.

WALLACE, R. S. 2009. The Anatomy of A.L.I.C.E. *In:* EPSTEIN, R., ROBERTS, G. & BEBER, G. (eds.) *Parsing the Turing Test: Philosophical and Methodological Issues in the Quest for the Thinking Computer.* Dordrecht: Springer Netherlands.

WALTS, D. Natural Language Access To A Large Data Base: An Engineering Approach. IJCAI, 1975. 868-872.

WALTZ, D. L. 1978. An English language question answering system for a large relational database. *Communications of the ACM,* 21**,** 526-539.

WARREN, D. H. & PEREIRA, F. C. 1982. An efficient easily adaptable system for interpreting natural language queries. *Computational Linguistics,* 8**,** 110-122.

WEIZENBAUM, J. 1966. ELIZA—a computer program for the study of natural language communication between man and machine. *Communications of the ACM,* 9**,** 36-45.

WOODS, W. An Experimental Parsing System for Transition Network Grammars.< u> Natural Language Processing. Courant Computer Science Symposium, 1973.

WOODS, W. A. 1972. *An experimental parsing system for transition network grammars*, Computer Science Division, Bolt Beranek and Newman.

WU, Q., HU, Y. & WANG, Y. Research on Data Persistence Layer Based on Hibernate Framework. 2010 2nd International Workshop on Intelligent Systems and Applications, 22-23 May 2010 2010. 1-4.

YAGHMAZADEH, N., WANG, Y., DILLIG, I. & DILLIG, T. 2017. Type-and Content-Driven Synthesis of SQL Queries from Natural Language. *arXiv preprint arXiv:1702.01168*.

ZELLE, J. M. & MOONEY, R. J. Learning to parse database queries using inductive logic programming. Proceedings of the national conference on artificial intelligence, 1996. 1050-1055.

Web references

Carpenter, R. (2007). Jabberwacky – live chat bot. [Online] [Accessed on 22 May 2017] http://www.jabberwacky.com/

Convagent Ltd. (2005) Convagent. [Online] [Accessed on 24 May 2017] http://www.convagent.com/

http://sapinsider.wispubs.com/Assets/Blogs/2011/October/Visualizing-SAP-The-Connected-SAP-Universe?openlogin=true [Accessed 20 May. 2017].

Docs.oracle.com, (2017). [online] Available at: http://docs.oracle.com/cd/B28359_01/server.111/b28328/img/comsc010.gif [Accessed 20 May. 2017].

Hunspell Dictionary, (2017). [online] Available at: http://hunspell.github.io/ [Accessed 20 May. 2017].

Microsoft.com, (2017). Download Northwind and pubs Sample Databases for SQL Server 2000 from Official Microsoft Download Center. [online] Available at: http://www.microsoft.com/en-us/download/confirmation.aspx?id=23654 [Accessed 21 May. 2017].

Usabilitynet.org. (2017). *UsabilityNet: Definition of usability*. [online] Available at: http://www.usabilitynet.org/management/b_what.htm [Accessed 30 Dec. 2017].

Appendices

Appendix – A- Questionnaire for phase two evaluation prototype one

MANCHESTER METROPOLITAN UNIVERSITY

Evaluation Questionnaire – ANEESAH Natural Language Interface to Database (NLIDB)

Dear Participant,

I am a PhD researcher at Manchester Metropolitan University. As part of my research project, I am currently conducting evaluation of project development to date. The purpose of this evaluation is to examine and determine the usability, design and effectiveness of the ANEESAH Natural Language Interface to Databases that you have recently used.

This questionnaire will take only few minutes to complete and I would appreciate it if you would care to complete it. The questionnaire is divided into two parts. The first part comprises of scale based feedback against every question. An answer of 1 on the scale would be strongly/very negative and an answer of 5 would be very positive. The second part of the questionnaire includes questions which can be answered in “Yes” or “No”.

Please be assured that information you provide will be used only for academic purpose. We can ensure full confidentiality of your information and its safety. The information collected will be kept on records for one year and later destroyed.

Thank you very much for your participation.

Participant No: ☐

"Please rate the degree to which you agree with the following on a scale from 1 to 5 where 1 means strongly/very negative and 5 means very positive.*"*

			1	2	3	4	5	
1	Are you satisfied with interface design & level of dialog naturalness during conversation?	Very Low	☐	☐	☐	☐	☐	Very High
2	It was easy to understand and use the system.	strongly disagree	☐	☐	☐	☐	☐	strongly agree
3	I can effectively complete my work using this system	strongly disagree	☐	☐	☐	☐	☐	strongly agree
4	I am able to complete my work actively using this system.	strongly disagree	☐	☐	☐	☐	☐	strongly agree
5	I am able to complete my work quickly using this system.	strongly disagree	☐	☐	☐	☐	☐	strongly agree
6	I found this system to be useful	strongly disagree	☐	☐	☐	☐	☐	strongly agree
7	ANEESAH's level of understanding your requirement	Very Low	☐	☐	☐	☐	☐	Very High
8	I feel comfortable using this system	strongly disagree	☐	☐	☐	☐	☐	strongly agree
9	Are you satisfied with ANEESAH's dialog responses?	Very Low	☐	☐	☐	☐	☐	Very High
10	Are you satisfied with information produced from domain Database?	strongly disagree	☐	☐	☐	☐	☐	strongly agree

11. Would you use these kind of systems in the future?

YES ☐ NO ☐

12. Would you use ANEESAH system instead of taking help from a SQL expert?

YES ☐ NO ☐

Any further comments you may have:

____

____

Thanks!

Test Scenarios for the evaluation of *ANEESAH (NLIDB).*

Total Scenarios: 7

Scenario – 1

You are a sales advisor at Manchester Computer Store Ltd. You don't know what products your company sell? Ask the system to assist you with this information?

Scenario - 2

You a new sales advisor at Manchester Computer Store Ltd. A customer has just asked you about what countries your company is operating in?
Take help from the system to see which countries your company is operating in.

Scenario - 3

You are required to provide a list of customer names, emails and contact numbers from Barcelona for marketing purpose. Ask the system to give you these customer details?

Scenario - 4

As a sales manager at the Store you need to find total turnover generated from sales in Japan? Take help from the system to give you this information.

Scenario - 5

You are assigned to compare the total profit made from different countries. Ask system to give you these figures?

Scenario - 6

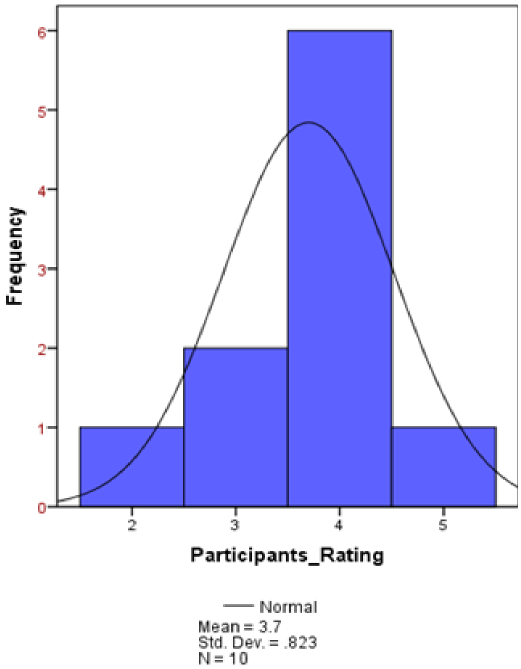
As part of product analysis you are required to find company's top five bestselling products in France during the year 1999? Ask the system to give you this information.

Scenario - 7

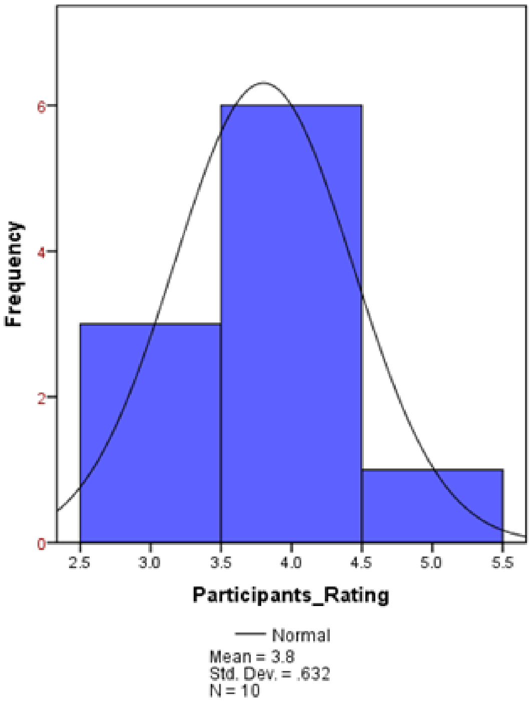
As sales manager you are required to find out the total sold quantity of mouse pads in Asia.

Ask the system to give this figure?

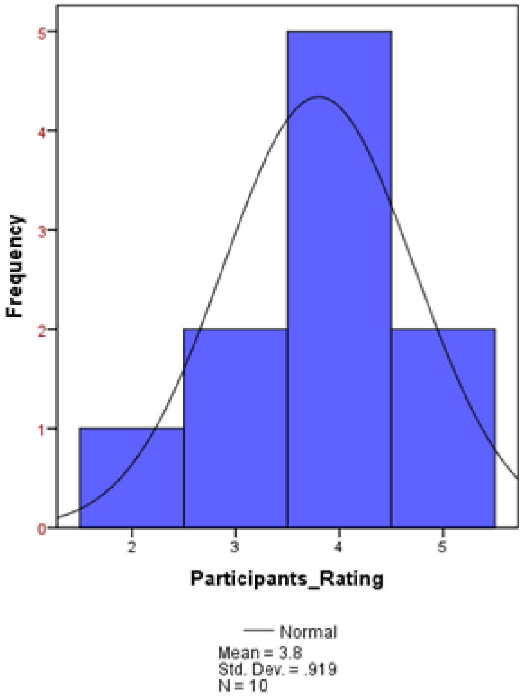
Appendix – C – Phase one evaluation data histograms



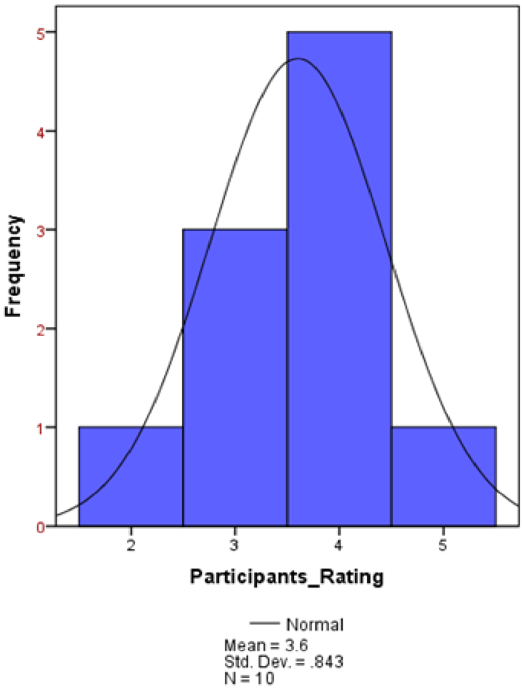
Interface and Level of dialogue naturalness during conversation. (Group-A)



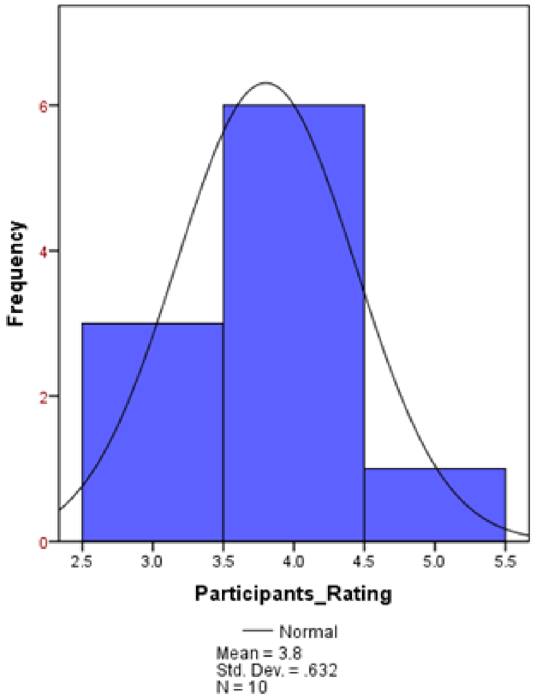
Interface and Level of dialogue naturalness during conversation. (Group-B)

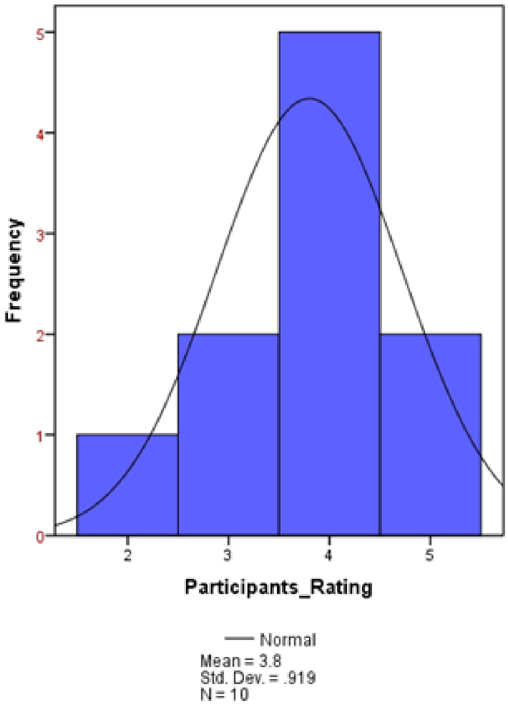


It was easy to understand and use the system.(Group-A)



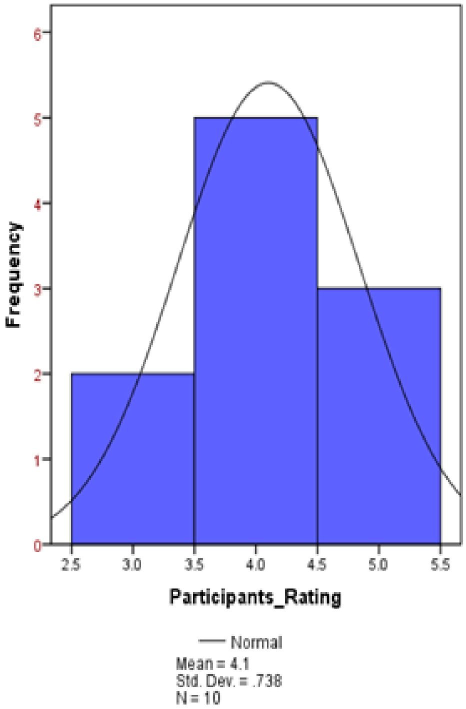
It was easy to understand and use the system.(Group-B)

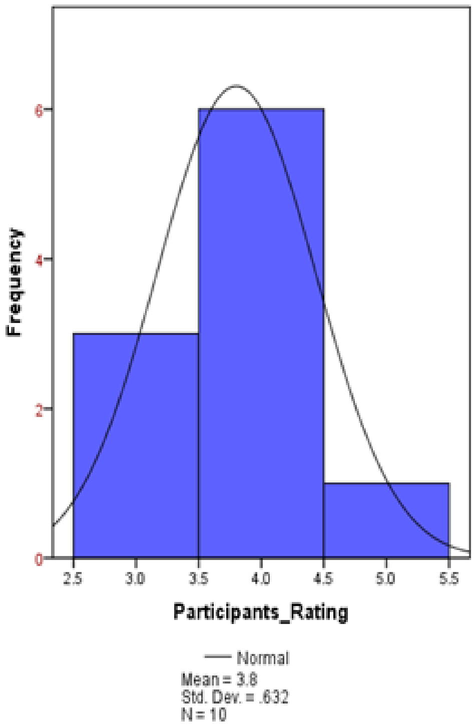




I can effectively complete my work using this system. (Group-A)

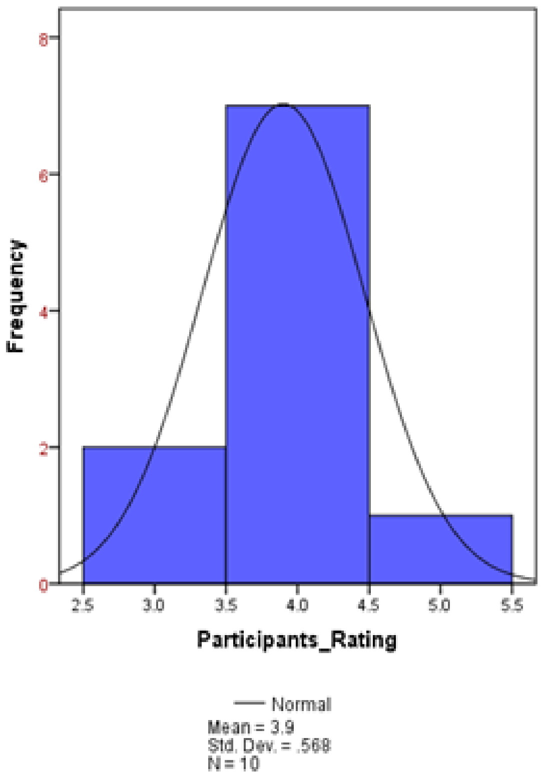
I can effectively complete my work using this system.(Group-B)



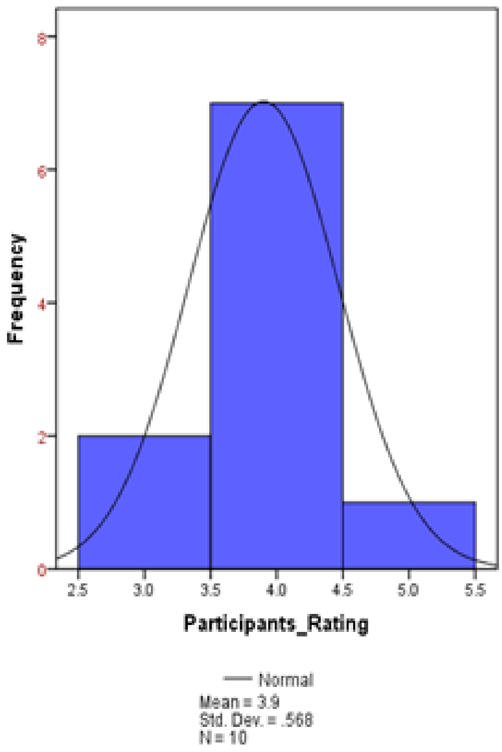


I am able to complete my work actively using this system. (Group-A)

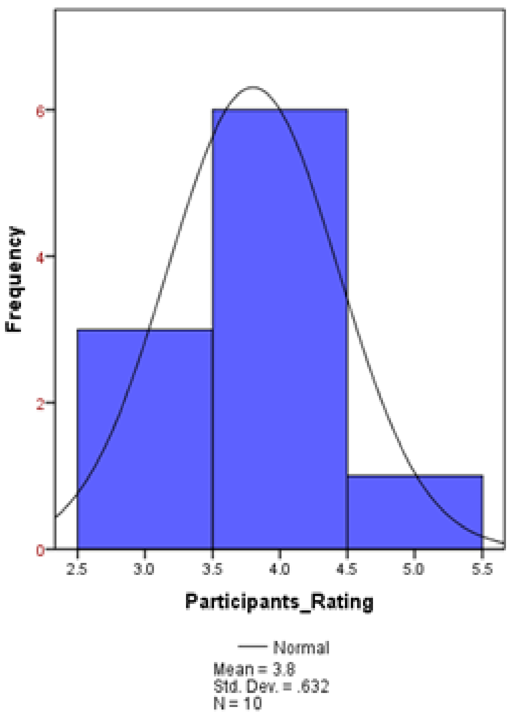
I am able to complete my work actively using this system. (Group-B)



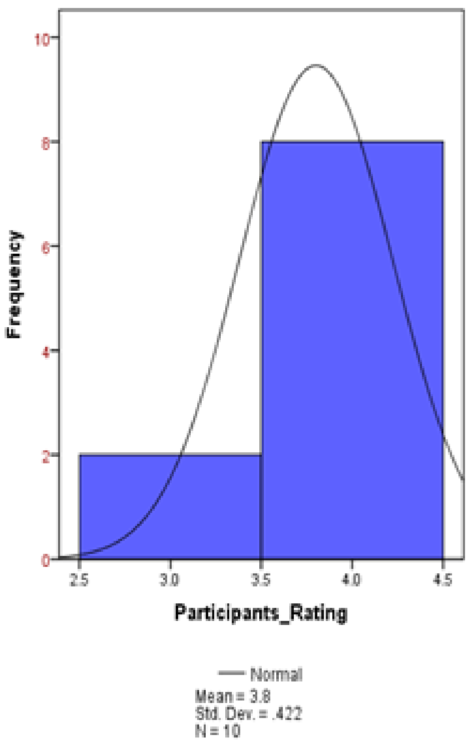
I am able to complete my work quickly using this system. (Group-A)



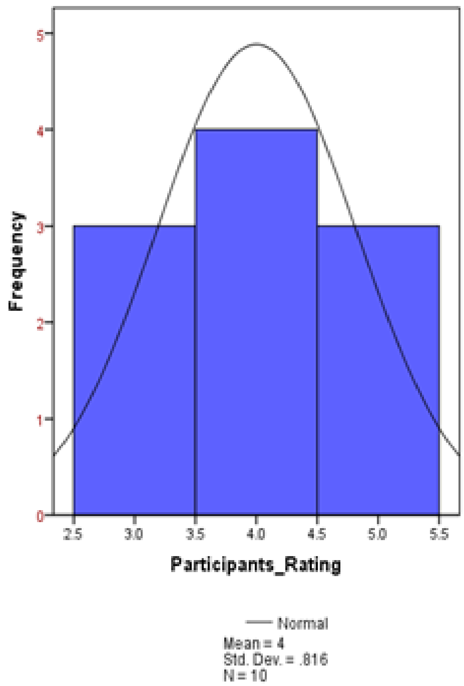
I am able to complete my work quickly using this system. (Group-B)



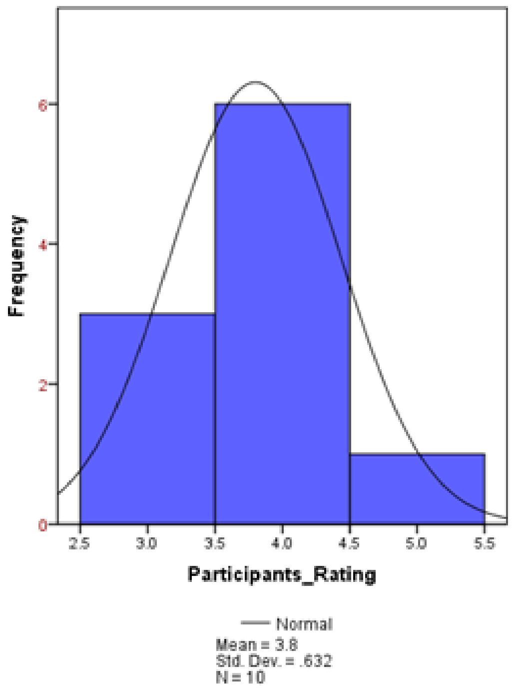
I found this system to be useful. (Group-A)



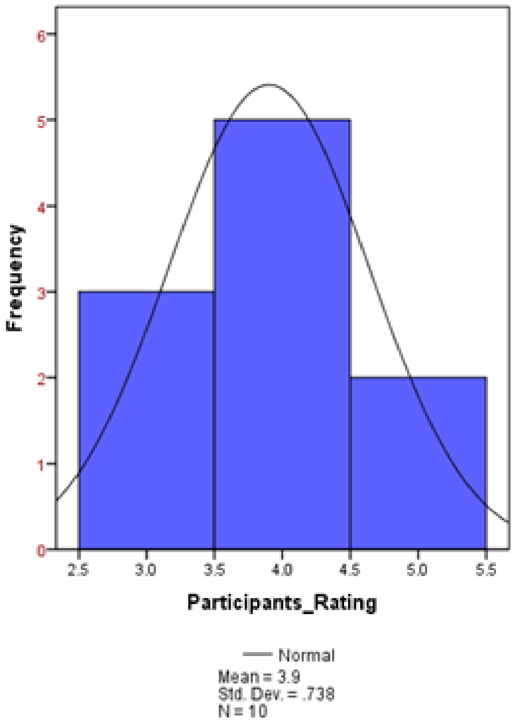
I found this system to be useful. (Group-B)



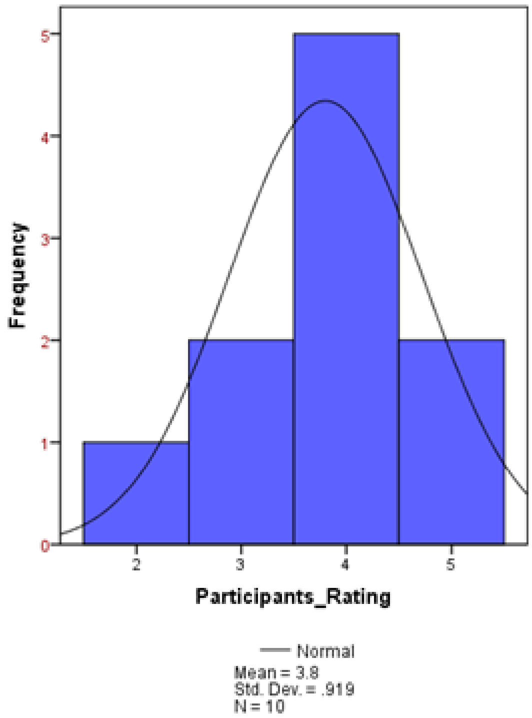
ANEESAH's level of understanding your requirement. (Group-A)



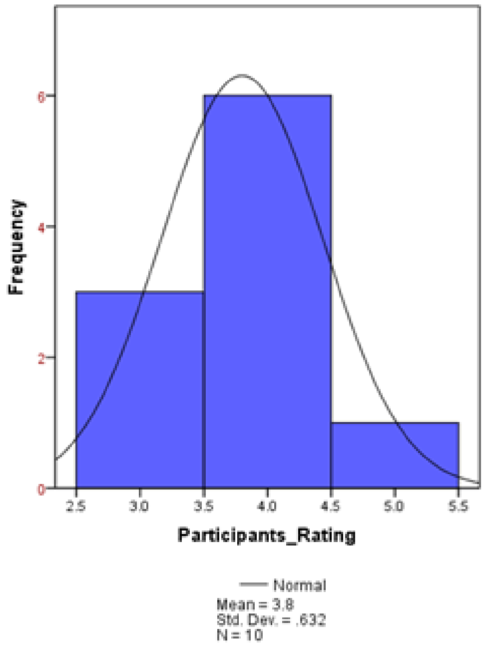
ANEESAH's level of understanding your requirement. (Group-B)



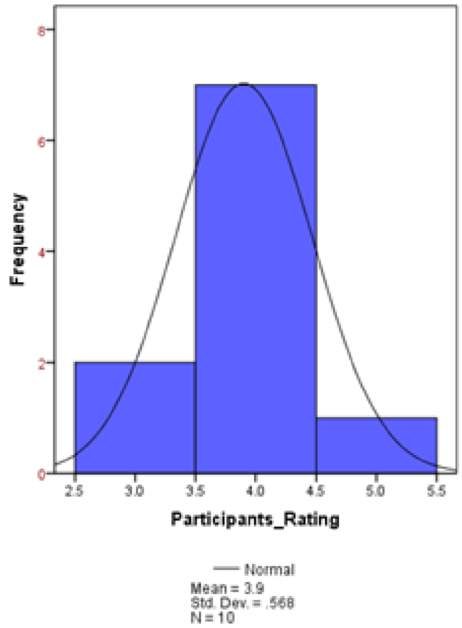
I feel comfortable using this system. (Group-A)



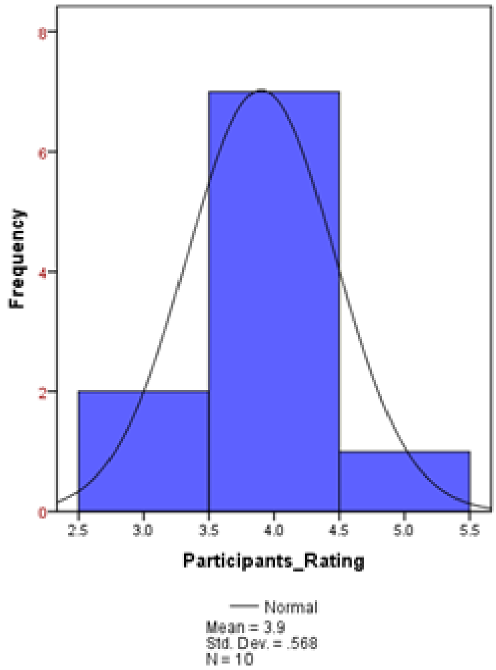
I feel comfortable using this system. (Group-B)



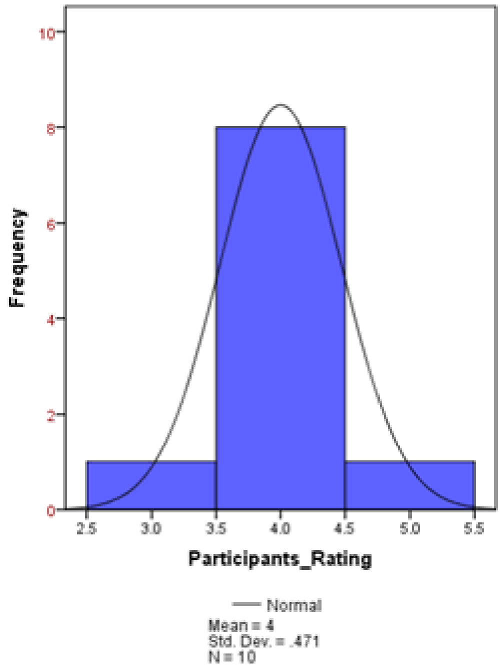
Are you satisfied with ANEESAH's dialogue responses? (Group-A)



Are you satisfied with ANEESAH's dialogue responses? (Group-B)



Are you satisfied with information produced from domain Database? (Group-A)



Are you satisfied with information produced from domain Database? (Group-B)

Appendix – D- Questionnaire for phase two evaluation

MANCHESTER METROPOLITAN UNIVERSITY

Evaluation Questionnaire – ANEESAH Natural Language Interface to Database (NLIDB)

Dear Participant,

I am a PhD researcher at Manchester Metropolitan University. As part of my research project, I am conducting evaluation of a developed prototype system. The purpose of this evaluation is to examine and determine the usability, design and effectiveness of this developed system (ANEESAH Natural Language Interface to Databases).

This questionnaire will take only few minutes to complete and I would appreciate it if you would care to complete it. The questionnaire is divided into two parts. The first part comprises of scale based feedback against every question. An answer of 1 on the scale would be strongly/very negative and an answer of 5 would be very positive. The second part of the questionnaire includes questions which can be answered in “Yes” or “No”.

Please be assured that information you provide will be used only for academic purpose. We can ensure full confidentiality of your information and its safety. The information collected will be kept on records for one year and later destroyed.

Thank you very much for your participation.

Participant No: ☐

☐ I write my own database queries.

☐ I create my own reports based on queries written by other people.

☐ I use database reports and queries developed by other people/applications.

☐ To the best of my knowledge I have never used a database

"Please rate the degree to which you agree with the following on a scale from 1 to 5 where 1 means strongly/very negative and 5 means very positive.*"*

			1	2	3	4	5	
1	I found this system to be useful?	strongly disagree	☐	☐	☐	☐	☐	strongly agree
2	I am able to complete my work actively and quickly using this system?	strongly disagree	☐	☐	☐	☐	☐	strongly agree
3	Overall, how would you rate your satisfaction level about using the system?	strongly disagree	☐	☐	☐	☐	☐	strongly agree
4	I think that I can effectively complete my work using this system.	strongly disagree	☐	☐	☐	☐	☐	strongly agree
5	To what extent do you agree with the system's ability to entertain/handle requests?	strongly disagree	☐	☐	☐	☐	☐	strongly agree
6	To what extent do you agree with the system s ability to refine information?	strongly disagree	☐	☐	☐	☐	☐	strongly agree
7	I think it was easy to understand and use the system.	strongly disagree	☐	☐	☐	☐	☐	strongly agree
8	I am confident about the system's level of understanding my inputs.	strongly disagree	☐	☐	☐	☐	☐	strongly agree
9	I found this system to be user friendly.	strongly disagree	☐	☐	☐	☐	☐	strongly agree
10	I am satisfied with the results produced from database and information refinement as part of completing the scenarios.	strongly disagree	☐	☐	☐	☐	☐	strongly agree
11	I am satisfied with the overall system's responses.	strongly disagree	☐	☐	☐	☐	☐	strongly agree
12	The system's dialogue during the conversation was natural.	strongly disagree	☐	☐	☐	☐	☐	strongly agree

13. Would you use a similar system again in the future?

YES ☐ NO ☐

14. Would you use ANEESAH system instead of taking help from a Database expert??

YES ☐ NO ☐

Any further comments you may have:

Thanks!

Test Scenarios for the evaluation of:

ANEESAH Natural Language Interface to Database (NLIDB)

Scenarios mentioned in the following sections have been prepared for test users to retrieve appropriate information from Aneesah. The benchmark answers against below scenarios will be compared and evaluated with system produced answers during the testing phase.

Total Test Scenarios: 7

Scenario – 1

You don't know what products your company sells?

i. Get the system to list the products available from the company?

***(Note:** Make note of any one product name, as this will be used later)*

Product Name……………………………………………………………………

Scenario - 2

You don't know what products your company sells?

i. Ask the system list the names of regions company trades in?

***(Note:** Make note of one region name, as this will be used later)*

Region Name……………………………………………………………………

Scenario - 3

You need to determine which sales channel has been most successful for the company in selling the product that you note in TEST SCENARIO - 1.

i. Get help from the system to show overall sales the for product that you noted above?

ii. Now get help from the system to include sales channels into this.

Make a note of the best sales channel

Best Sales Channel Name ……………………………………………………

Scenario - 4

There is a requirement to find most profitable month for the company in 1998 (fiscal year) for the region that you noted in TEST SCENARIO - 2.

i. Ask the system to show overall profit made by the company for above year for the region that you noted in TEST SCENARIO - 2.

ii. Get the system to add monthly breakdown in the results.

Make a note of month with highest monthly profit

Month Name…………………………………………………

Scenario - 5

You are asked to find the following:

i. Get help from the system to see the average sale for either “Mouse Pad” or “Deluxe Mouse” for the October, 1998. (fiscal period)

ii. Now ask the system to add product that you noted in TEST SCENARIO – 1.

Make a note of highest average sales product name

Product Name…………………………………………………….

Scenario - 6

You are asked to perform the following:

i. Get help from the system to discover top five products from Spain and Italy.

ii. Instruct the system to remove one country of your desire from shown results.

Make a note of the country names that you can see on screen

Country name…………………………………………………….

Scenario - 7

You are asked to perform the following:

i. Get help from the system to count total orders quantity for ‘y box’ or ‘Laptop Carrying Case’ (product) received by the company in calendar year 1998.

ii. Get help from the system to replace y box with the product that you noted in TEST SCENARIO – 1.

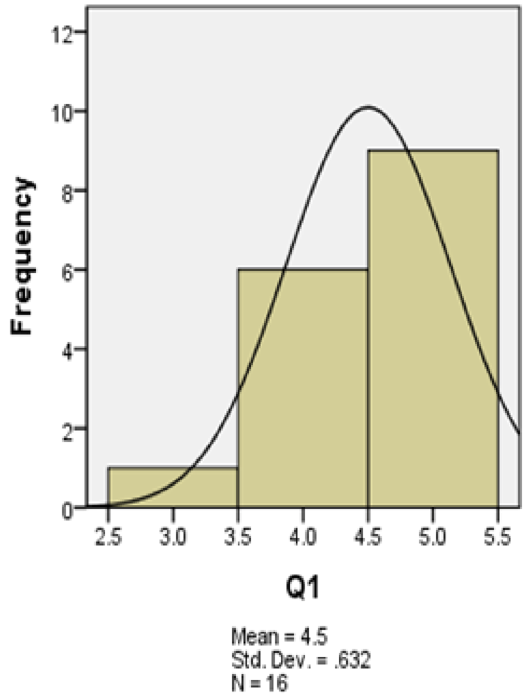
iii. Now ask to add region that you noted in TEST SCENARIO – 2

Make a note of the total orders quantity

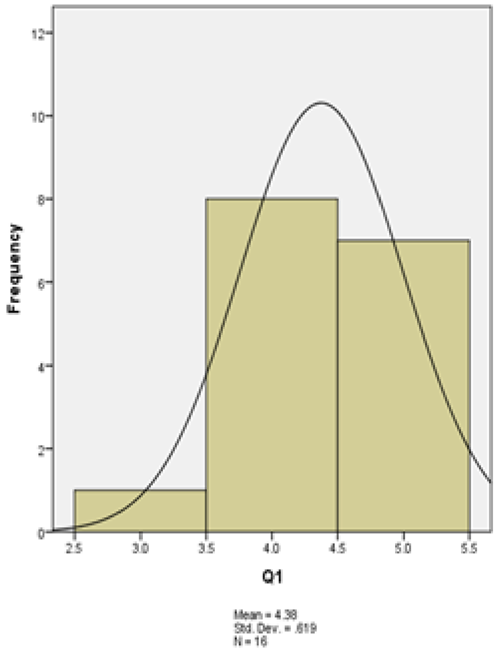
Total Order Quantity…………………………………………………….

Appendix – F – Phase two evaluation data histograms

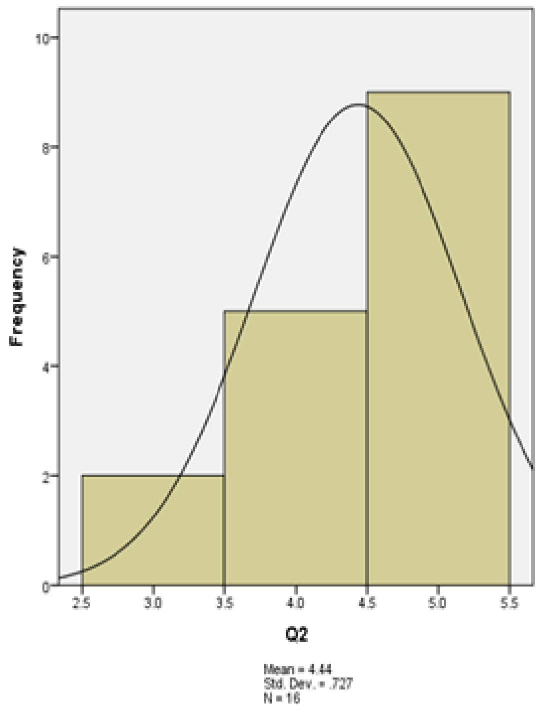
Group-A Participants | **Group-B Participants**



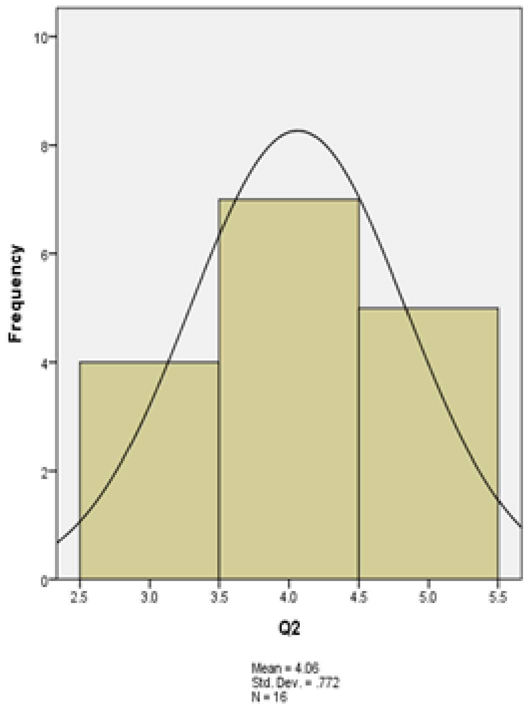
I found this system to be useful?



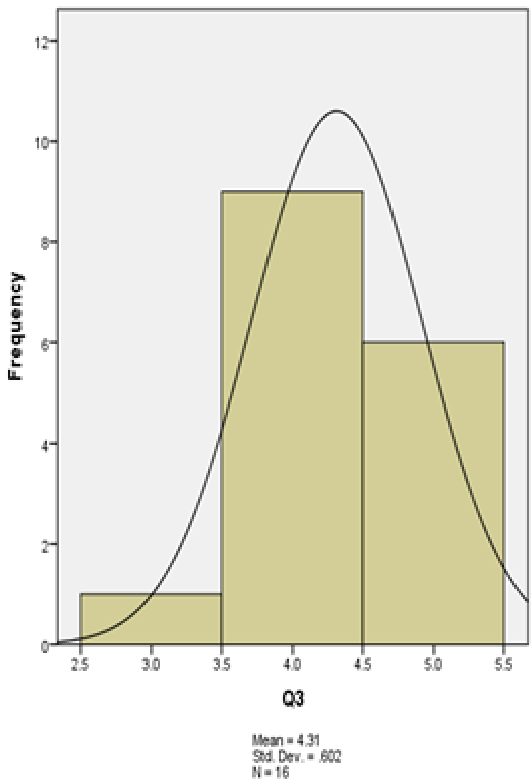
I found this system to be useful?

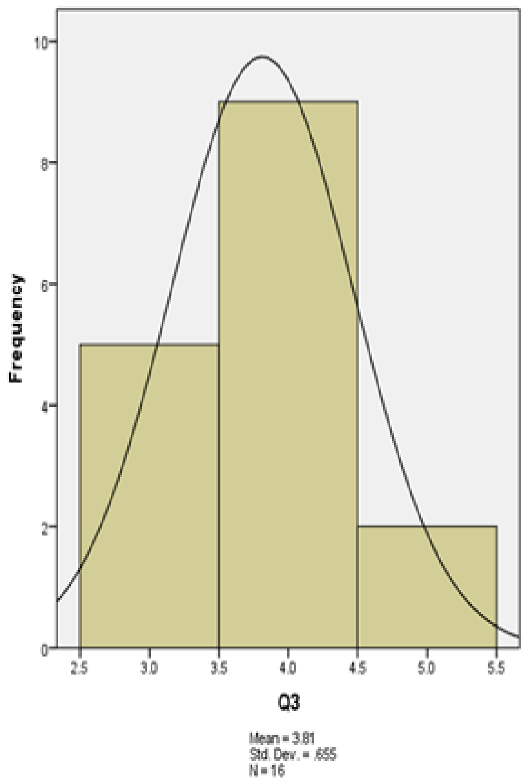


I am able to complete my work actively and quickly using this system?



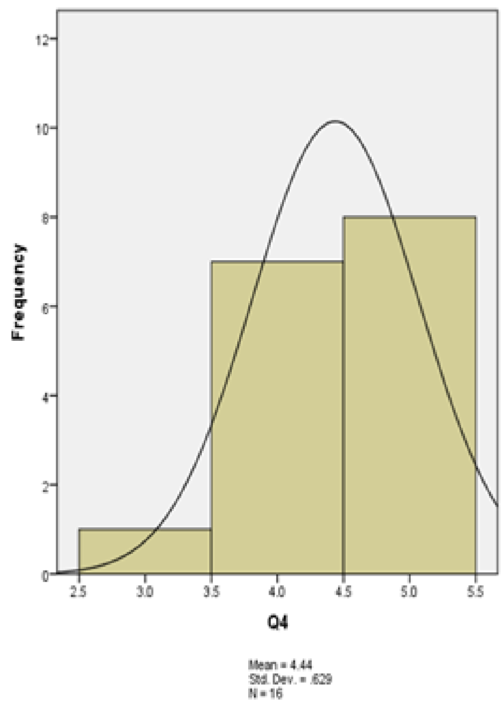
I am able to complete my work actively and quickly using this system?

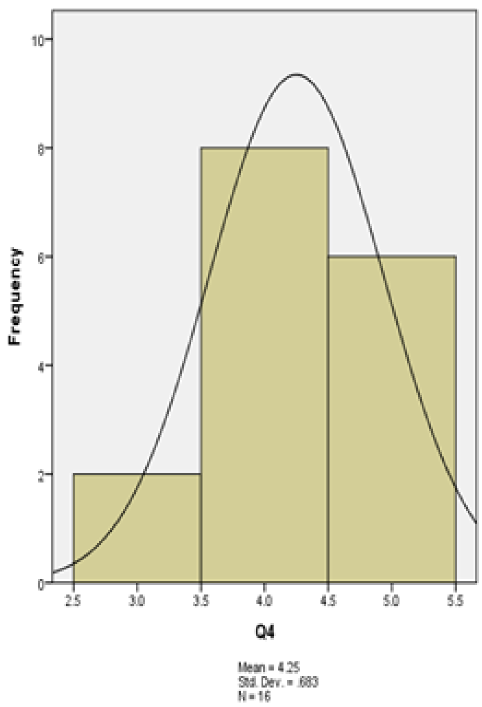




Overall, how would you rate your satisfaction level about using the system?

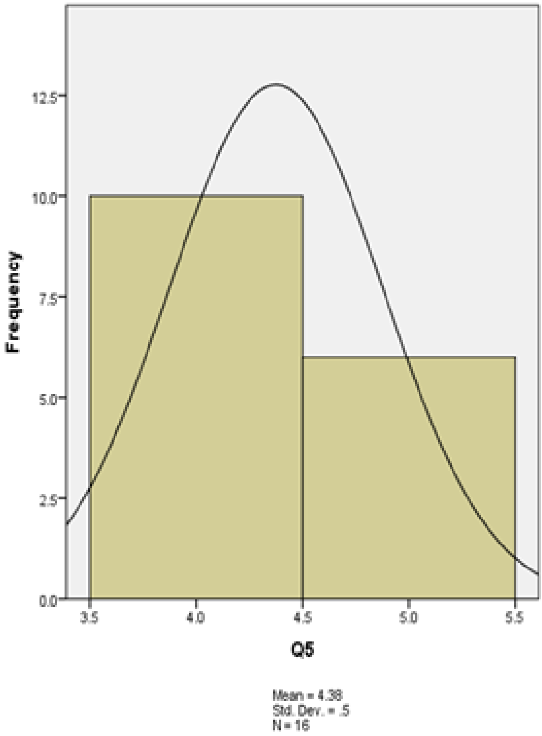
Overall, how would you rate your satisfaction level about using the system?

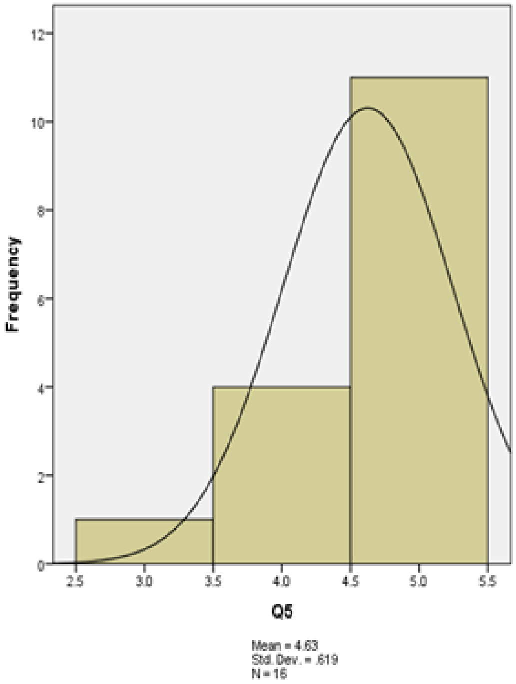




I think that I can effectively complete my work using this system.

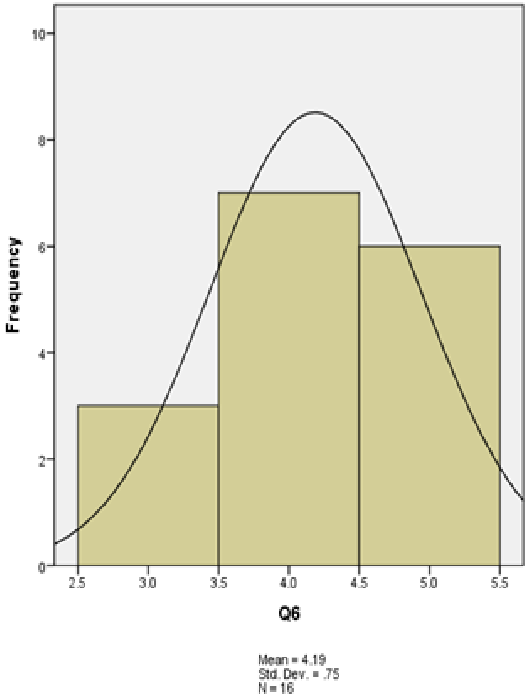
I think that I can effectively complete my work using this system.

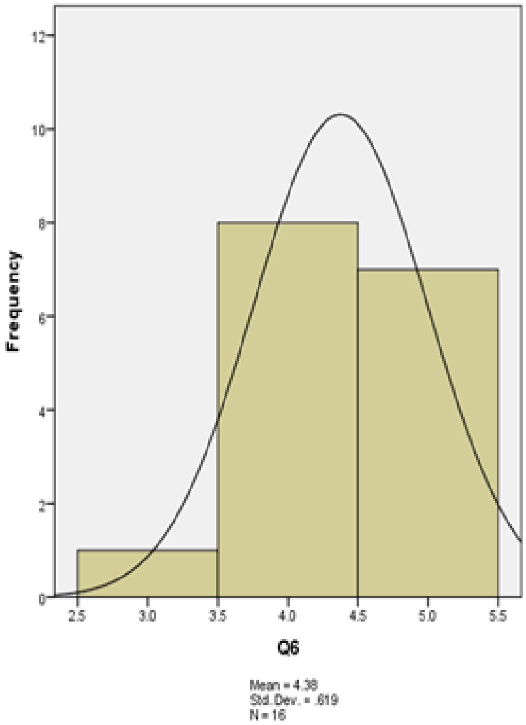




To what extent do you agree with the system's ability to entertain/handle requests?

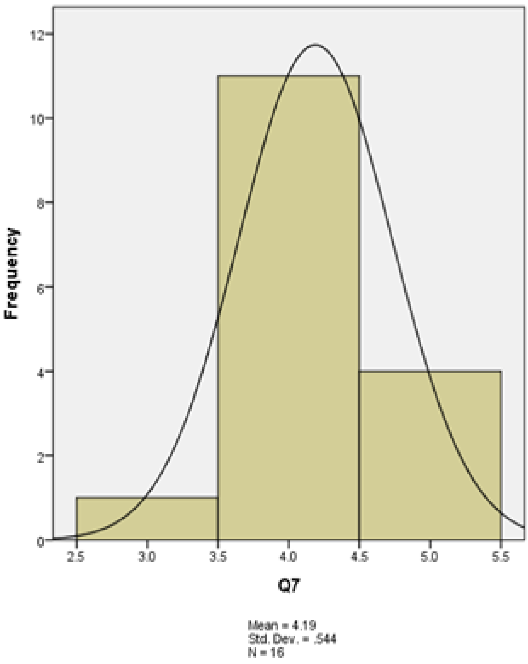
To what extent do you agree with the system's ability to entertain/handle requests?

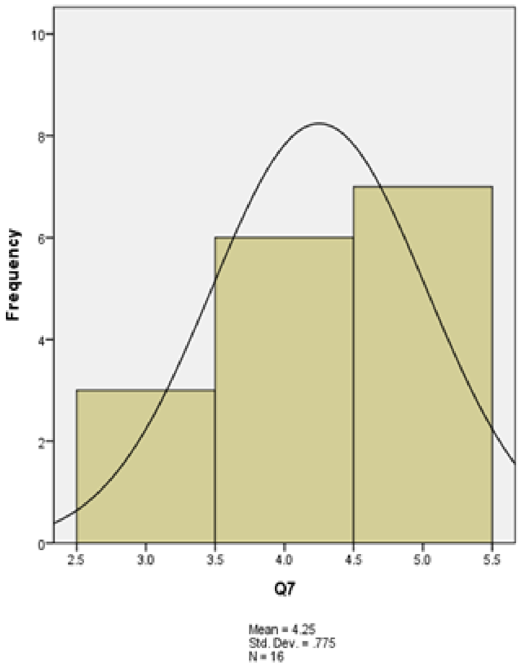




To what extent do you agree with the system s ability to refine information?

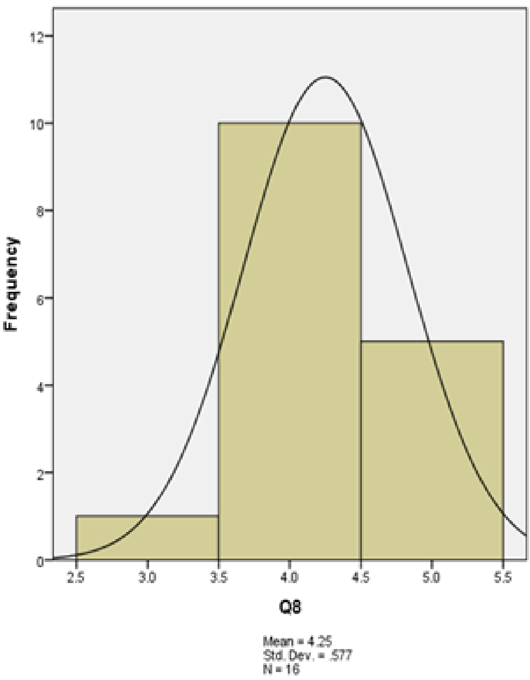
To what extent do you agree with the system s ability to refine information?

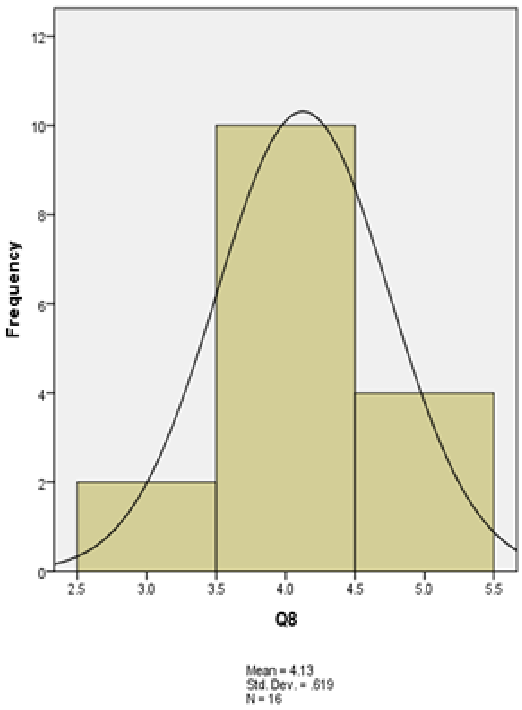




I think it was easy to understand and use the system.

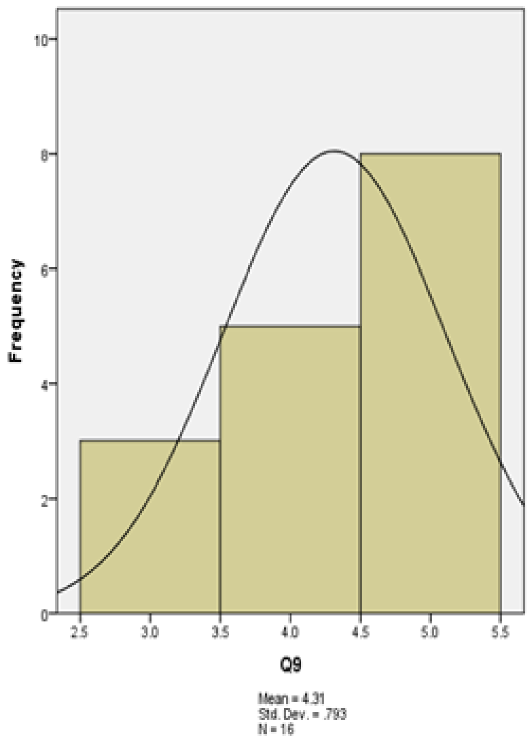
I think it was easy to understand and use the system.

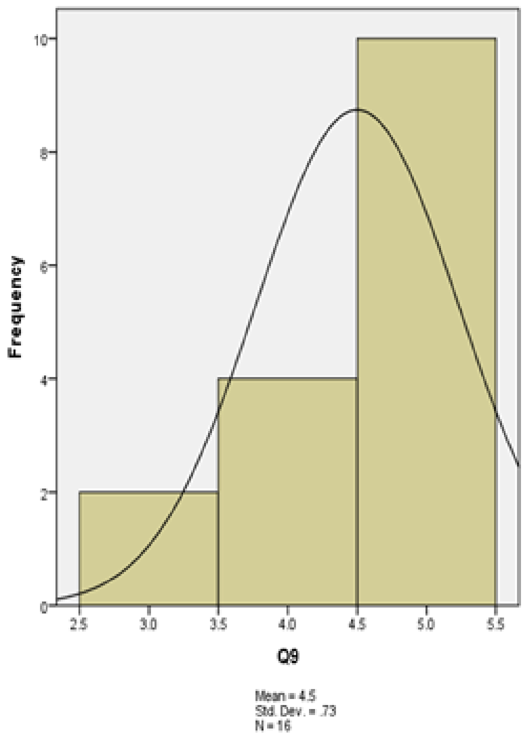




I am confident about the system's level of understanding my inputs.

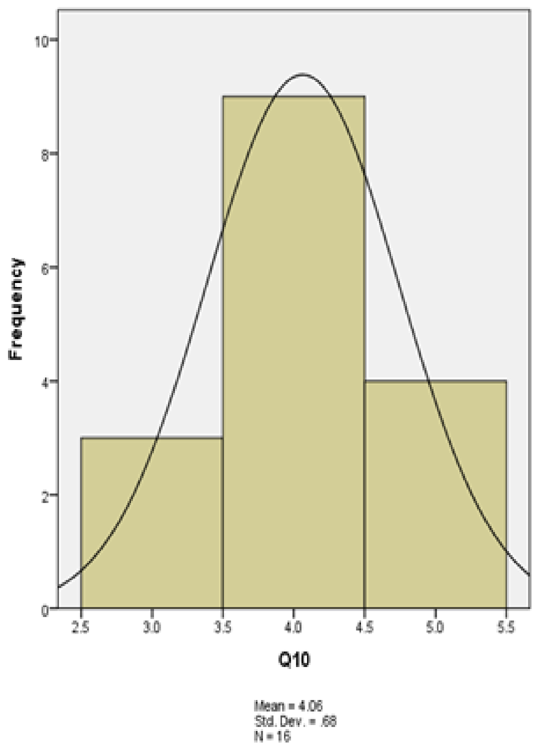
I am confident about the system's level of understanding my inputs.

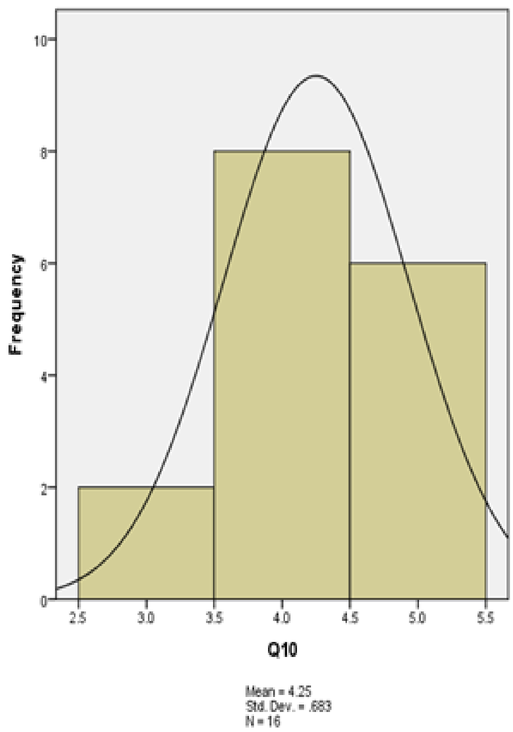




I found this system to be user friendly.

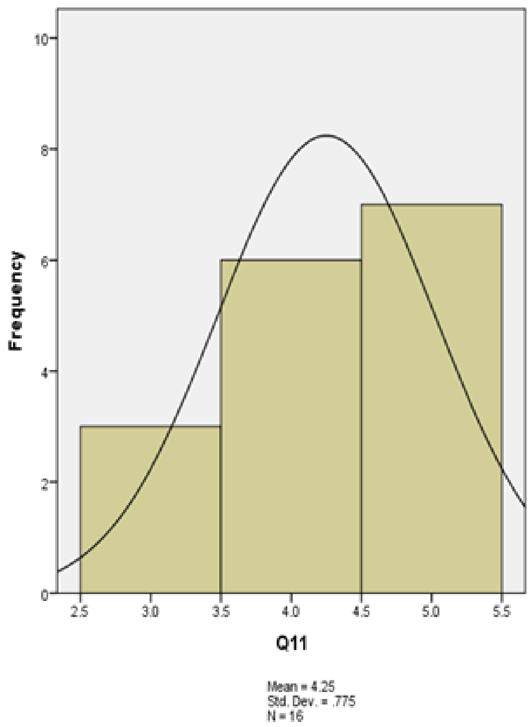
I found this system to be user friendly.

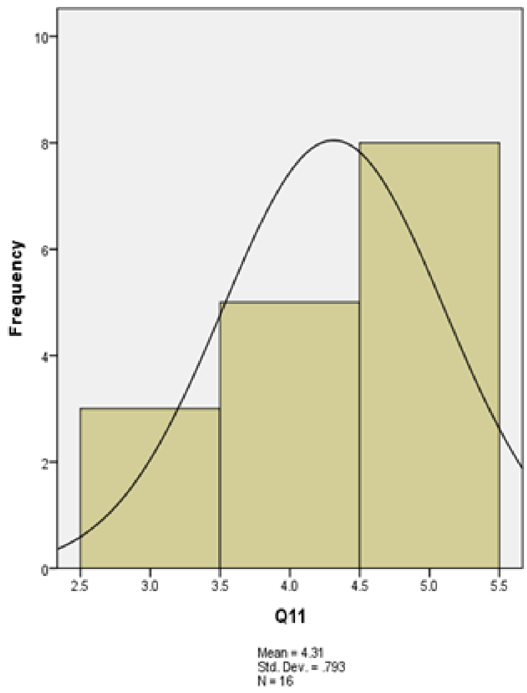




I am satisfied with the results produced from database and information refinement as part of completing the scenarios.

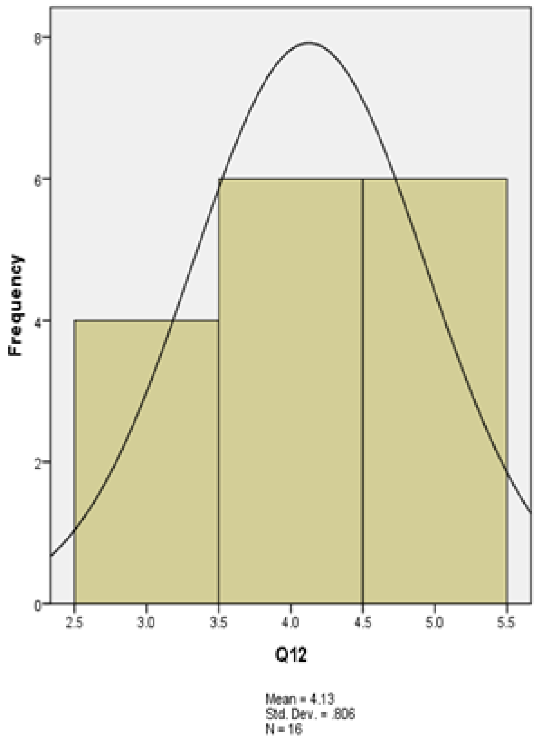
I am satisfied with the results produced from database and information refinement as part of completing the scenarios.

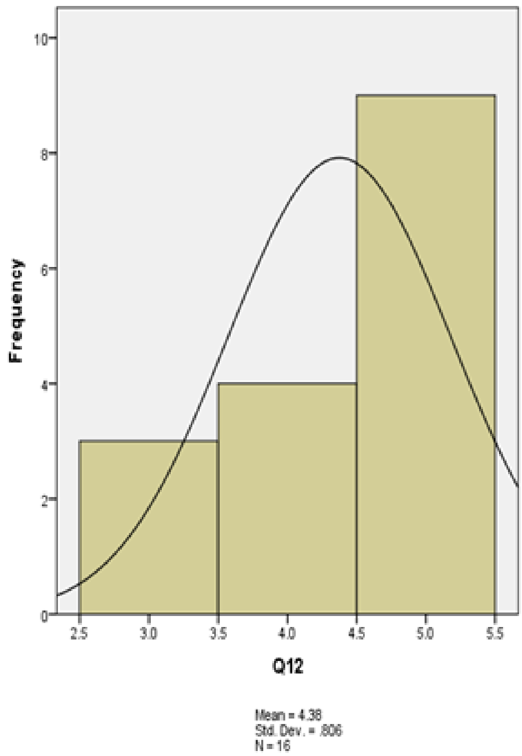




I am satisfied with the overall system's responses.

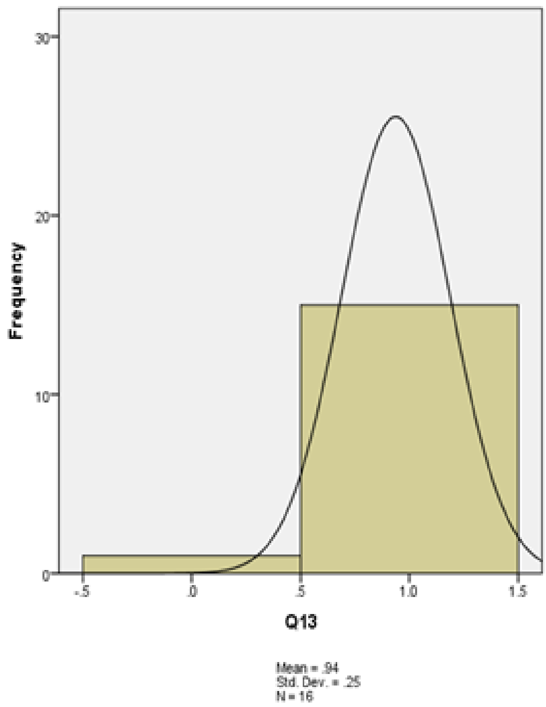
I am satisfied with the overall system's responses.

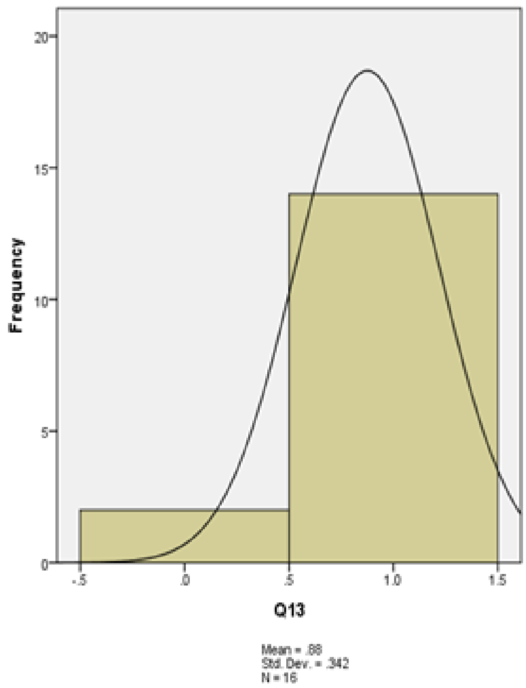




The system's dialogue during the conversation was natural.

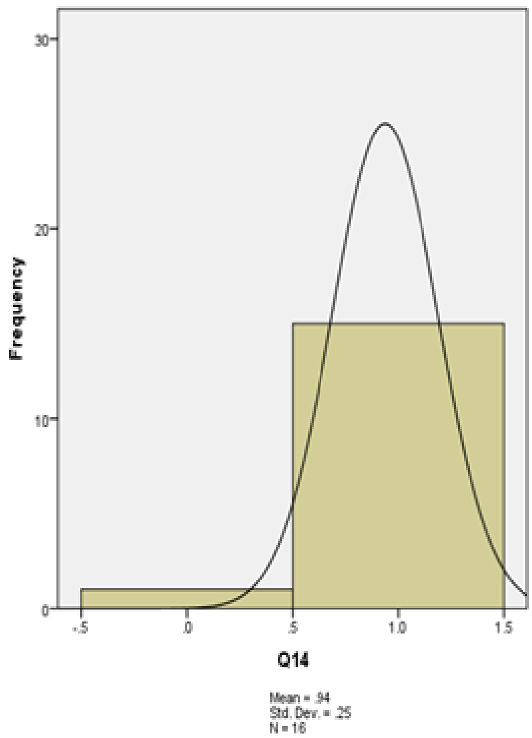
The system's dialogue during the conversation was natural.

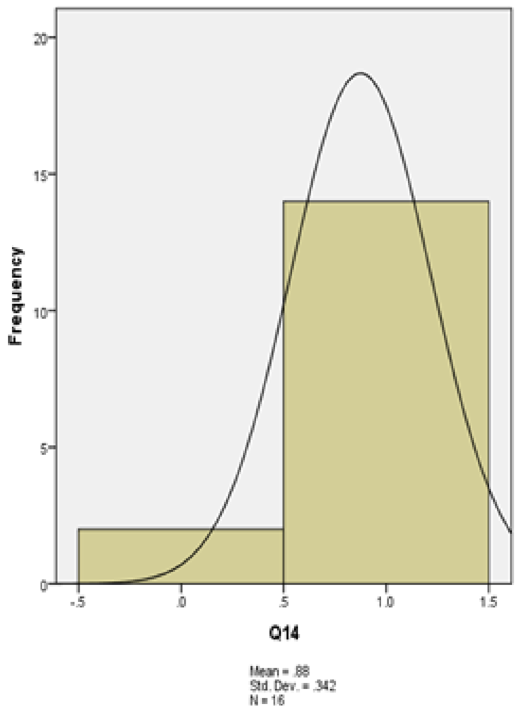




Would you use a similar system again in the future?

Would you use a similar system again in the future?





Would you use ANEESAH system instead of taking help from a Database expert?

Would you use ANEESAH system instead of taking help from a Database expert?

Appendix – G –ANEESAH's Log File

Below are few dialogue examples from ANEESAH's logfile illustrated after formatting.

Example 1: ANEESAH greeted the user (Rebecca) followed by example response and information refinement example after confirmation from the user.

LOG_INDEX	LOG_SESSION_ID	LOG_DATE	LOG_TIME	LOG_APPSTART	LOG_ACTOR	LOG_ACT_IOPUT	LOG_MATCH_HEADER	LOG_ITEM_	LOG_DESCRIPTION	LOG_CONV_INDEX	LOG_MAIN_CONTEXT	LOG_SCENARIO_NO	LOG_MATCH_STENGTH	LOG_PARTICIPANT_NO	LOG_SQL_KNOWLEDGE
1	38b5e87e-b0a0-44bc-9ff0-c75d083fd840	28/10/2016	16:45	28/10/2016 16:45	Aneesah:	Aneesah: Welcome to Aneesah NLIDB. My name is Aneesah, What is your name?				12897		1		1	3
2	38b5e87e-b0a0-44bc-9ff0-c75d083fd840	28/10/2016	16:45	28/10/2016 16:45	User:	User: my name is rebecca				12898		1		1	3
3	38b5e87e-b0a0-44bc-9ff0-c75d083fd840	28/10/2016	16:45	28/10/2016 16:45	Aneesah:	Aneesah: Nice to meet you Rebecca, What would you like to know about the sales?				12899		1		1	3
4	38b5e87e-b0a0-44bc-9ff0-c75d083fd840	28/10/2016	16:45	28/10/2016 16:45	Rebecca:	Rebecca: can you tell me what products you sell?				12900		1		1	3
7	38b5e87e-b0a0-44bc-9ff0-c75d083fd840	28/10/2016	16:45	28/10/2016 16:45	Aneesah:	Aneesah: I have discovered information reflected in 'ResultsView' window relevant to your input. Is this what you were looking to find out from the database?	SQLQUERY	SELECT PRODUCTS.PROD_NAME FROM PRODUCTS	Executed query response.	12905	DOMAIN	1	1	1	3
8	38b5e87e-b0a0-44bc-9ff0-c75d083fd840	28/10/2016	16:45	28/10/2016 16:45						12906		1		1	3
9	38b5e87e-b0a0-44bc-	28/10/2016	16:46	28/10/2016 16:46	Rebecca:	Rebecca: can you tell me in what region you offer these products?				12907		2		1	3

	9ff0-c75d083fd840														
10	38b5e87e-b0a0-44bc-9ff0-c75d083fd840	28/10/2016	16:46	28/10/2016 16:46	Aneesah:	Aneesah: Is this in connection with your previous request?				12908		2		1	3
11	38b5e87e-b0a0-44bc-9ff0-c75d083fd840	28/10/2016	16:46	28/10/2016 16:46	Rebecca:	Rebecca: no				12909		2		1	3
15	38b5e87e-b0a0-44bc-9ff0-c75d083fd840	28/10/2016	16:46	28/10/2016 16:46	Aneesah:	Aneesah: I have discovered 10000 records in response to your input. Would you like to limit these records. You can say like top 100 etc?				12914		2		1	3
16	38b5e87e-b0a0-44bc-9ff0-c75d083fd840	28/10/2016	16:46	28/10/2016 16:46	Rebecca:	Rebecca: can you tell me what regions you have in the database?				12915		2		1	3
21	38b5e87e-b0a0-44bc-9ff0-c75d083fd840	28/10/2016	16:46	28/10/2016 16:46	Aneesah:	Aneesah: I have discovered information reflected in 'ResultsView' window relevant to your input. Is this what you were looking to find out from the database?	SQLQUERY	SELECT COUNTRIES.COUNTRY_REGION FROM COUNTRIES	Executed query response.	12921	DOMAIN	2	0.9	1	3
22	38b5e87e-b0a0-44bc-9ff0-c75d083fd840	28/10/2016	16:46	28/10/2016 16:46						12922		2		1	3
23	38b5e87e-b0a0-44bc-9ff0-c75d083fd840	28/10/2016	16:47	28/10/2016 16:47	Rebecca:	Rebecca: what are the total sales results for y box				12923		3		1	3
24	38b5e87e-b0a0-44bc-9ff0-c75d083fd840	28/10/2016	16:47	28/10/2016 16:47	Aneesah:	Aneesah: Is this in connection with your previous request?				12924		3		1	3
25	38b5e87e-b0a0-44bc-9ff0-c75d083fd840	28/10/2016	16:47	28/10/2016 16:47	Rebecca:	Rebecca: no				12925		3		1	3

29	38b5e87e-b0a0-44bc-9ff0-c75d083fd840	28/10/2016	16:47	28/10/2016 16:47	Aneesah:	Aneesah: I have discovered information reflected in 'ResultsView' window relevant to your input. Is this what you were looking to find out from the database?	SQLQUERY	SELECT * FROM (SELECT PRODUCTS.PROD_NAME, SUM(PROFITS.AMOUNT_SOLD) FROM PRODUCTS INNER JOIN PROFITS ON PRODUCTS.PROD_ID = PROFITS.PROD_ID WHERE PROD_NAME = 'Y Box' GROUP BY PRODUCTS.PROD_NAME)	Executed query response.	12931		3	1	1	3
30	38b5e87e-b0a0-44bc-9ff0-c75d083fd840	28/10/2016	16:47	28/10/2016 16:47						12932	DOMAIN	3		1	3
31	38b5e87e-b0a0-44bc-9ff0-c75d083fd840	28/10/2016	16:48	28/10/2016 16:48	Rebecca:	Rebecca: in what way have the company made these sales				12933		3		1	3
32	38b5e87e-b0a0-44bc-9ff0-c75d083fd840	28/10/2016	16:48	28/10/2016 16:48	Aneesah:	Aneesah: Is this in connection with your previous request?				12934		3		1	3
33	38b5e87e-b0a0-44bc-9ff0-c75d083fd840	28/10/2016	16:48	28/10/2016 16:48	Rebecca:	Rebecca: yes				12935		3		1	3
41	38b5e87e-b0a0-44bc-9ff0-c75d083fd840	28/10/2016	16:48	28/10/2016 16:48	Aneesah:	Aneesah: I have discovered information reflected in 'ResultsView' window relevant to your input. Is this what you were looking to find out from the database?	SQLQUERY	SELECT * FROM (SELECT PRODUCTS.PROD_NAME, SUM(PROFITS.AMOUNT_SOLD), CHANNELS.CHANNEL_DESC FROM PRODUCTS INNER JOIN PROFITS on PRODUCTS.PROD_ID = PROFITS.PROD_ID INNER JOIN CHANNELS on PROFITS.CHANNEL_ID = CHANNELS.CHANNEL_ID WHERE PROD_NAME = 'Y Box' GROUP BY PRODUCTS.PROD_NAME, CHANNELS.CHANNEL_DESC)	Executed query response.	12942	DOMAIN	3	1	1	3

Example 2: Example shows ANEESAH performing conflict resolution.

LOG_INDEX	LOG_SESSION_ID	LOG_DATE	LOG_TIME	LOG_APPSTART	LOG_ACTOR	LOG_ACT_IOPUT	LOG_MATCH_HEADER	LOG_ITEM	LOG_DESCRIPTION	LOG_CONV_INDEX	LOG_MAIN_CONTEXT	LOG_SCENARIO_NO	LOG_MATCH_STENGTH	LOG_PARTICIPANT_NO	LOG_SQL_KNOWLEDGE
4412	d3beb7ff-7be8-4ded-b627-d68c793f569e	26/11/2016	11:43	26/11/2016 11:43	Jake:	Jake: I want you to call overall profit information made by company for year 1998				18581		4		16	1
4413	d3beb7ff-7be8-4ded-b627-d68c793f569e	26/11/2016	11:43	26/11/2016 11:43	Aneesah:	Aneesah: Is this in connection with your previous request?				18582		4		16	1
4414	d3beb7ff-7be8-4ded-b627-d68c793f569e	26/11/2016	11:43	26/11/2016 11:43	Jake:	Jake: no				18583		4		16	1
4420	d3beb7ff-7be8-4ded-b627-d68c793f569e	26/11/2016	11:43	26/11/2016 11:43	Aneesah:	Aneesah: I have found duplicate records for 1998 in database: 1. 1998 in calendar_year <i>A period of time containing 365 (or 366) days or period combining 12 calendar months</i> 2. 1998 in fiscal_year <i>Accounting or Tax period of 12 months of the company</i> Please select which record did you mean or enter record number to select appropriate record?				18585		4		16	1
4421	d3beb7ff-7be8-4ded-b627-d68c793f569e	26/11/2016	11:43	26/11/2016 11:43	Jake:	Jake: make a selection of option 2				18586		4		16	1

4422	d3beb7ff-7be8-4ded-b627-d68c793f569e	26/11/2016	11:43	26/11/2016 11:43	Aneesah:	Aneesah: I have discovered information reflected in 'ResultsView' window relevant to your input. Is this what you were looking to find out from the database?	SQLQUERY	SELECT * FROM (SELECT TIMES.FISCAL_YEAR , SUM(AMOUNT_SOLD - UNIT_COST) AS PROFIT FROM PROFITS INNER JOIN TIMES ON PROFITS.TIME_ID = TIMES.TIME_ID WHERE FISCAL_YEAR = '1998' GROUP BY TIMES.FISCAL_YEAR)	Executed query response.	18591	DOMAIN	4	0.95	16	1
4424	d3beb7ff-7be8-4ded-b627-d68c793f569e	26/11/2016	11:44	26/11/2016 11:44	Jake:	Jake: was there any from Africa region				18593		4		16	1
4425	d3beb7ff-7be8-4ded-b627-d68c793f569e	26/11/2016	11:44	26/11/2016 11:44	Aneesah:	Aneesah: Is this in connection with your previous request?				18594		4		16	1
4426	d3beb7ff-7be8-4ded-b627-d68c793f569e	26/11/2016	11:44	26/11/2016 11:44	Jake:	Jake: yes				18595		4		16	1
4435	d3beb7ff-7be8-4ded-b627-d68c793f569e	26/11/2016	11:44	26/11/2016 11:44	Aneesah:	Aneesah: I have found duplicate records for Africa in database: 1. Africa in country_region <i>Region is often known as continent that based on geographical location i.e. Europe etc.</i> 2. Africa in country_subregion <i>Subregion is a part of a larger region or continent and based on location i.e. Western Europe etc.</i> Please select which record did you mean or enter record number to select appropriate record?				18598		4		16	1
4436	d3beb7ff-7be8-4ded-b627-d68c793f569e	26/11/2016	11:44	26/11/2016 11:44	Jake:	Jake: option 1				18599		4		16	1

4437	d3beb7ff-7be8-4ded-b627-d68c793f569e	26/11/2016	11:44	26/11/2016 11:44	Aneesah:	Aneesah: I do not think there are any records present in database meeting your input. Would you like to ask information about something else?	SQLQUERY	SELECT * FROM (SELECT TIMES.FISCAL_YEAR, COUNTRIES.COUNTRY_REGION , SUM(AMOUNT_SOLD - UNIT_COST) AS PROFIT FROM times INNER JOIN profits on times.time_id = profits.time_id INNER JOIN customers on profits.cust_id = customers.cust_id INNER JOIN countries on customers.country_id = countries.country_id WHERE FISCAL_YEAR = '1998' and COUNTRY_REGION = 'Africa' GROUP BY TIMES.FISCAL_YEAR, COUNTRIES.COUNTRY_REGION)	Executed query response.	18604	DOMAIN	4	1	16	1
4439	d3beb7ff-7be8-4ded-b627-d68c793f569e	26/11/2016	11:45	26/11/2016 11:45	Jake:	Jake: I want you to include month names in it				18606		4		16	1
4450	d3beb7ff-7be8-4ded-b627-d68c793f569e	26/11/2016	11:45	26/11/2016 11:45	Aneesah:	Aneesah: I do not think there are any records present in database meeting your input. Would you like to ask information about something else?	SQLQUERY	SELECT * FROM (SELECT TIMES.FISCAL_YEAR, COUNTRIES.COUNTRY_REGION, TIMES.CALENDAR_MONTH_NAME , SUM(AMOUNT_SOLD - UNIT_COST) AS PROFIT FROM times INNER JOIN profits on times.time_id = profits.time_id INNER JOIN customers on profits.cust_id = customers.cust_id INNER JOIN countries on customers.country_id = countries.country_id WHERE FISCAL_YEAR = '1998' and COUNTRY_REGION = 'Africa' GROUP BY TIMES.FISCAL_YEAR, COUNTRIES.COUNTRY_REGION, TIMES.CALENDAR_MONTH_NAME)	Executed query response.	18616	DOMAIN	4	1	16	1

Appendix – H –Author Publications

Proceedings of the World Congress on Engineering 2015 Vol I
WCE 2015, July 1 - 3, 2015, London, U.K.

Aneesah: A Conversational Natural Language Interface to Databases

K. Shabaz, Jim D. O'Shea, *Member, IEEE*,
Keeley A. Crockett, *Senior Member, IEEE,* A. Latham, *Member, IEEE*

***Abstract* – This paper presents the design and development of a novel Natural Language Interface to Database (NLIDB). The developed prototype is called Aneesah the NLIDB, which is capable of allowing users to interactively/conversely access desired information stored in a relational database. This paper introduces the novel conversational agent enabled architecture of Aneesah NLIDB and describes the scripting techniques that has been adopted for its development. The proposed framework for Aneesah NLIDB is based on pattern matching techniques implemented to converse with users, handle complexities and ambiguities for building dynamic SQL queries from multiple dialogues in order to extract database information. The preliminary evaluation results gathered following a pilot study reveal promising results.**

***Index Terms* – Natural Language Interface to Databases (NLIDB), Conversational Agents (CA), Knowledge base, Artificial Intelligence (AI), Pattern Matching (PM).**

I. Introduction

Much of largest sources of information storages in public or commercial environment are databases. The information is a key element required when making decisions. Retrieving information from a database normally requires querying the database using a specialised programming code called SQL query, which is not accessible to the ordinary users. In order to avoid the difficulty of using SQL language for inexperienced users, there exists a need for creating applications that permit users to access desired information stored in a database. Natural Language Interfaces to Databases (NLIDBs) are computer programmes, which replace the requirement to react with a skilled programmer to query a database by using natural conversation with an agent.

Recently, the demand for Natural Language Interfaces to Databases has risen in result of many users accessing information from different types of systems such as computers, tablets, and smartphones [1]. NLIDB development attempts cover a time span of over half a century, however there still remain unsolved key challenges for their wider acceptance in public and commercial environment. Despite many development approaches, there are number of challenges that have been identified by researchers such as linguistic problems, conversational abilities, query translation, result refinement, domain independence, and ease of configuration [2].

This paper is organized as follows: Section II & III will give background of NLIDB and Conversational Agent applications. Section IV will describe existing problems and challenges in developing NLIBDs followed by Section V explaining the knowledge engineering of domain. Section VI describes the structure of framework adopted for prototype NLIDB (Aneesah). Section VII describes the Pilot Study, results and discussion. Section VIII will conclude and highlight areas for further work.

II. Natural Language Interface to Database

The idea of a computer taking the role of a human in conversation was first proposed by Alan Turing [3] and later was put into practice in the 70s with the development of the ELIZA Chatbot, which simulated a psychotherapist [4]. Among early development of NLIDBs, the most popular NLIDB was LUNAR [5]. LUNAR was built to perform moon rocks analysis based on an underlying database but it had functional limitations and could not be generalized to other domains. Soon after, the development of RENDEZVOUS was intended to simulate open dialogues to users in order to devise database queries [6]. The users anticipated questions had to closely match for the system to understand entered text, for query processing. LADDER development targeted big data and distributed databases. It was expected that LADDER would require substantial customization (.e.g. new grammar, domain knowledge), for adopting new domains [9]. PLANES designed features were based on the principals of RENDEZVOUS, which used a flight and airport database to answer users' questions [10]. PLANES system also required a new grammar and extensive customization to be able to work with a different domain [9].

In 80s, several other systems were developed such as CHAT80 which translated natural language in Prolog language [10] and TEAM which translated natural language queries into Simple Object Database Access query language [11]. Other systems such as PARLANCE built to resolve domain configuration issues and allowed users to manually configure underlying domain [12]. ASK also appeared as a cross application NLIDB [13] that allowed users to input their requests in natural language to generate appropriate responses from underlying database [14]. The development on NLIDBs continued to evolve in 90s with research on various commercial systems such as CHILL [2], INTELLECT [15], DATALINKER, Q&A Symantec and LOQUI [16]. In recent years some online systems appeared such as Wolfram Alpha, Powerset and TrueKnowledge that

Manuscript received March 23, 2015.
All authors are with the Intelligent Systems Group,
Department of Computing & Mathematics, Manchester Metropolitan University, Manchester, M1 5GD, UK. Email:
{k.shabaz, j.d.oshea, k.crockett, a.latham}@mmu.ac.uk

ISBN: 978-988-19253-4-3 (revised on 29 July 2015) WCE 2015
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

were designed to rely and work on initial information imported in the development [17]. More recent developments have contributed less efforts to create interaction between user and NLIDB systems. ITG is another developed NLIDB that urged users to repeatedly validate predicted text as user enters his/her input [18]. C-Phrase was developed on Codd's tuple calculus to allow context free grammar [19] and NaLIR a keyword based search interface developed using Natural Language Processing [20].

III. Conversational Agents

Conversational Agents (CAs) are also computer dialogue applications which allow users to communicate with computers using natural language. CAs have been used effectively across many fields i.e. advice & guidance, customer services, computerized learning etc [21]. Among best known early developments were ALICE that was tasked to keep the users engaged in non-goal oriented conversations [22]. Other types of CAs developments include goal based Conversational Agents such as ADAM built to simulate a debt advisor for students [23] and OSCAR which was developed for predicting learning styles whilst simulating as a tutor [24].

IV. Existing Challenges faced by Natural Language Interface to Databases

The development of Natural Language Interfaces has been around for decades, but to date these systems are not in wider popular use. There are number of associated factors which discouraged the industry from adopting NLIDB systems [25]. Among notable challenges and problems identified by most NLIDB researchers are namely; linguistic problems, domain independence, translation process (i.e. database query formulation), multimodality and ease of configuration, conversational abilities etc [26, 2]. The problems such as grammatical utterances, anaphora and quantifier scoping are some of the linguistic problems that a NLIDB has to tackle when attempting to interpret a user utterance [27]. The lack of ability to elicit intended goals from inexperience users to retrieve information from a relational database, query results refinement and autonomous behavior remain among unsolved key challenges. In complex and distributed databases, many NLIDB systems have revealed ellipsis problems. Selection of conceptual models in constructing NLIDBs are also highlighted as one of the linguistic problems [28]. It is not ideal for users to remember what kind of question a given NLIDB system can or cannot answer. In the case where a NLIDB fails in understanding user requirements, it is often difficult for a user to judge the reason behind system failure (i.e. whether its scope of the system, system abilities or coverage of domain etc) [17]. Major problems confronting NLIDB in formulation of SQL queries have been defined as semantic ellipsis and wrong words or phrases (i.e. missing key information, adjectives, verbs and prepositions etc), coverage capabilities of SQL (i.e. several tables, aggregative functions, excessive information), user errors (i.e. request for information outside the database records or invalid utterances [27]. SQL query formulation issues are also associated with the adopted development approach [26]. Other challenging areas that are relative in hindering the advancement of NLIDB technology to develop an ultimate system can be described as namely; achieving high accuracy rates with domain independence, portability of knowledge domain and underlying database to work in different environment, ability to read meta knowledge (knowledge about knowledge), and explore big data in real time etc [2].

Additionally, more relevant to the proposed research, lack of conversational abilities in NLIDBs have been also outlined as a frequent problem by the users [18]. The abilities to converse naive users [7, 26, 29]. The development of NLIDB systems have been largely focused on single query response transactions. Little attention has been given to social adaptability, sustained interaction and pro-active conversation to elicit what an end user envisages about the domain, and how domain knowledge is structured [30]. Development of Natural Language Interfaces (NLIs) to retrieve information from structured data, requires understanding of Natural Language and complexities, ambiguities etc [31]. A number of architectural techniques have been adopted to use limited natural language to generate a logical query for structured data. Such techniques include Syntax-Based Family of Architectures, Semantic-Grammar Family of Architectures and Pattern Matching [5].

V. Knowledge Engineering the Domain

The knowledge engineering for the prototype system was carried out by researching complex SQL query architecting techniques and leading commercial business intelligence reporting systems used in different organisations. A number of sales reports were studied in the interest of understanding business process mapping. The structure of sales reports was also analysed in contrast with the common SQL query syntax used to generate information from the database. The research conducted on sales reports and complex query structures helped in constructing SQL query formulation features for the Aneesah system.

A relational database in a commercial environment often represents complex structure and burdened with millions of records. The records maintained in an organization's database are used for various purposes and can be classified into variety of layered information. The layered information can be further divided into master data, transaction data, enterprise information, and database objects/meta data [32]. A conventional knowledge engineering technique was adopted after a critical review and research on existing NLIDB systems and commercial enterprise reporting systems (i.e. Oracle, Microsoft, SAP, ASAP etc) to engineer the knowledgebase for the prototype system. This information modelling technique relies on capturing, representing, encoding and evaluation of expert knowledge (Chu, Hwang, Huang, and Wu, 2008). The prototype system has been equipped with expert knowledge with a generic approach in mind to allow naïve users to interactively retrieve and analyse information stored in chosen database for the developed system.

VI. ANEESAH NLIDB

Aneesah is a novel NLIDB system with conversational abilities to provide interactive and friendly environment to assist users with desired information stored in a relational database. Aneesah is able to correspond to the user requirements, guide the user to his/her intended goal and produce desired information from underlying relational database. The proposed architecture was developed by adopting the Pattern Matching (PM) approach.

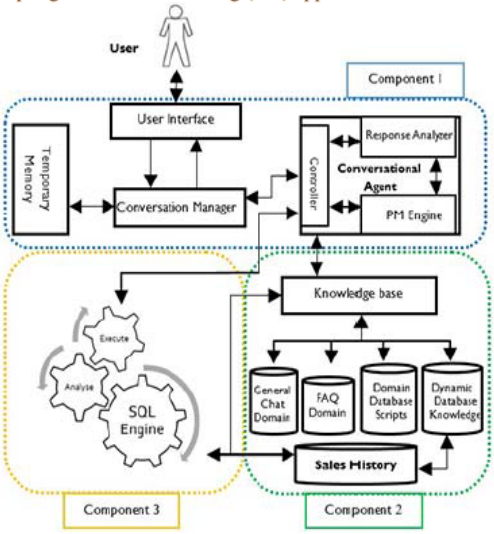


Fig 1. Aneesah NLIDB Architecture

Aneesah employs a novel framework constructed with a conversational agent based on scripting language, knowledge base and SQL engine. The developed prototype (Aneesah NLIDB) has been tested through a new evaluation methodology implemented using subjective and objective metrics to determine usefulness and effectiveness based on system experiments conducted following a pilot study. Figure 1 gives a high level overview of proposed architecture and role of components in supporting the Aneesah system's role to simulate as a human SQL developer by leading users in conversation to produce desired information from the relational database. A generic approach has been used to develop the proposed architecture, which is modular by nature for ease of development and maintenance with flexibility to integrate and adopt other domains with customization. The proposed framework comprises three major components as shown in Figure 1.

Component 1

Component 1 of the Aneesah system's architecture comprises of Conversation Manager, User Interface, and Temporary Memory and Conversational Agent components (Controller, Pattern Matching (PM) Engine, Scripting Language, and Response Analyzer). The Aneesah system features a novel Conversational Agent (CA) that is a fundamental part of the proposed architecture, which enables users to interact/converse with the Aneesah system for information retrieval purposes.

A. Controller

The controller module has been designed to handle direction of conversation with the users towards achieving desired information from the database. The controller leads conversation with the system users during interactive sessions. The controller also takes the responsibility of validating each user utterances before passing them to the PM engine for further processing in the system. An invalid utterance is a one off topic or not relevant to the domain, contains offensive or swearing words, or contains unnecessary characters. The controller features a built in three warnings rule in situations where it detects offensive words or abuse of the system. The user is shown with three warning messages detailing reason and error on their part, before system is closed by the controller. The validated utterances are forwarded to the PM engine by controller.

B. Pattern Matching Engine

The Aneesah system has been implemented using a novel PM engine. The PM engine controls user utterance matching against scripts in the system's knowledge base. The PM engine works based on a two tier approach (Tier 1 and Tier 2) approach. Tier 1 deals with user utterance matching against information stored in domain database (i.e. sales history database etc) tables to capture co-occurrence of attribute or key records leading to the formulation of a query based response, and Tier 2 deals with utterance matching against hand scripted patterns stored in Frequently Ask Questions (FAQ) and General Chat domains. The PM engine has been designed to work with rule based and non-rule based response handling. A rule based response can be described as a scripted textual response, executed following a successful utterance matching in either FAQ domain or General Chat domain. A non-rule based response relates to the formulation of a database query following successful utterance processing against Domain Database Scripts. The PM engine works on principle of pattern matching approach, which partially resembles an approach implemented in InfoChat system [4]. The PM engine works in conjunction with sentence similarity strength function to determine an appropriate response, and also interact with the user to resolve any ambiguity during conversation. The PM engine is equipped to deal with more than one response match situation. In the case, where a user utterance has attracted duplicate responses from different knowledge base domains, the PM engine uses sentence similarity calculation to execute the highest match value response. The user utterance is categorized once a match has been found in the knowledge base. The relevant domain (Domain Database Scripts, FAQ Domain or General Chat Domain) is activated once a user utterance is matched in the knowledge base. The domain activation is used by the PM engine to conversely engage with user and staying relevant to the conversing topic.

C. Pattern Matching Scripting Language

This section details the new scripting language approach adopted for the Aneesah system that allows users to engage in conversation with the system. The development of NLIDBs have been researched by number approaches, which represent associated challenges as discussed in literature review section (2.0) of this report. The pattern matching approach has been selected as the most suitable mechanism for developing a scripting language for the development of the Aneesah system. In order to model a novel scripting language for the Aneesah system, an extensive research was conducted on conversational systems such as InfoChat system [4].

D. User Interface

The Aneesah system is implemented with command line based user interface. The user interface is responsible for serving as an interaction platform between the user and Aneesah system. The user interface receives user utterances and displays back responses in the interface window. The information presenting technique has been implemented that has made it possible to display variable system responses in all forms i.e. text, quoted notions, query results etc.

Component 2

Component 2 of Aneesah's architecture consists of Knowledge base to serve as Aneesah's brain that has been developed based on the knowledge engineering techniques to specifically work with the domain database (Sales History database). The developed knowledge base can be customized with knowledge engineering to work with a different database. Responsible for supporting a two tier based user utterance matching across three different domains contexts. It is also responsible for providing domain knowledge to carry out user utterance mapping to match domain database information and support requests processing with minimum information from the users.

E. Knowledge base

The Knowledge base for Aneesah has been implemented in four layers of stored scripts namely; Domain Database Scripts, Frequently Asked Questions (FAQ) Domain, General Chat Domain, and Dynamic Database Knowledge (DDK). The Domain Database Scripts module plays fundamental role in prototype system by housing domain database relevant scripts. It is responsible for assisting PM engine to perform user utterance mapping against database information to capture database relevant information. FAQ contexts handles user questions related to the database structure. The General Chat domain handles user utterances outside database relevance for example user might want to talk about football, weather, etc. In this situation, the system attempts to motivate the user to stay relevant to the system's built purpose. The information maintained in sample sales history database, used for Aneesah's evaluation, is dynamically loaded into the DDK module on execution of system. The DDK module allows PM engine to perform matching of data records at all times and vacates the actual database from being reserved for match processes.

Component 3

Component 3 of the Aneesah architecture comprises novel SQL Query Engine with its components. The SQL Query Engine after having received database relevant syntax/information from controller component, is responsible for formulation of SQL queries in contrast with the users' utterances. The SQL engine identifies type and nature of queries following by activation of SQL configurator component.

F. SQL Query Engine and Components

The SQL Engine takes pivotal role in the Aneesah system's architecture. The SQL engine is responsible for performing dynamic query translation/formulation process from user utterances. The SQL engine works based on implemented techniques together with other sub components (SQL Configurator, SQL Execution, and SQL Analyser) in order to retrieve information from the database. The SQL engine relies on database relevant information delivered by the system's controller to analyse syntax requirements to engineer a query structure. The Aneesah system's query authoring knowledge extends users abilities to request information from any of 118 database columns in complex combination. The Aneesah system has been tested to support up to five tables joins. The table 1.1 illustrates the scope of SQL query formulation capabilities. The SQL engine utilises implemented query syntax to formulate database queries depending on the nature of users' requests.

The SQL Query engine is equipped with expert knowledge to identify level of complexity involved in a query formulation. In a real world scenario, a simple user request might lead to the need of variety of SQL query related syntax (i.e. database objects, aggregation functions, selection of results, filter etc) to satisfy his/her requirement. Therefore, the Aneesah system features an automated query structuring approach, which has the ability to gather necessary syntax to formulate query structures. The query formulation process adopts a step by step query syntax preparation process. The PM engine also initiates the SQL configurator to formulate a query. The query information is forwarded to the SQL configurator for further processing. The SQL configurator is responsible for generating syntactic information required to put together a database query structure. The formulated queries are transferred to SQL execution component, which takes the responsibility of executing and retrieving queries relevant results from the database. In addition to the query returned results, a scripted text response is also displayed to the user. The SQL analyser component has been implement as a failsafe tool, which takes the role of a database query structure analyser. Figure 2 shows the representation of query returned results in the user interface.

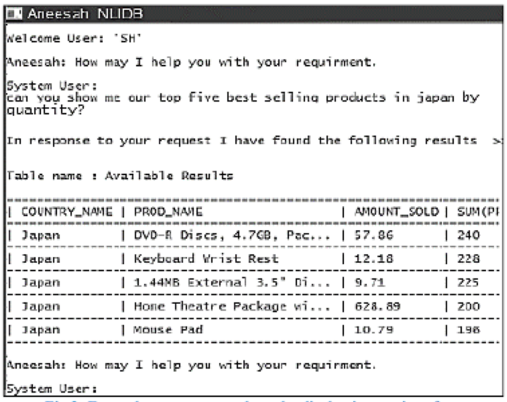


Fig 2: Example query returned results display in user interface

VII. Pilot Study

A. Domain

The domain database (Sales History database) used for prototype system's evaluation plays a fundamental role. It serves the main information retrieval source during

ISBN: 978-988-19253-4-3 (revised on 29 July 2015) WCE 2015
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

conversation sessions with the users. The chosen database comprises complex schema structure with records of large data related to Sales History (SH) records. The short listed SH schema for Aneesah system is particularly designed to demonstrate large amount of data in complex relational structure. The structure of the SH schema has been developed with sample sales records of electronic products belonging to an assumed company in view, which maintains a high volume operating business. The structure of SH database comprises 8 database tables with 114 columns containing domain relevant information.

B. Experimental Methodology

The evaluation methodology for the system required use of wider spectrum of evaluation (subjective & objective) metrics. The formulation of evaluation metrics was related to the major features of Aneesah system namely; dialogues naturalness, robustness and information accuracy etc. The developed system was evaluated through designed experiments, based on the formulated evaluation metrics. There were two experiments designed to provide the information required to carry out evaluation of Aneesah system. The information obtained from experiments was measured through statistical tests such as descriptive and inferential statistics. There were twenty participants selected to take part in evaluation. The participants were divided in two groups (Group-A & Group-B). The first Group-A comprised ten participants without possession of any databases or Structural Query Language (SQL) knowledge. The equal number of participants were selected for the second group (Group-B), who had familiarity with relational databases and SQL language either because of academic subjects or employment in similar field. The main purpose behind grouping participants in two equal groups was to take rigour system insights at initial testing stage. The null hypothesis (H0) was assumed that a general user cannot interact with a Natural Language Interface to Database to formulate a query and retrieve desired information from a relational database.

C. Results and Discussion

The questionnaire results for Experiment-1 from both participant groups (Group A & Group B) show that the Aneesah system has been well received. Overall 75% of participants from both groups rated the system interface (frontend) and level of understanding at high, however 30% of participant's rated these features between low and medium. Around 70% of the participants from both groups found the Aneesah system to be active, efficient and useful. The satisfaction level from Aneesah's dialog responses have been rated high by 70% of participants from both groups. Overall 90% of the participants agreed on using a similar system. The aim of Experiment 2 was designed to perform analysis such as robustness and accuracy. The information captured from the system's log file, which recorded occurrences of dialogs between participants and the Aneesah system during test sessions. For Group-A participants without possession of any databases or Structural Query Language (SQL) knowledge, the system has shown accuracy at 94.02% and precision was recorded at 90%. The evaluation of results performed for Group-B participants having familiarity with relational databases and SQL query language, measured less accuracy and precision comparing to the Group-A participants. The accuracy for Group-B participant is recorded at 80% and its precision is recorded at 87.50% for the same. In addition to the system's overall accuracy for both experimental groups (Group-A & Group-B) being scored at 85.01% and precision at 92.96%, the harmonic mean or F-measure equation is used to measure test of system accuracy by combining above accuracy and precision, which was recoded at 88.80%. Therefore, the null hypothesis (H0) is rejected and alternative hypothesis (H1) is assumed that, a general user can interact with a Natural Language Interface to Database to formulate a query and retrieve desired information from a relational database.

VIII. CONCLUSION AND FUTURE WORK

This paper presents a novel framework for the development of a conversation based NLIDB system. During evaluation, the proposed approach has shown its ability to allow users to take conversation based insights into the domain database. The development of Aneesah system was carried out by investigating existing approaches adopted for the development of similar systems. The aim of new approach described in this report, is to interpret users' requests by understanding and interacting with the users during conversation to produce required information from the database. The pattern matching approach solves the language complexities by simply matching user requests against scripted patterns and database information. Future work will be directed to evaluate Aneesah's abilities to handle more complex user requirements and to sustain dialogues and engage users in smart conversation to perform analysis of information by refining query results.

REFERENCES

[1] Elmasri, R., Navathe, S.B., 2010. Fundamentals of Database Systems, 6 edition. ed. Addison Wesley, Boston.

[2] Zelle, J.M., Mooney, R.J., 1996. Learning to Parse Database Queries Using Inductive Logic Programming, in: Proceedings of the Thirteenth National Conference on Artificial Intelligence - Volume 2, AAAI'96. AAAI Press, Portland, Oregon, pp. 1050–1055.

[3] Turing, A.M., 1950. Computing machinery and intelligence. Mind 433–460.

[4] O'Shea, J., Bandar, Z., Crockett, K., 2011. Systems Engineering and Conversational Agents, in: Tolk, A., Jain, L.C. (Eds.), Intelligence-Based Systems Engineering, Intelligent Systems Reference Library. Springer Berlin Heidelberg, pp. 201–232.

[5] William A. Woods, Robert M. Kaplan, and Bonnie Lynn Nash-Webber. The Lunar Sciences Natural Language Information System: Final Report. Technical Report BBN Report 2378, Bold Beranek and Newman Inc., Cambridge, Massachusetts, 1972.

[6] Woods, W. (1973). An experimental parsing system for transition network grammars. In Natural Language Processing, R. Rustin, Ed., Algorithmic Press, New York.

[7] Hendrix, G.G., Sacerdoti, E.D., Sagalowicz, D., Slocum, J., 1978. Developing a Natural Language to Complex Data. ACM Transactions on Database Systems 3, 105–147.

[8] Waltz, David (1975) Natural language access to a large database: an engineering approach. In Advance papers for the Fourth International Joint Conference on Artificial Intelligence (IJCAI-75), 868–872, Tbilisi, Georgia, USSR. – 8

[9] Waltz, D.L., 1978. An English Language Question Answering System for a Large Relational Database. Commun. ACM 21, 526–539. doi:10.1145/359545.359550

[10] David H. D. Warren and Fernando C. N. Pereira. An Ecient Easily Adaptable System for Interpreting Natural Language Queries. Computational Linguistics, 110-122. , July 1982. ISSN 0891-2017. URL http://portal.acm.org/citation.cfm?id=972942.972944

[11] Barbara J. Grosz, Douglas E. Appelt, Paul A. Martin, and Fernando C. N. Pereira. TEAM: An experiment in the design of transportable natural language interfaces. Artificial Intelligence, 32(2):173 – 243, 1987.

[12] Madeleine Bates (1989) "Rapid Porting Of the Parlance Natural Language Interface" In Proceedings of the Speech and Natural Language Workshop February 1989, Morgan Kaufmann Publishers, Inc., San Mateo, California, Pages 83-88.

[13] B.H. Thompson and F.B. Thompson. Introducing ASK, A Simple Knowledgeable System. In Proceedings of the 1st Conference on Applied Natural Language Processing, Santa Monica, California, pages 17–24, 1983.

[14] B.H. Thompson and F.B. Thompson. ASK is Transportable in Half a Dozen Ways. ACM Transactions on Office Information Systems, 3(2):185–203, April 1985.

[15] N. Ott, "Aspects of the Automatic Generation of SQL statements in a Natural Language Query Interface", Information Systems, 17 (2), 1992, pp. 147-159.

[16] W. Sijtsma and O. Zweekhorst. Comparison and Review of Commercial Natural Language Interfaces. In F.M.G. de Jong and A. Nijholt, editors, Natural Language Interfaces, From Laboratory to Commercial and User Environments – Proceedings of the 5th Twente Workshop on Language Technology, Enschede, Twente University, NL, June 1993.

[17] Pazos, R.R.A., Rojas, P.J.C., Santaolaya, S.R., Martínez, F.J.A., Gonzalez, B.J.J., 2010. DialogueManager for a NLIDB for Solving the Semantic Ellipsis Problem in Query Formulation, in: Proceedings of the 14th International Conference on Knowledge-Based and Intelligent Information and Engineering Systems: Part II, KES'10. Springer-Verlag, Berlin, Heidelberg, pp. 203–213.

[18] Pazos R, R.A., González B, J.J., Aguirre L, M.A., Martínez F, J.A., Fraire H, H.J., 2013. Natural language interfaces to databases: an analysis of the state of the art. Recent Advances on Hybrid Intelligent Systems 463–480.

[19] Raymond J. Mooney, Learning Language from Perceptual Context:A Challenge Problem for AI, American, Association for Artificial Intelligence (2006).

[20] Li, F., Jagadish, H.V., 2014. NaLIR: An Interactive Natural Language Interface for Querying Relational Databases, in: Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data, SIGMOD '14. ACM, New York, NY, USA, pp. 709–712. doi:10.1145/2588555.2594519.

[21] P. Reis, N. Mamede, and J. Matias, "Edite – A Natural Language Interface to Databases: a New Dimension for an Old Approach", Proceeding of the Fourth International Conference on Information and Communication Technology in Tourism, Edinburgh, Scotland, 1997.

[22] Wallace, R. S. (2008). ALICE: Artificial Intelligence Foundation Inc. Available at:http://www.alicebot.org.

[23] Pudner, K., Crockett, K.A., Bandar, Z., 2007. An Intelligent Conversational Agent Approach to Extracting Queries from Natural Language. In: World Congress on Engineering. pp. 305–310.

[24] Latham, K. Crockett, D. McLean and B. Edmonds (2010). Oscar: An Intelligent Conversational Agent Tutor to Estimate Learning Styles, 2010 IEEE International Conference, Barcelona.

[25] R, R.A.P., B, J.J.G., L, M.A.A., F, J.A.M., H, H.J.F., 2013. Natural Language Interfaces to Databases: An Analysis of the State of the Art, in: Castillo, O., Melin, P., Kacprzyk, J. (Eds.), Recent Advances on Hybrid Intelligent Systems, Studies in Computational Intelligence. Springer Berlin Heidelberg, pp. 463–480.

[26] Crockett, K., Bandar, Z, O'Shea, J and Mclean, D. (2009) 'Bullying and Debt: Developing Novel Applications of Dialogue Systems' In International Joint Conferences on Artificial Intelligence. Knowledge and Reasoning in Practical Dialogue Systems.

[27] Androutsopoulos, I., Ritchie, G., Thanisch, P.: Natural Language Interfaces to Databases - An Introduction. Natural Language Engineering 1, 29–81 (1995).

[28] H.R. Tennant, K.M. Ross, M. Saenz, C.W. Thompson, and J.R. Miller. Menu-Based Natural Language Understanding. In Proceedings of the 21st Annual Meeting of ACL, Cambridge, Massachusetts, pages 151–158, 1983.

[29] P.R. Cohen. The Role of Natural Language in a Multimodal Interface. Technical Note 514, Computer Dialogue Laboratory, SRI International, 1991.

[30] Owda, M., Bandar, Z., Crockett, K., 2007. Conversation-based natural language interface to relational databases, in: Web Intelligence and Intelligent Agent Technology Workshops, 2007 IEEE/WIC/ACM International Conferences On. pp. 363.

[31] Damljanovic, D., Agatonovic, M., Cunningham, H., 2012. FREyA: An Interactive Way of Querying Linked Data Using Natural Language, in: García-Castro, R., Fensel, D., Antoniou, G. (Eds.). Springer Berlin Heidelberg, pp. 125–138.

[32] Sapinsider.wispubs.com, (2015). Visualizing SAP: The Connected SAP Universe. [online] Available at: http://sapinsider.wispubs.com/Assets/Blogs/2011/October/Visualizing-SAP-The-Connected-SAP-Universe [Accessed 11 Jan. 2015].

[33] Docs.oracle.com, (2015). Sample Schema Scripts and Object Descriptions. [online] Available at: http://docs.oracle.com/cd/B28359_01/server.111/b28328/scripts.htm#Cihijfgh [Accessed 23 Mar. 2015].

ISBN: 978-988-19253-4-3 (revised on 29 July 2015) WCE 2015
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)