Natural Language Access to Relational Databases:

An Ontology Concept Mapping (OCM) Approach

By Lawrence Muchemi Githiari

P80/80034/2008

Supervisor: Dr. Wanjiku Ng'ang'a.

Thesis presented for the Award of Degree of Doctor of Philosophy in Computer Science

School of Computing and Informatics

University of Nairobi, Kenya

Declaration

This thesis is my original work and has not been presented for a degree in any other University.

Signature …… Date ……

Name Lawrence Muchemi Githiari

This thesis has been submitted for examination towards the fulfillment for the award of degree of Doctor of Philosophy in Computer Science with my approval as the Supervisor.

Signature …… Date ……

Name Dr. Wanjiku Ng'ang'a

School of Computing and Informatics,

University of Nairobi, Kenya

Abstract

Many applications that are accessed by non-technical or casual users, who prefer the use of natural language, rely on relational databases. Examples of such applications include government data repositories such as government tender information portals or application specific databases such as agricultural support systems. The problem of natural language (NL) processing for database access which has remained an unresolved issue forms the main problem addressed in this work. The specific challenges include lack of a language- and domain-independent methodology for understanding un-restrained NL text that accesses monolingual of cross-lingual databases as well as concepts extraction from database schema.

It is demonstrated that an ontology based approach is technically feasible to handle some of the challenges facing NL query processing for database access. The Ontology Concept Modelling (OCM) approach relies on the ability to convert databases to ontologies from which we obtain the underlying concepts. The database concepts are matched against the concepts obtained from natural language queries using a semantically-augmented Levenshtein distance algorithm. This thesis presents the architecture and the associated algorithms for an OCM-based model for NL access to databases.

In order to evaluate and benchmark the OCM model, data was generated from a prototype based on the developed OCM-based model. Quantitative parameters such as accuracy, precision, recall and the F-score and qualitative measures such as domain-independence, language independence, support for cross-lingual querying and the effect of query complexity on the model were evaluated across five data sets. Studies were conducted for English, Kiswahili and English-Kiswahili pair of languages in a cross-lingual manner from which attainment of language and domain independence for database access are demonstrated. For this language pair, it is also shown empirically that it is adequate to incorporate a bilingual dictionary at gazetteer level for cross-lingual data retrieval.

To evaluate the performance of the developed OCM-model, test-beds comprising of mono-lingual, cross-lingual as well as cross-domain performance measurements capacity were designed to test various aspects of the model. Tests were then conducted and the results indicated that OCM has a marginally better precision of 0.861 compared to other bench-marking models selected for comparison. Further OCM has an average F-score of 0.78 which compares well to other bench-marking models.

The main contribution of this work especially on the OCM architecture, processing algorithms such as OWoRA (Ontology Words Recovery Algorithm) and Frameworks such as QuSeT (Query semantics transfer framework) and evaluation models have a huge significance to the research and developer communities as they provide novel approaches to NL database access and model evaluation techniques.

Keywords: Natural Language Query, Database Access, Ontology Concept Modeling

Acknowledgement

PhD is a journey which is long with many meanders that make it one of the most interesting that an individual makes in the academic arena. Thanks to almighty God for journey mercies and seeing a successful end of this journey. Being a part-time venture made it particularly challenging and unto this point I thank the University of Nairobi as an institution for putting in place measures that helped ease this burden. In particular, I acknowledge UoN for providing fees waiver and allowing reduced staff workload.

This work would not have seen its infant days without the assistance of Dr. Kate Getao who provided the initial guidance in shaping the overall research area. An immeasurable amount of gratitude goes to my supervisor Dr. Wanjiku Ng'ang'a who walked with me through the entire journey providing the required critique in shaping the work to the current status. In particular her contributions in thesis structuring, insights into grammar based components and diligent thesis review all made my PhD journey smoother. I appreciate her efforts in the much needed assistance in critiquing and reviewing the academic paper on tokens and phrase tree SQL template mapping (TTM) presented in Makerere in 2008 at the 4th ICCR conference. My heart felt gratitude also goes to Prof. Fred Popowich who gladly accepted me as a visiting PhD student at Simon Fraser University Canada in the year 2010. Thanks for his diligence in co-authoring and reviewing one of the most informative papers arising from this thesis and later published by Springer International Journal as a book Chapter. The work was also presented at HCII conference Las Vegas, USA in July 2013. During my stay in Canada I met a wonderful group of students at the Natural Language Lab who together we studied and reviewed some interesting and often challenging NLP problems; I thank them all. Part of this research was made possible by Foreign Affairs and International Trade Canada (DFAIT) funding through the Canadian Commonwealth Scholarship Program. It was also supported in part by a Discovery Grant from the Natural Sciences and Engineering Research Council of Canada.

Much gratitude goes to the faculty at School of Computing and Informatics. In particular, thanks to Dr. Elisha Opiyo for providing encouragement when we were office-mates, Prof. Wagacha for making everything look possible and easy, Prof. Okelo-Odongo for the perfect administrator he is and providing critique right from the proposal stages, Prof. Waema and Omwenga for encouragement and prodding me to keep on going. I also direct my gratitude to other faculty members who were PhD students and finished last year (Orwa, Oboko, Omwansa, Chepken and Mwaura) or are about to finish (Miriti, Ayienga, Ruhiu, and Christine) for keeping each other company and providing useful suggestions.

Finally, I thank friends and family members particularly my wife Alice who is a Kiswahili academic and provided a lot of support in grammar related issues in this work. My mum Esther for urging me to finish and always asking me *"Kibuku gigathira ri?"*(When are you going to be through with the big book (PhD thesis)?), Sarah for assisting in data collection for Kiswahili questions and all that contributed in one or more ways. I also thank my children for having endured a dad who was always on the computer doing seemingly endless things.

Dedication

To all those who work in pursuit of

Elevating the position of African Languages in the international arena through

Natural Language technology advancement

Table of Contents

List of Tables

List of Figures

List of Acronyms

AET Addition of Elements Transformation

AI Artificial Intelligence

AMT Automatic Machine Translation

CCG Combintory Categorial grammar

CFG Context Free Grammar

CNL Controlled Natural Language

CT Coordination Transformation

DAT Deletion of Agent Transformation

DB Database

DET Deletion of Excessive Elements Transformation

DSF Deep Structure Form of a query as defined in Transformational-Generative Theory

DCG Definite clause grammar

FOL First Order Logic

FSM Feature Space Model

GO Gene Ontology

HCS Helsinki Corpus of Swahili

IE Information Extraction

I-O-B Tags indicating word is Inside, Outside or beginning of a chunk

IT Imperative Transformation

MBSMA-s Memory Based Tagger that works at syllable level

MR Meaning Representation

NLTK Natural Language Tool Kit

NL/U/P/Q Natural Language/Understanding/Processing/Query

NLADB Natural Language Access to Data-Base

NT Negation Transformation rule

OWL Web Ontology Language acronymed OWL, is a knowledge representation language that has three sublanguages Lite, DL and Full. It is used for authoring ontologies and is written in either RDF/XML format. OWL is endorsed by the World Wide Web Consortium (W3C)

OCM Ontology Concepts Mapping

OWoRA Ontology Words Construction Algorithm

PoS Part of Speech

PCCG Probabilistic Combinatory Categorial Grammar

PT Passive Transformation

QA Question Answering systems

QT Question Transformation rule

QuSeT Query Semantics Transfer Model

RDF Resource Description Framework

RegExp Regular Expression

SACoMA Semantically Augmented Concepts Matching Algorithm

SALAMA Swahili Language Manager

SeRQL Sesame RDF Query Language is an RDF query language used to query RDF repositories

SPARQL Is an RDF query language that is emerging as the de facto RDF query language (W3C endorsed)

SQL Structured Query Language for querying relational databases

SRL Semantic Role Labeling, also called shallow semantic parsing, is based on theta roles analysis

SSF Surface Structure Form of a query as defined in Transformational-Generative Theory

SVM Support Vector Machine

SVO Subject-Verb-Object in a query

SWATWOL a morphological parsing program based on two-level formalism (for handling morphophonological processes, which occur principally in morpheme boundaries.

TTM phrase-Tree Template Matching Approach to NLQ access to Relational databases

TUKI Taasisi ya Uchunguzi wa Kiswahili (Institute of Kiswahili Research, University of Dar es Salaam)

W3C World Wide Web Consortium

Chapter 1: INTRODUCTION

1.0 Background

Language is the principal manifestation of human intelligence and therefore its processing and effective use through technology is an epitome of artificial intelligence. Natural Language Processing (NLP) has developed over the years from a minor sub-branch of Artificial Intelligence (AI) to a well-researched major sub-field of AI. Currently major topical fields of NLP include Sentiment Analysis, Opinion Mining, Automatic Machine Translation (AMT), Question Answering Systems (QA), Information Extraction (IE), Deciphering, Dictation Systems, and Transcription Systems among others. The latter two areas involve speech processing as opposed to the former which are text based. The work undertaken in this thesis falls in the area of question answering system but in the focused area of database access. Question answering systems are specialized information access systems that have deduction capability that enables formulation of answers from information repositories through synthesized natural language queries. Synthesize of Natural Language queries involves natural language understanding. Natural Language (NL) understanding refers to the process of comprehending and making intelligent language use once the concepts are known. In general natural language understanding is a notoriously difficult problem because it seeks to understand open-ended natural language utterances that require knowledge and reasoning skills that people use in everyday life (Mateas & Stern, 2011). Making judgments on grammaticality is not a goal in language understanding (Robin, 2010). Robust systems should therefore understand ungrammatical sentences with semantic value. This necessitates exploring of approaches that are robust enough to handle issues of ungrammatical texts, sentence fragments and short queries. Question answering systems should be distinguished from ordinary search engines by the fact that in QA systems a direct answer is deduced from the information source as opposed to provision of a set of web links that could contain the answer.

According to Lopez (2007), QA systems can be grouped into four distinct categories which are based on the nature of data being accessed. The first category involves data source which is highly structured, such as a relational database, and is normally accessed by highly formal languages, such as SQL (Structured Query Language), SPARQL(recursive acronym for SPARQL protocol and RDF Query Language pronounced ‘sparkle’) and SeRQL (acronym for Sesame RDF Query Language) etc. The second category encompasses access to semi-structured data source such as health records or

yellow pages information. The third category involves question answering over free text from such sources as the web pages. The fourth category involves accessing annotated images or video via ontologies. The work reported in this thesis is in the area of Natural Language Access to Data-Base (NLADB) which is in the first category as described above.

Research in NLADB has been ongoing since the sixties as evidenced by the 1961 program named BASEBALL (Green,1961) and the 1964 attempt by Bobrow (Bobrow, 1964) on Natural Language (NL) for Algebra program, a doctorate thesis at the Massachusetts Institute of Technology (MIT). Active research continued in the early 70's and popular programs included SHRDLU (Winograd, 1971) another doctoral thesis at MIT, which demonstrated NL control to a robot's arm that is placed on a table. LUNAR (Woods, 1972) answered close to 90% of the questions about geological properties of rocks returned by the Apollo missions (Lopez, 2007). Other popular programs of mid-70's included PLANES (Waltz, 1975), REL (Thomson, 1975) and LADDER (Hendrix, 1978) among others. The motivation in 1970's was provision of natural language access to Expert Systems (Akerkar & Joshi, 2009). BASEBALL for example had an NL interface to an expert system that helped answer close to 80% of questions on United States baseball league information. Almost all NLADB systems were designed and developed with a particular database in mind, an approach that is not tenable because of the inherently high cost of development of these single use-interfaces.

Programs in the late 70's and 80's differed from those of the 60's and early 70's in that they utilized semantic grammars while the earlier ones used purely syntactic grammars (Akerkar & Joshi, 2009). The semantic grammar approach is an approach in which non-terminal symbols of the developed grammar use word-entities such as maths_score, rank_of_worker etc. while syntactic grammar utilizes syntax trees with syntactic categories such as noun-phrases and part of speech categories such as verbs as non-terminal symbols. The use of syntactic and semantic grammars in the direct conversion of English free text to formal languages such as prolog predicates as used in MASQUE (Androutsopoulos, 1993) continued to be used in 80's and 90's. Later developments advocated for conversion of NL to SQL queries as observed in MASQUE/SQL (Androutsopoulos, 1993). The inherent challenges brought by direct conversion of NL to structured language such as lack of language and domain independence were discovered. Restrictions on language usage were a big bottleneck to this approach. For example sentences were required to be grammatically correct and in a predefined format and more often than not required paraphrasing. The method does not allow shifting from one language to another or from one domain to another without vast customization

efforts that are not viable. An Interlingua approach was proposed and adopted by several researchers (Vanessa Lopez, 2007). For example Dong-Guk Shin and Lung-Yung Chu (1998) developed a theory of using concept terms, which they referred to as c-terms which essentially acted like interlingua representations. Efforts in the use of interlingua culminated in development of powerful commercially available systems such as English Wizard (EasyAsk), English Query (Microsoft) and ELF (Elf Software Co). These earlier efforts have been less successful than it was once predicted, mainly because of the development of alternative graphic and form-based databases (Akerkar & Joshi, 2009). The long time desire to minimize the communication gap between computers and humans through NLADB is persistent and hence the need for continued research in this area (Rashid, Mohammad, & Rahman, 2009). In usability studies reported in Kaufmann and Bernstein (2007) involving 48 end users of a QA system, NL was the most preferred access technique compared to menu and graphical interfaces. Over the years NLADB researchers have recognized this potential and hence the need for continued research.

1.1 Advances in Natural Language Query (NLQ) Processing for QA

The problem of NL access to databases has been recognized as having two integral parts which include a linguistic layer that handles natural language processing tasks and a database access layer that handles structured queries mainly in SQL but recently SPARQL and SeRQL. Other researchers often include a third layer to cater for the intermediate representation (interlingua). Generally speaking research has concentrated in these three areas with much work done through machine learning, statistical, rule based or a hybrid of these methods. Currently, research efforts are at a cross roads where researchers are grappling with the question of using rule based approach or moving to the more attractive statistical and machine learning approaches which have significantly improved the results of other NLP problems such as automatic translation. The problem has been cast as indicated in figure 1.1.

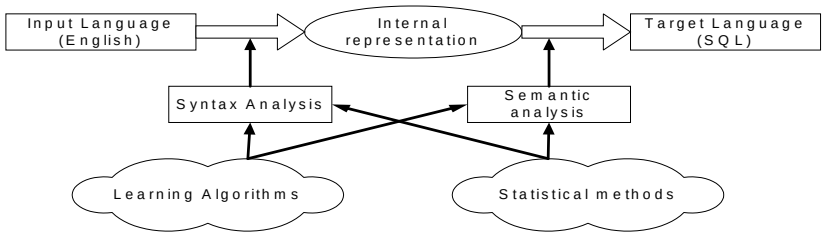


Figure 1.1 NLADB Using Machine Learning and Statistical Methods (adopted from (Minker, 1997))

Researchers pursuing machine learning and statistical approach can be categorized into various schools of thought as noted by Mingxia, Jiming, Ning, and Furong, (2007). One such category involves researchers who view the problem as a classification or a clustering problem. The core aim is to provide efficient semantic parsers. Generally speaking semantic parsers take in free NL text and map this to some formal representation of meaning. First Order Logic (FOL) is usually used for formal meaning representation. The meaning representations (MRs) are subsequently mapped to SQL or other structured languages via a machine learning classification process. This process is illustrated in figure 1.2

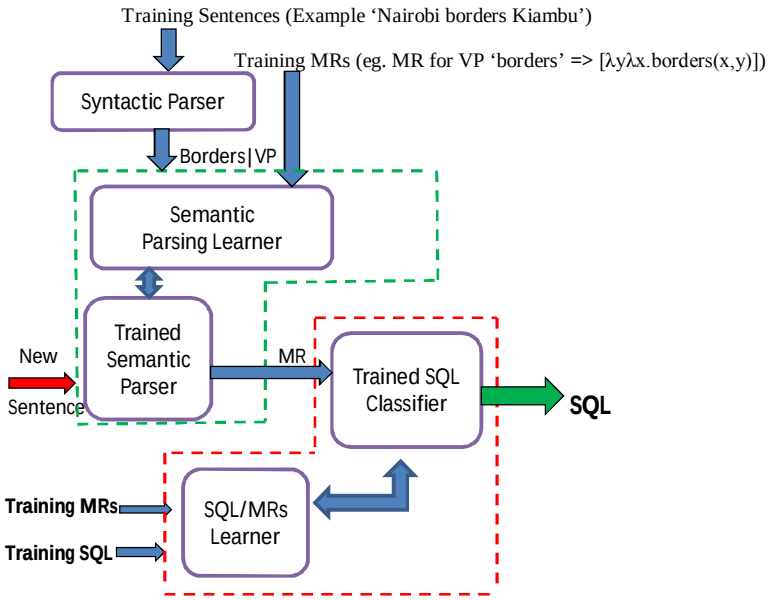


Fig. 1.2 Semantic Parsing Approach to SQL Generation

The most preferred grammars are the definite clause grammar (DCG) and probabilistic combinatory categorial grammar (PCCG). An example for illustrating the use of DCG in semantic parsing is given in figure 1.3.

To represent the sentence 'Kiambu county borders Nairobi', the sentence can be represented using define clause grammar which can be viewed as context free grammar written and interpreted through first order logic notation as:

Relation: Verb [λyλx.borders(x,y)]→ borders

Object 1: NP[Kiambu] → Kiambu

Object 2: NP[Nairobi]→ Nairobi

Fig. 1.3 Example Illustrating the Use of DCG in Semantic Parsing

These three rules will be triggered upon the system identifying the lexical entries 'border/s', 'Kiambu' and 'Nairobi' within the sentence tokens. One objective of semantic parsing is to compose the meaning of larger fragments from their parts (Domingos & Poon, 2009). Rules for doing this would appear as shown in figure 1.4.

VP[rel(obj)]→Verb[rel] NP[obj]

S[rel(obj)] →NP[obj] VP[rel]

Fig. 1.4 Rules for Composing Meaning of Larger Fragments from their Parts

The first rule would fire upon seeing relationship 'borders' and either 'Kiambu' or 'Nairobi' in the sentence to give the meaning "'Kiambu or Nairobi' 'borders' 'another town' ". Upon seeing the second object the second rule fires giving the meaning 'Kiambu borders Nairobi' or 'Nairobi borders Kiambu'.

These rules and lexical entries were traditionally manually constructed from text. However research into supervised and unsupervised learning approaches has been on the rise especially for applications that are intended to process text from the web.

Other semantic parsing systems work in a similar manner with major variations being in the meaning representation language and the extent of labeling in supervised learning. Different supervised machine learning algorithms for semantic parsing were proposed in Zettlemoyer and Collins, (2005); Mooney, (2007) among others. In particular Zettlemoyer and Collins (2005) introduced an approach of learning to map sentences to a logical form through the use of structured classification with probabilistic categorial grammars (see section 2.3.1 for a detailed explanation of the learning task). A categorial grammar is a phrase structure grammar and represents words using categories. A category can be combined with another category to produce a new category because it behaves like a function which can take an argument either from its left or right side neighboring category in a sentence. Categorial grammars can also be combined with λ (lambda) calculus which is normally used to represent computable functions. Since semantics is often represented using functions, λ calculus is therefore used to represent semantic expressions (Nakorn, 2009). This method provides a rapid means of mapping NL text to logical form which if required can then be converted to SQL or other database formal languages. As Domingos and Poon (2011) notes, providing the target logical form for each sentence is costly and difficult to do consistently and with high quality, thus making supervised approach less attractive.

An unsupervised approach has been applied to information extraction which is considered as shallow semantic parsing (Banko, 2009). Unsupervised learning approach has also been applied to semantic role labeling (SRL) which is also considered as a shallow semantic task. SRL is concerned with identification of predicates or verbs in a sentence and determining all the objects associated with it. These objects are then assigned to semantic groups which are predetermined, in supervised leaning (Jurafsky & Gildea, 2002), or clusters in unsupervised learning (Swier & Stevenson, 2004). Recent search for an unsupervised semantic parser has been proposed in Domingos and Poon (2011). In the proposed approach dependency trees are converted to quasi logic forms namely the lambda-forms which are recursively clustered into various semantic variations. In yet another attempt to unsupervised semantic parsing reported in Mingxia, Jiming, Ning and Furong, (2007), the problem is

reformulated as an optimization problem. The following quote from their report underlines their basic approach:

> *"The basic ideas underlying our method can be stated as follows: first we translate the tokens of a question as well as their syntactical and semantic relations (as in NLP) into constrained question variables and functions, and thereafter, we utilize an optimization-based assigning mechanism to substitute the question variables with the corresponding constructs in OWL knowledge bases."* (Mingxia, Jiming, Ning, & Furong, 2007)

The solution to the problem of mapping question tokens automatically to OWL (Web Ontology Language given the acronym OWL) elements in web-based QA system is an important step towards providing answers from a structured source. Inquiry into the solution for this problem when the source is a relational database forms the core of research reported in this thesis.

As noted in Danica et al. (2009) it is not trivial to translate successfully parsed question into the relevant logical representation or a formal query which will lead to the correct answer and none of the developed solutions is a tight solution to this quandary. To sum up weaknesses of machine learning approach Popescu et al. (2003) note,

> *"…… to parse questions posed to a particular database, the parser has to be trained on a corpus of questions specific to that database. Otherwise, many of the parser's decisions will be incorrect. ….. On the other hand, manually creating and labeling a massive corpus of questions for each database is prohibitively expensive."*
>
> (Popescu, Etzioni, & Kautz, 2003)

The above challenge could be viewed as a domain adaptation problem, which is also known as transfer learning or cross-domain learning. Attempts to address similar challenges have been made in other fields of machine learning. For example in computer vision, the domain of interest (target) may contain very few or no labeled data while an existing auxiliary domain (source) may contain many labeled examples. Domain adaptation algorithms, such as SVM based algorithms have been proposed to bootstrap the target domain. Whereas these methods could be adopted for the relational database problem, such attempts have not been reported. The challenge thus remains unsurmounted to date.

A new school of thought has been developing alongside machine learning efforts. This has sprung up after years of research into ontology development mainly for representing information on the

semantic web. Research in automatic NL information entry and access to semantic web has been on-going. Some prominent ground breaking works have been reported in AquaLog (Lopez, Pasin, & Motta, 2004) at Open University in UK; Querix (Esther, Abraham, & Renato, 2006) at University of Zurich; NLP Reduce (Kaufmann, Berstein, & Fischer, 2007); QuestIO (Tablan, Damljanovic, & Bontchev, 2008) at University of Sheffield and FREyA (Damljanovic, Agatonovic, & Cunningham, 2010). In these works free NL text is parsed into concepts which are mapped onto mentions of ontology resources. Studies into question understanding for purposes of database access must be solved if such an approach was adopted for solving NLADB problem.

When casual users interact with systems, it is not the case that they concentrate on the grammatical accuracy of their inputs (Muchemi L. , 2008). Consequently, suitable algorithms must handle issues of ungrammatical texts, sentence fragments and short queries. Tablan et al.,(2008) observe that due to the popularity of search engines such as Google, people have come to prefer search interfaces which offer a single text input field where people usually type in short fragments often ungrammatically arranged (Tablan, Damljanovic, & Bontchev, 2008). This requirement inevitably limits the applicability of machine learning techniques to real life applications. Semantic and statistical based machine learning systems as well as rule based systems expect grammatical sentences that can be syntactically and semantically parsed while the users provide short fragments that are not guaranteed to be grammatical. This necessitates research on an approach that is robust enough to handle this challenge. The languages selected as case studies for this research were English and Kiswahili. A preliminary survey carried out to study the use of Kiswahili as a query language reveals that most databases' metadata is a concatenation of words or abbreviations in English. This poses an unresolved issue of cross-lingual analysis of the NLADB problem which introduces a cross-lingual aspect of NLADB problem that must be addressed for Kiswahili.

This thesis tackles the challenges presented in the foregoing section with a special focus on English and Kiswahili as the medium for querying.

1.2 Problem Statement

The problem of NL processing for database access which has remained an unresolved issue forms the main problem addressed in this work. As described in the foregoing section the specific challenges include lack of a suitable language and domain independent methodology for

understanding un-restrained NL text. The challenge of developing a generalizable methodology that maps any given natural language to a suitable structured query language is the main issue tackled. Most databases have to grapple with the challenge of cross-lingual interaction. The *cross-lingual* aspect arises from the observation that most systems which use Kiswahili as a media of querying predominantly use concatenation of words or abbreviations in English as databases' metadata.

Further the problem of mapping ontology concepts (formed from the underlying relational database) to parsed NL text remains largely unstudied. Previous studies have tended to concentrate on web text sources or pre-populated ontologies. This challenge was addressed alongside the task of parsing NL free text into concepts.

1.3 Objectives

The main objective of this research is to bring forth an architecture that facilitates natural language understanding of user queries and that helps build a structured language query that can be used to access highly structured information source such as a relational database.

The specific objectives are stated as follows:

- Develop a suitable language and domain independent methodology for understanding un-restrained NL text.
- Design an *architectural model* and algorithms thereof that facilitate access of data from databases using English and Kiswahili as case-study languages. Specifically algorithms for parsing free NL text and data structure for holding the parsed queries are to be designed. Further algorithms for extracting concepts from ontologies and matching functions are to be designed.
- Develop a *prototype* upon which *performance evaluations* can be done.

1.4 Significance of Research

This research involves design and development of language and domain independent architectural model that facilitates the understanding of un-restrained natural language text. Kiswahili and English are used as case study languages.

Successful solution to this problem significantly contributes to the body of knowledge within NLDBA field. This leads to better understanding of the problem and brings form methodologies that

developers could use. This research therefore postulates that successful solution to this knowledge gap will lead to novel methodology upon which natural language interfaces to databases can be built. Potentially this leads to the following application oriented social benefits:

- Access of data repositories within governments' and private sector databases by users who prefer use of natural language (casual database users).
- With increased usage of mobile devices there's more direct interaction with casual users hence a renewed interest in catering for their NL interaction as noted by Kauffman and Bernstein, (2007).
- Perhaps more significantly the solution is an important intermediate step in speech processing for voice access to databases. This has real potential of invigorating use of mobile phones for direct database access using NL.

1.5 Thesis Overview

The remaining sections of this thesis are organized as follows:

Chapter 2 provides a background to the QA problem. It explores the state-of-art techniques in NLQ processing for solving the QA problem specific to database access. This chapter provides an in-depth view of what it entails to perform deep structure semantic analysis of queries and how the concepts of natural language understanding can be realized in the context of databases. Specifically traditional approaches such as direct mappings to prolog and SQL through syntactic structure processing and interlingua approaches are examined. Machine learning techniques such as the supervised grammar based parsing and statistical approaches paradigms are also explored. The chapter explores deep into some of the related works that utilize deep structure analysis in semantic analysis of NL queries. In addition to this, concepts in resource description framework and implementation in OWL language are also highlighted. Further a review of related concept mapping methodologies especially for the semantic web is made. The chapter ends with a thorough analysis and design of a conceptual framework. In order to evaluate the models proposed in this work, concepts in accuracy, precision and recall measures as applied to NLQ processing are reviewed.

Chapter 3 provides an in-depth look at the approaches and resources selected at each stage of the proposed solution. The chapter presents results and analysis of surveys done for language queries

and database schema authorship. It provides an overview of the selected resources. Finally a comprehensive summary of all techniques and tools selected for this work are examined and justified

Chapter 4 presents evaluation techniques and tools adopted in this research. It also presents evaluation results of the proposed model as measured on the prototype. It also provides a comprehensive discussion on observations made.

Chapter 5 presents the major contributions and conclusions arising from this work as well as the implications to the research community and other stakeholders. Insights into areas that may be pursued for further research are also highlighted.

Chapter 2: LITERATURE REVIEW

2.0 Preamble

This chapter provides an overview of the important theoretical underpinnings of the work carried out in this research. The problem of accessing relational databases in response to users' queries is treated as a sub-problem of the more general question-answering (QA) problem. In addition to this sub-problem QA also involves tasks such as information retrieval from various sources (such as web texts and domain-specific ontologies such as in bioinformatics ontologies) as well as document retrieval where a document can be retrieved from a collection of documents. QA also encompasses design of dialog systems. The general problem of QA is first explored under which the main approaches are expounded. This is then followed by issues specific to QA specific to database access.

2.1 The QA problem

Question answering is the task of providing an answer to a question posed in NL text from an information source such as a document or a data repository. The document may be web-based or simply a text document while the data repository may be a relational database or a specialized knowledge base such as resource description framework ontology used in semantic web. The goal of QA using NL is to provide users with the ability to use their own terminology in an unrestricted manner and receive answers that are satisfactory. QA systems must have a degree of intelligence that enables formulation of answers from information repositories through synthesized natural language queries. The general QA problem is illustrated in Fig. 2.1 here below.

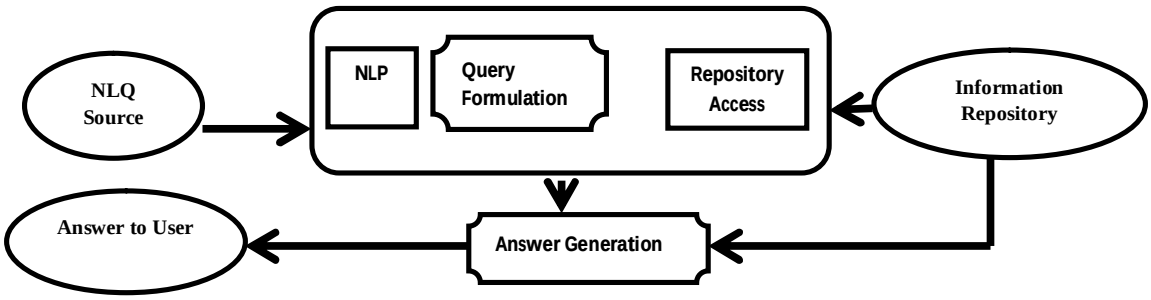


Figure 2.1 The General QA Problem

Natural language processing (NLP) involves tasks such as tokenization, morphological analysis, shallow or deep syntax analysis and semantic processing. Morphological analysis deals with the word structure. Shallow syntax processing involves syntax analysis targeted at the phrase level. On the other hand deep syntax processing will involve the grammaticality of the entire sentence. NLP for QA task also involves NL understanding (NLU). NLU refers to the process of comprehending the meaning of a user input and making intelligent usage of it once the semantics of the lexicon and phrases contained therein are known. Repository access refers to the extraction of meaning-bearing elements from the data source, say ontology, and representing them in a formalism that makes it easier for inferencing. Answer generation includes all those processes that combine the elements of NLP and data source processing and provide answers from the repository to the user. Key cross-cutting research areas in the QA problem revolve around the design of better NLP parsing and semantic analysis algorithms, more intelligent answer generation schemes and more efficient information representation formalisms.

2.1.1 Introdution to Database Access Task

This is a specialized task of the general QA problem. It specializes in information access from relational databases through natural language querying. The input may be free NL or controlled NL. Different approaches to solving this problem have been applied and two major schools of thought have emerged dominant. These are the traditional logic based mapping and the use of machine learning.

The methodology of parsing the NL and subsequent formation of a solution from the information source is determined by several factors namely:

- If the NL is controlled or unrestrained (Smart, 2008),
- Structure of information source (Lopez et al. (2007)and
- Preferred approach which may be, either traditional predicate logic mapping as in Androutsopoulos et al., (1995), Mingxia et al. (2007) among others; machine learning QA such as in Zettlemoyer and Collins, (2005); Mooney, (2007), Domingos and Poon, (2011) among others; or semantic QA approaches as reported in Lopez et al. (2007) among others.

An overview of the QA for database access dominant parsing methods applied is shown in figure 2.2.

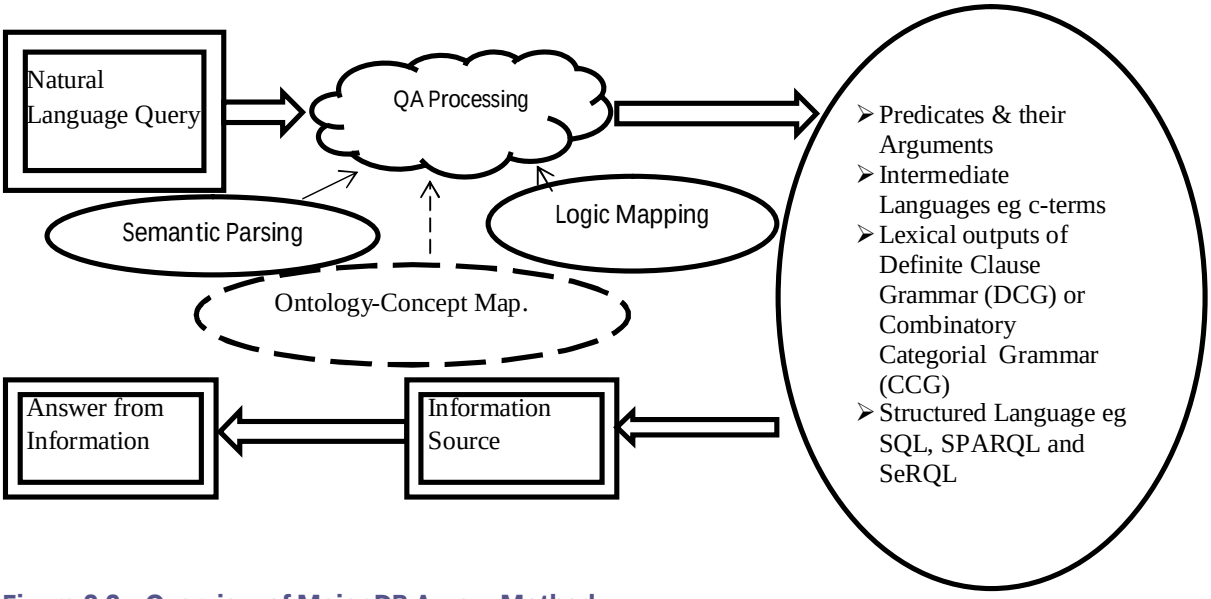


Figure 2.2 Overview of Major DB Access Methods

2.1.2 Challenges in Database Access Task

QA for database access task presents different challenges from the general QA problem in several ways highlighted here below. This task involves highly structured information repository and therefore calls for more stringent processing techniques for the NLQs. A highly structured source, such as a relational database, is normally accessed by highly formal languages, such as SQL, SPARQL and SeRQL and therefore NLQ processing must yield to these highly constrained formalisms. The implication is that NLQ processing must result in concepts that match the database schema as opposed to a set of references that may contain the answer. Answers given must be exact and not subject to probability of accuracy. This calls for improved NLQ representation and inferencing formalisms.

NL access to database task also differs from NL access to free texts and general ontologies in those databases present strict formalisms not present in the other sources for example special conditions such as foreign keys between data. Foreign keys ensure data integrity in relational databases. This integrity enhancement mechanism ought to be inbuilt in any envisaged access method. This leads to the question of whether methods that can handle these formalities can be designed.

Many language processing tasks, such as machine translation, have been more successfully solved through machine learning techniques. From a machine learning perspective, the database problem is a two-stage classification problem. The NL is converted to an intermediate internal meaning

representation (MR) as a first stage and then inturn mapped to a structured query. This back-to-back classifier arrangement inherently reduces the performance of machine learning solutions. Another challenge in the NL database access task is the inherent limitation of the databases that have poor domain adaptability. To parse questions posed to a particular database, the parser has to be trained on a corpus of questions specific to that database. Creating and labeling massive corpus of questions for each database is prohibitively expensive. Researchers are still grappling with this problem despite the use of the state-of-the-art annotation-cost reduction methods such as semi-supervised learning. Although the use of machine learning techniques in this sub-problem is still a fascinating theme for researchers in this area none of the proposed methods adequately addresses the training issue.

Another challenge is the development of language independent methodologies which do not heavily rely on language specific tools but rather some universal tools applicable across languages. State-of-art approaches which include semantic-parse based and logic-mapping based methods involve grammar manipulations. For example semantic-parse based methods use definite clause grammar (DCG) or combinatory categorial grammar (CCG) to produce intermediate elements, the meaning representations. An example of this is demonstrated in Thomson, et al. (1997), Zettlemoyer & Collins, (2005) among others. On the other hand logic-mapping based approaches such as syntactic-based mapping inherently use grammar manipulations in the formation of the intermediate phrase trees. Examples of these architectures are demonstrated in Garcia et al. (2008), Popescu et al. (2003) among others. This dependency on grammar renders these methods language dependent. The realization of a language independent method would provide a universal access method across languages.

It has been observed that communities who use multiple languages, such as the use Kiswahili as business language and English or French as official languages in East Africa also encounter another type of challenge. Database authors use concatenations or abbreviations of the official language as database schema language while ordinary persons prefer using business language to query these databases. This presents a new dimension to this task, the problem of ‘cross-linguality’. Cross-linguality refers to the phenomenon of using a given language to query a database whose schema is authored in a different language. The abbreviations and concatenations of words forming the object and field names are done in a different language from the one used to query. This problem is

prominent in countries with multi-language policy. For example sub-Saharan countries are affected by this phenomenon with East Africa having a cross-lingual issue of Kiswahili and English.

Some other cross-cutting issues that are subject to intense research in this field include portability which means the ease of porting from one database to another and from one domain to another with the ultimate goal being the need to minimize or eradicate the requirement for manual customization and re-crafting of code when porting across domains or databases. Finally a challenge lies in the synthesis of NL queries because it involves natural language understanding which is a non-trivial task and the selection of appropriate query language that can be efficiently parsed and mapped onto the data repository. The choice is usually between whether to use controlled natural language or un-restrained language. These are next discussed in section 2.2.

2.2 Controlled NL (CNL) versus Unrestrained Text

While designing QA architectural models it is important to consider the nature of input queries. The primary goal of QA is to provide users with ability to query in an restrained manner, however some researchers have noted some inherent complexities of natural language that are not easy to computationally solve. These include ambiguity brought by anaphoric references, semantic ambiguity and lexical ambiguities. These are respectively illustrated in the following sentences,

Jane invited Susan but she told her she was late for work …. Anaphoric ambiguity

Bill kissed his wife, and so did Chris. (Did Chris kiss Bill's wife or his own?)... semantic ambiguity

The mouse was in my house … lexical ambiguity

To overcome some of these difficulties some QA researchers have proposed the use of controlled natural language (CNL). CNLs are subsets of natural language whose grammars and dictionaries have been restricted in order to reduce or eliminate both ambiguity and complexity. According to Smart, (2008) the concept of CNL was first introduced in the 1930s by linguists who sought to create a 'minimal' variety of English that would be accessible to non-native English speakers. This concept has been adopted by QA researchers especially those pursuing semantic QA approach. These include CLoNE (Funk et al. 2007), Rabbit (Hart et al. 2008) among others. Although CNL has been in research labs for over a decade now, it has failed to take the lead because of various reasons. First CNL as a language must be learnt and understood by users, a task that is daunting for casual users. Some researchers have overcome this by creating an interface that dynamically generates suggested

words (from CNL lexicon) in the input dialog boxes as is the case with GINSENG (Guided Input Natural Language Search Engine) (Bernstein, Kaufmann, & Kiefer, 2006). The challenge here is that CNL introduces another layer of processing which introduces errors and thereby reducing the overall performance of the system. More over CNL in itself cannot replace ontology layer and therefore it is only a superfluous effort with little gain. It is for these reasons that this research opts to work with unrestrained text as opposed to CNLs.

2.3 Related Works

This section discusses in detail past efforts by researchers in trying to solve the Natural Language Access to Data-Base (NLADB) problem. The section is divided into three schools of thought who have developed architectures that are useful in NLADB revolving around well-established theories in closely related areas. These related areas include semantic parsing, logic mapping and ontology concept mapping. The review presented here focuses on those efforts that lead to solution of the NLADB problem or other QA architectures.

2.3.1 Semantic Parsing

Semantic parsing refers to the transformation of a natural language sentence into its meaning representation. It is distinct from other tasks such as semantic role labeling and information extraction in that it aims at transforming NL into computer executable form as opposed to the former which deliver human readable outputs. Different types of meaning representation languages are used but variations of first order predicate logic are prevalent. Meaning representation languages are designed by the creators of an application to suit the application's needs and are independent of natural language (Kate & Wong, 2010). For example the sentence, 'Which rivers run through the states bordering Mississippi?' can be answered by the machine readable meaning representation "answer(traverse(next_to(stateid('mississippi')))" as derived from GeoQuery (Kate & Mooney, 2010). Earlier systems used manually generated semantic parse representatations but manually authoring and tuning a semantic grammar for each new database is brittle and prohibitively expensive. This has resulted in active research in the use of machine learning and statistical methods in generation of meaning representations or grammars that would do this.

i) Statistical Semantic Parsing

Statistical semantic parsing is understood to mean finding the most likely meaning M_0, given a string of input words W and a discourse history H. The task of a statistical language understanding system

is therefore to search among the many possible discourse-dependent meanings M_d for the most likely meaning M_0 (Miller, Stallard, Bobrow, & Swartrtz, 1996):

$$M_0 = \text{argmax}_{M_d} P(M_d | W, H).$$

This model is recast interms of *pre-discourse meaning M_{pd}, syntax parse tree T, discourse history H, and a given list of words W as*

$$M_0 = \text{argmax}_{M_d} \{\max_{M_{pd}, T} [P(M_d | H, M_{pd}) P(M_{pd}, T) P(W | T)]\}$$

This model can easily be integrated with syntactic and discourse statistical models as reported in Miller, (1996) as shown in figure 2.3

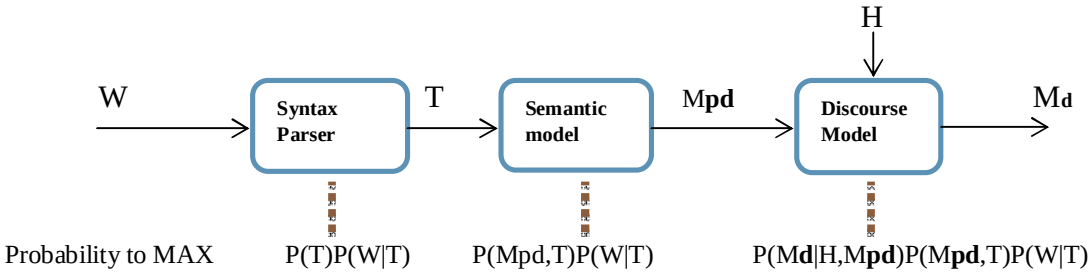


Fig. 2.3 Integration of Syntactic, Semantic and Discourse Statistical Models.

ii) Grammar-based Semantic Parsing

Semantic parsing can also be expressed as a grammar-based machine learning problem and is cast as shown in figure 2.4

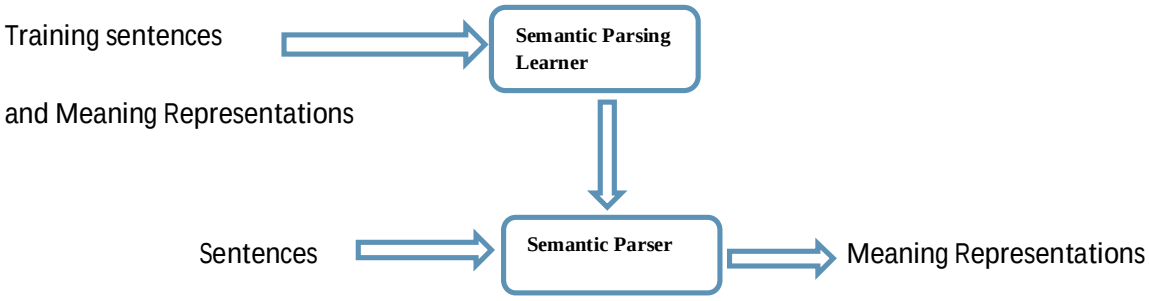


Fig. 2.4 Machine Learning problem in Semantic Parsing.

The most preferred grammars are the definite clause grammar (DCG) and probabilistic combinatory categorial grammar (PCCG) because meaning representatations can easily be expressed as first or

higher order logic elements. Grammar rules are used to combine various elements to build larger elements with known semantics. Higher order logic deals with functions and is expressed using the λ operator. For example the function *gender(x)* can return 'female' or 'male' and can be expressed as *λx gender(x)*. Similarly a function involving two arguments x and y such as x^2-y can be written as λx,y x^2-y using the lambda operator. If the operator is called using argument (2,3) the function is written as λx,y x^2-y(2,3) and returns 2^2-3=1. This is very important in expressing predicates (verbs) that express relations and arguments (noun phrases) that represent objects. In Definite Clause Grammar (DCG) for example, 'Mary loves John' becomes

Relation: Verb [λyλx.loves(x,y)]→ loves ; Object 1: NP[Mary] → Mary; Object 2: NP[John]→ John

This can be combined using the following rules

VP[rel(obj)]→Verb[rel] NP[obj] ; S[rel(obj)] →NP[obj] VP[rel]

As the semantic parser processes the tokens from a text, it would recognize verbs such as loves as valid predicates and noun objects such as John and Mary as valid arguments. If there are no other restricting rules the parser would recognize that John loves Mary and Mary loves John. The semantic parse learner takes pairs of tokenized sentences and their meaning representations and learns to map them on to each other. The tuned parser can then be used to parse new sentences and obtain meaning representations.

A recent entry in the semantic parsing grammar theory is the combinatory categorial grammar (CCG) (Hockenmaier & Steedman, 2002) and (Hockenmaier & Steedman, 2007) which represents words using categories unlike in the context free grammar which defines the structure using a set of rules (Hockenmaier & Steedman, 2007). CCG is a lexicalized grammar and uses chart parsing technique. It uses two categories namely primitive (S, N, NP) and complex categories. A complex category is a combination of two categories with directionality, which is either forward or backward (Nakorn, 2009). In CCG slash and backslash are used for representing directionality. A/B is the category which takes B as an argument on its right. Therefore, A/B B results in the category A. A\B is the category which takes B as an argument on its left. Therefore, B A\B results in the category A. For example the sentence 'Atieno loves Kamau' has a semantic parse tree shown in figure 2.5

Atieno	loves	Kamau	
NP	(S\NP)/NP	NP	>
.	S\NP		<
	S		

Fig. 2.5 Semantic Parsing Using Combinatory Categorial Grammar (CCG)

CCG generates the lexicon Atieno → NP; loves → (S\NP)/NP; Kamau → NP and equivalent lambda forms atieno, (λx.λy.borders(y, x), and kamau respectively upon seeing the respective tokens. Training data therefore consists of sentences and their meanings in lambda form and equivalent meaning derivations, d. A semantic learner's primary goal is to estimate feature weights. A feature $f_i(L,S,T)$ is the number of times a lexical item *i* is used in the parse T that maps from sentence S to logical form L. Figure 2.6 illustrates this concept of a CCG semantic parser using feature weight estimation for future sentence parsing. The feature weight adjustment is usually error-driven, much like perceptron back-propagation delta minimization of a neural network.

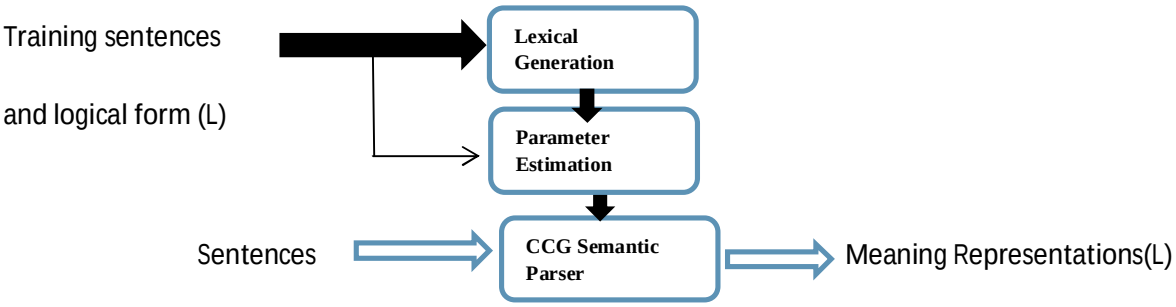


Fig. 2.6 Learning Probabilistic CCG (Zettlemoyer & Collins, 2005); (Kate & Wong, 2010).

The probability of obtaining the logical form L and meaning-derivation tree, T as trained on the sentence S, P (L,T|S) is thus maximized.

Development of new concepts in semantic parsing has been the main pre-occupation of most research in the area of the QA problem. Several applications of these concepts have been made with the most prominent being CHILL which is a supervised learner (Zelle & Mooney, 1996), *SCISSOR* which is a statistical semantic parser that integrates syntax and semantics (Ge & Mooney, 2005), *WASP* a statistical parser (Wong, 2005), *PCCG-based parser* (Zettlemoyer & Collins, 2005), *KRISP*

a supervised learner using kernels (Kate & Mooney, 2006) and a semi-supervised learning using support vector machine (Kate & Mooney, 2007) all tested on the querying of *Geobase*[1]. Performance has improved over the time with the higher precisions being recorded with fewer training examples. The *Geobase* database is in prolog and contains USA geography with about 800 facts. CHILL uses DCG and after being trained on 150 sentences that are matched with equivalent logical forms, achieves approximately 56% accuracy in answering novel questions compared to the semi-supervised learner of Kate and Mooney, (2007) which attains approximately 75%. As seen from above review the tendency is to move towards unsupervised learning as reported in Domingos and Poon, (2009).

While as these initiatives towards NL processing for information access are great, their application is limited when it comes to accessing highly structured information sources such as relational databases. Access to database task differs from access to free texts and general ontologies in that any approach for databases access must grapple with the issue of strict formalisms not present in the other sources. An example of such formalism is constraints such as foreign key which ensure data integrity in relational databases. While this challenge is not unique to machine learning methods other challenges are specific to this approach.

One such hindrance to use of machine learning in accessing databases is the need of the two learning processes namely sentence-to-meaning representation (MR) learning and MR-to-SQL learning to be superimposed on each other. The two processes must be placed in series as illustrated in figure 1.2 and this greatly reduces the accuracy limiting the applicability of this method. Further the conversion of logic meaning representations to SQL is a problem that is far from being understood. Not much efforts have been directed to this area of research mainly because question answering from the web, (a task that usually does not require MR-SQL conversion) has become ubiquitous therefore obsecuring the attention or need for research into conversion of MRs to SQL. However the challenging task of accessing information from structured sources such as relational databases has never been adequately solved.

Another great hinderance to the use of machine learning methods in automatic conversion of NL to SQL is the need for suitable training datasets comprising of pairs of NL and SQL for every database.

[1] *Geobase* was initially supplied with reference guide of Turbo Prolog 2.0 Borland International (1988)

This greatly hampers portability. If a rapid training method can be developed and used to train datasets and then the trained classifier applied to a database as a 'plug-in', this would be a great step forward in solving portability issue in machine learning approach. One such attempt is the design of a methodology for development corpora for automatically learning to map natural language questions into SQL queries (Giordani & Moschitti, 2010). Here a corpora containing matching pairs of NL and SQL queries in the form of syntactic trees and that contains both correct and incorrect training sets is developed. The paper only reports on the corpora development process and shows direction as to how the corpora would be used to train an SVM classifier which would in turn be used to rank pairs of new queries and existing SQL queries. The classifier would select from all possible SQL statements from the database and return the highest ranking pair. This method would be great if portability is not a key consideration for the particular application. Initial costs are high because creating a new corpora containing all possible SQL and NL statement is prohibitively expensive and tedious exercise. But perhaps the greatest draw back to the robust use of this method is need for human intervention in the creation of semantic-based clusters of NL-SQL query pairs. Giordani and Moschitti, (2010) note that 'clustering is performed semi-automatically'. Since mapping is done at the syntactic level, resource scarce languages such as that being studied in this research would suffer a practical drawback due to the requirement of efficient syntactic parsing.

Popescu et al. (2004) while opting for mapping as opposed to machine learning approach noted that machine learning has some inherent challenges such as the need for re-training during porting and the problem of creating NLQ-SQL pairs for every new database. This is quoted thus,

> *"However, attempting to use a statistical parser in a database-independent NLI leads to a quandary. On the one hand, to parse questions posed to a particular database, the parser has to be trained on a corpus of questions specific to that database. Otherwise, many of the parser's decisions will be incorrect. ….. On the other hand, manually creating and labeling a massive corpus of questions for each database is prohibitively expensive."*
>
> (Popescu, Armanasu, Etzioni, Ko, & Yates, 2004)

This observation remains true up to date despite the few efforts in corpus creation and labeling technique such that by Giordani and Moschitti, (2010) discussed above. Obtaining the training set

containing NL questions that map to SQL for every database is probably even a greater practical challenge than obtaining more efficient learners. This then begs the question of what is the way forward. Would a non-machine learning-based solution be a better option?

2.3.2 Logic Mapping

Analysis from literature reveals two dominant schools of thought in terms of model design. These are basically syntactic-based processing and token-matching processing. The two approaches are illustrated in figure 2.7.

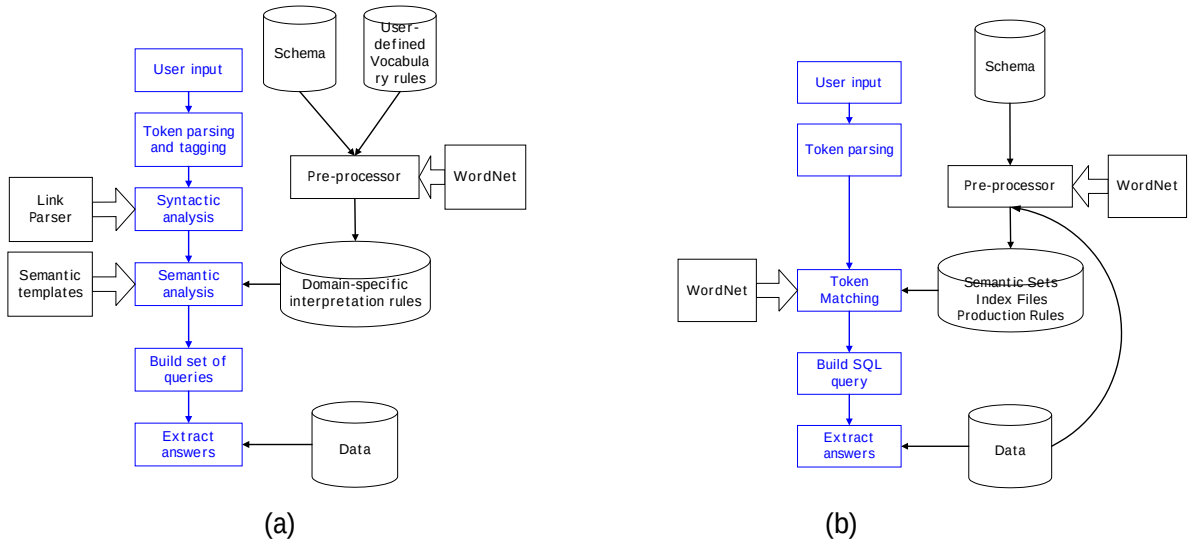


Fig. 2.7 (a) Syntactic-based Parsing and (b) Token-matching Parsing (Minker, 1997)

Older generation systems relied on syntactic-based parsing which requires both syntactic and semantic analysis. Syntactic parsers are well developed for many languages especially for use by other NLP problems such as automatic translation. On the other hand semantic analysis is in most cases implemented as semantic tags. However due to reliance on syntactic information for the entire sentence, systems developed from these models tend to perform poorer than those from token-match parsing because users tend to use short phrases usually agrammatical. The method encounters difficulties in parsing ungrammatical sentences.

Some of the recent prominent works widely quoted in literature such as PRECISE (Popescu, Etzioni, & Kautz, 2003) and the largely successful system developed by Dittenbach and Berger, (2003) for

accessing tourism data on *Tiscover*[2] database (Dittenbach & Berger, 2003) are based on token-match parsing. This means that a query is broken down into its constituent parts and all syntactic marker-words stripped. The tokens which are usually nouns are matched to the names of databases, tables or instances through a suitable algorithm.

An attempt for token parsing for Kiswahili is reported in Muchemi, (2008). This method relies on both tokens and partial syntactic parsing coupled with semantic tagging and uses structured query templates. It is modeled as a template matching problem. This approach is similar to the one introduced by Popescu et al. (2003) which uses a noun-based token parsing method but models it as a graph matching problem and uses the max-flow algorithm. The system by Popescu et al. (2003) had an F-score of 0.65. The Kiswahili noun phrase-based token matching had a comparable rate of success averaging 0.64 on F-score. The method encounters difficulties in parsing ungrammatical sentences because of the templates usage though. From this research its concluded that incorporating both tokens and phrases for mapping purposes improves the results. In other research reported in Muchemi and Narin'yan, (2007), the following conclusion was arrived at,

> *"When analyzing a NL text input, it is necessary to use its lexical semantics within the subject domain to reconstruct its probable meaning. Only if this meaning has several variants then it would be useful (as local as possible) to turn to syntactic aspects of the text to resolve this ambiguity*." (Muchemi & Narin'yani, 2007).

This conclusion and that arrived at in the Kiswahili noun-phrase based mapper (Muchemi, 2008) are essential in the sense that syntactic information should be used for improving precision and recall in SQL formation processes.

Logic based approach has been the dominant approach for many years and yielded to restricted narrow domain applications. This has culminated in commercially available programs such as English-Wizard[3] and English Query[4] which have thus far been discontinued for varied reasons. Some of the state-of-art applications employing token matching approach in commercial use today include Siri[5] by Apple, Quiri[6] by Easy-Ask group and Watson[7] by IBM. Virtual strategy magazine[8]

[2] *Tiscover is the largest Austrian web-based database for tourism*
[3] English Wizard is by Easy-Ask can be accessed at http://www.pcmag.com/encyclopedia_term/0,1237,t=English+Wizard&i=42616,00.asp
[4] English Query by Microsoft can be accessed at http://msdn.microsoft.com/en-us/library/aa198281%28v=sql.80%29.aspx
[5] Siri is a voice enabled phone functions control software and details can be accessed at http://www.apple.com/iphone/features/siri.html
[6] Quiri is a voice and NL text enabled desktop and mobile application that processes NL to access corporate data. It can be accessed at http://www.easyask.com/products/quiri/

reports that “EasyAsk has long been a leader in natural language information analysis and delivery software. Firms such as Coldwater Creek, Lands’ End, Lillian Vernon, Aramark, TruValue, Siemens, Hartford Hospital, Ceridian, JoAnn Fabrics and Harbor Freight Tools rely on the EasyAsk software products to run their business and e-commerce operations daily.” A closer look at these systems reveals that they rely on three-layer architecture that has a speech-to-text analyzer, a grammar analyzer, and a set of service providers. A comprehensive comparison of these three leading software (EasyAsk, 2010) shows that Quiri provides speech recognition and interfaces with corporate data while Siri connects with mobile phone functions via voice command. IBM describes Watson as a "computer system that can understand natural language and deliver a single, precise answer to a question. Upon closer examination Watson combines natural language processing, complex algorithms that choose the best answer from the available options, and a large scale smart analytic system designed to feed potential answers to the questions.” (EasyAsk, 2010).

In all these, the grammatical analysis performed here involve searching a string for certain key words and using those words to build up a simple model of what the user wants to do and what is to be done (Jeff, 2011). The success of these systems depends on the scope of focus of the domain area. For example Jeff, 2011 observes that, “Siri’s limited focus on appointments, contacts, messages, and maps makes this technically viable”.

The above token based approaches differ in the way the tokens are mapped to the underlying database. While direct maping of key words to database, table and column names has produced only moderate rates of success, this research aims at exploring improved database schema processing and information representation.

Relational databases NL access problem has mainly been tackled through logic mapping (Shin & Chu, 1998). Some successiful applications have been implemented through mapping of phrases. A representative sample is the natural language interface to the largest Austrian web-based tourism platform Tiscover (Dittenbach & Berger, 2003). In this approach language processing invoves identification of language, spell checking, phrase detection and tagging. This is followed by SQL query formation and eventually the generation of results. In this approach noun phrases and synonyms in the NL query are identified. A light weight grammar is applied so that all possible

[7] Watson is a desktop application that processes NL text for accessing corporate data. It can be accessed at http://www-03.ibm.com/innovation/us/watson/index.html
[8] Virtual Strategy Magazine can be accessed via http://www.virtual-strategy.com/

modifications on terms (through introduction of prepositions, adverbial or adjectival structures) can be identified before tagging occurs. The introduction of grammar makes this approach language dependent because different languages behave differently. Tagging of terms in a query with relevant predefined classes is necessary so that each term can be labeled with the relevant concept tag with the domain. For example 'hotel' is labeled with 'accomodation' and 'sauna' with 'facility'. Tagging provides semantic interpretation however it inevitably introduces errors similar to those in semantic labeling such as poor classification. Manual tagging is costly and not easy when when working with many domains. Due to these issues among others research has advanced in search of improved architectures.

A more recent approach has been reported by Nokia Research Centre Cambridge (Ran & Lencevicius, 2012) where they solve the problem of accessing information stored in RDF repositories targeted to mobile phones users. These works along with ground breaking works reported in AquaLog (Lopez, Pasin, & Motta, AquaLog: An Ontology-Portable Question Answering System for the Semantic Web, 2004) at Open University in UK; Querix (Esther, Abraham, & Renato, 2006) at Univ. Of Zurich; NLP Reduce (Kaufmann, Berstein, & Fischer, 2007); QuestIO (Tablan, Damljanovic, & Bontchev, 2008) at University of Sheffield and FREyA (Damljanovic, Agatonovic, & Cunningham, 2010) among others are extensively reviewed and analyzed in the subsequent section of this chapter under the heading related works in an effort of developing a novel architecture that facilitates access of data from relational databases with cross-lingual problem in this case Kiswahili-English.

2.3.3 Ontology-based Approach to DB Access

An ontology is framework that represents knowledge in form of concepts within a domain. The relationship between the concepts must be represented using a fixed set of syntax and semantic rules. Most computing efforts for representing knowledge have shifted significantly from logic based knowledge representation schemes to ontology-based schemes. This is evidenced by much interest and funding for the semantic web activities. The ubiquity of the world wide-web and pervasiveness of internet has brought impetous to semantic web research. This is also attested by the large number of conferences organized around semantic web activities. Subsequently this has led to development of web ontology languages mainly based on the *de facto* standard resource description framework

(RDF). OWL 2 XML format advocated by the W3C and developed by OWL working group[9] is the dominant while RDF/Turtle, RDF/XML syntax and Manchester Syntax languages are also used to a lesser extent.

RDF based ontologies represent elements as attributes or as resources. These elements are equivalent to resource names, labels, comments and string property values which are usually mapped to concepts within natural language sentences in an ontology concepts mapping process. Figure 2.8 illustrates this concept.

```
<rdf:RDF xml:base="http://www.xxx.org/periodictable/PeriodicTable">
   <owl:Ontology rdf:about="">
       <owl:versionInfo
        ….
       </owl:versionInfo>
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
Periodic Table of  the Elements
</rdfs:comment>
<……/>
</owl:Ontology>

  <owl:Class rdf:ID="Group">
    <owl:DatatypeProperty rdf:ID="name"/>
    <owl:onProperty rdf:resource="#number"/>
    <owl:onProperty rdf:resource="#element"/>
     ……..

    <color rdf:datatype="http://www.w3.org/2001/XMLSchema#string">silvery lustrous
grey</color>
  </owl:Class>

</rdf:RDF>
```

Fig. 2.8 An Example of OWL based RDF Resource

This example shows an RDF resource with a single ontology. The ontology contains a class called *group* which has a *name, number, elements* etc. It can also be seen that the ontology is defined as an RDF resource with a unique unique resource identifier (URI). The ontology contains many elements such as comments, resources and datatypes as illustrated in figure 2.8.

Ontologies have been used in many natural language processing tasks. Common tasks include information retrieval and extraction, question answering, machine translation and summarization among others (Buitelaar & Ciamiano, 2006). Of particular interest to this research is question-answering models. The tasks may either be question analysis, answer selection or ontology based question answering by mapping. Question analysis deals with ontology-based semantic

[9] http://www.w3.org/2007/OWL/wiki/OWL_Working_Group

interpretation such as WordNet (Miller G. , 1995) while answer selection deals with ontology-based reasoning for answer-type checking. The task of data-base access that is addressed in this research is closely related to the question-answering architectures by mapping described in Lopez et al. 2007. The successes in this area are described in section 2.4.

2.4 Successes and Shortcomings in Ontology-based NL Access

Reasonably accurate representation of data from relational databases to ontologies has been reported widely and applications utilizing these techniques developed. A notable one is reported in Wu et al. (2007) in which a semantic-based search and query system for the traditional Chinese medicine community has been reported. Automatic discovery of mappings between ontology and RDBMS has been successful and a typical state-of-art approach is found in Hu and Qu (2008). Tools for converting databases to ontologies have been developed with notable ones being 'Datamaster' (Csongor, Martin, & Samson, 2009) and 'Datagenie' (Gennari, Nguyen, & Silberfein, 2007). In these tools a table is mapped to an ontology class, a column to a datatype property while a row is mapped to an instance of the ontology. Further it is observed that if a relational database table has foreign key references to other tables, these are replaced by instance pointers when the database is converted into an ontology.

Research into the use of NL to access semantic web ontologies has been active and with modest levels of success. Pioneer systems include AquaLog (Vanessa et al., 2004), Querix (Esther et al. 2006), NLP Reduce (Kaufmann, et al. 2007), PANTO (Wang, Xiong, Zhou, & Yu, 2007), QuestIO (Tablan et al., 2008) and FREyA (Damljanovic, et al. 2010) among others. In all these works NL free text is parsed into concepts which are mapped onto mentions of ontology resources.

The two processes described above, that is conversion of relational databases to ontologies and processing of natural language into concepts are illustrated (as blocked arrows) in figure 2.9.

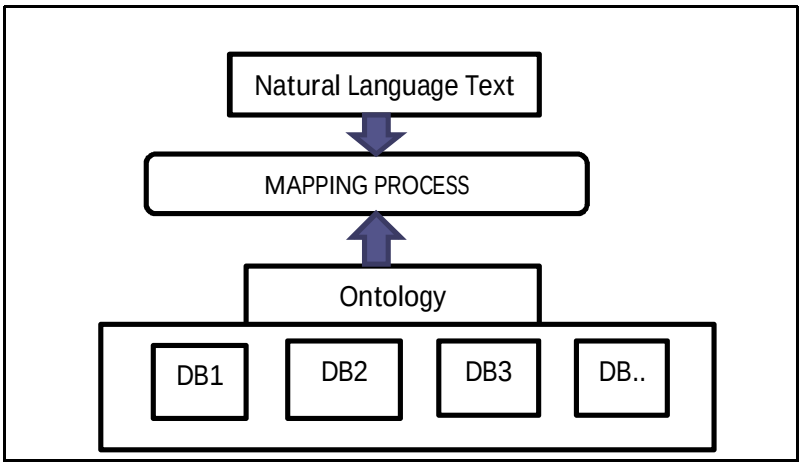


Fig. 2.9 Overview of Ontology-based DB Access Task

Considerable research efforts in this area have been driven by the need for developing ontologies as a means for accessing heterogeneous data sources (Zorzi, Tessaris, & Dongilli, 2007). In this case ontologies have been viewed as an extra data representation layer that provides shared conceptualizations and acts as a mediator to the underlying data layer. This research concentrates on relational database and the scope does not entail heterogeneous sources.

Research in the area of ontology-based access to databases has concentrated in domain specific applications with the driving force being the need for providing common taxonomies, merging different ontologies within the same domain and querying formalisms. Danica et al. (2009) observes that although many natural language interfaces to ontologies have been developed, those that have reasonable performance are domain-specific and tend to require extensive customisation for each new domain. Many disciplines have developed standard ontologies with standard features that domain experts use to share information exclusively in their fields raising issues of portability across domains. For example the Gene Ontology project provides a controlled vocabulary of terms for describing gene product characteristics and gene product annotation data from GO Consortium members, as well as tools to access and process this data (Gene Ontology Consortium, 2001). It provides a standard nomenclature to terms that has three parts namely prefix ‘GO’, a unique zero-padded seven digit identifier called the term accession number and a unique term name. Two ontology concepts are linked by a relationship. For example ‘GO:0031966 : mitochondrial

membrane' *part of* 'GO:0005740 : mitochondrial envelope' has the link '*part-of*'. This ontology's nomenclature and structure is unique to it and not easily reusable in a different field say medical field with an example of an equivalent ontology described in Munir et al., (2008).

If an ontology is created directly from a relational database, the elements ported into the ontology are mainly the table names, column names and the data within rows. These names are usually short forms, concatenations, acronyms and abbreviations which do not have a standard naming style. This makes it difficult to decode and map onto the underlying concepts. This is one such challenge addressed in this work. At the heart of this problem is the challenge of bringing forth an intervening layer that sits on the ontology (shown in figure 2.10 as Data Layer) that maps labels to the underlying concepts implied in the ontology. This requires a study into the naming styles of databases elements and implementing algorithms that provide this mapping. An appropriate research question here would be whether there exists a finite set of patterns that authors of database schema use in representing database schema object names and whether appropriate processing algorithms are feasible. A portion of this work attempts to answer these questions. Arising from above discussions this work seeks to contribute in the area of domain independent concepts discovery from ontologies created from relational databases.

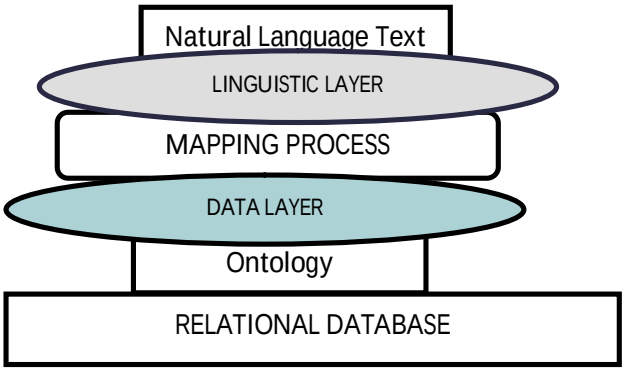


Fig. 2.10 Research Shortcomings In Ontology-based DB Access Task

Another challenge encountered by ontology-based DB access method low recall due to dependence on nouns and nominal phrases for concept identification only yet it is well established that concepts within a domain go beyond these types (Krishnamurthy & Mitchell, 2011). Further there is often language-dependence on processing methods. For example PANTO Wang et al., (2007) uses language dependent-syntactic processing to provide parse trees that enhance recall. Methods such as

those described in QuestIO (Tablan, Damljanovic, & Bontchev, 2008), AquaLog (Lopez, Pasin, & Motta, 2004),NLP-Reduce (Kaufmann, Berstein, & Fischer, 2007), Ontology-Assisted Query Reformulation (Munir, Odeh, & McClatchey, 2008) systems among others rely on identification of nouns and noun phrases mostly involving proper nouns as concepts. Concepts are however more diverse that this as identified by Krishnamurthy & Mitchell, (2011). Studies for Kiswahili language have also revealed diverse patterns of term formations (Sewangi, 2001). Terms represent concepts and therefore should be accounted for in concepts discovery process. Further more nouns identification should include such noun patterns as identified Ohly, (1982) such as inclusion of the following categories,

- Norminalized verbs eg undungaji sindano (needle injection),
- Deverbative head with noun complement eg kiweka dawa (medicine insertor)
- Combination of nouns eg haidrojeni perokisidi (Hydrogen peroxide)
- Noun and adjective eg kuku wanene (big chicken)
- Nouns with –a connector eg mavi ya kuku (chicken's dung)
- Noun with –a connector and a nominalized verbs eg sindano ya kutungia (needle for piercing)

This diversity of concepts formation ought to be captured in a manner that ensures language independence. These linguistic processing concepts are studied and abstracted into a linguistic layer indicated in figure 2.10. Therefore this study investigates through the use of language independent linguistic theories and seeks to contribute in the area of language independent concepts formation process in the OCM approach.

Different concepts-discovery techniques have been developed, the most basic being noun strings matching. Discovery of implicit concepts within a query is much harder task. Most of these techniques are usually designed for the particular ontologies. For example in Munir et al., (2008), techniques that interpret ontology-based search results and associated domain knowledge reformulate a relational query so as to assist users and their applications in formulating queries without requiring complete knowledge of the information structure of underlying data sources. To illustrate this technique an example is provided below from Munir et al. (2008),

> "E.g. interpreting the query '*Give me all MRI scan images of brains for children with an Astrocytoma Tumour disease in a specific age group'*. This query cannot be fully resolved by the HeC data model because there is no direct information available in the databases that matches with the term

> 'Astrocytoma Tumour'. Here the query reformulation system receives a simple input into the system as 'Astrocytoma Tumour', the system then extracts all of the clinical tests and related values that confirms the possibility of Astrocytoma Tumour disease in the brain." (Munir, Odeh, & McClatchey, 2008)

Another technique used to resolve discovery of implicit concepts is the use of hypernyms. Hypernyms are superordinates or words that are more generic. For example animal is a hypernym of chicken therefore if a person interrogates for a chicken, the characteristics of animal also apply.

As can be observed some techniques of concept discovery are domain specific while others are generalizable across domains and languages. One goal of this research is to bring forth language and domain independent methodology and this requires design of techniques applicable to relational databases and that can be generalized across domain. The respective natural language processing techniques for discovering explicit and implicit concepts are studied in this work and abstracted to the respective layers as shown in figure 2.10.

A prime motivation behind this research is the quest for accessing data from databases using Kiswahili text. Kiswahili has over 150 million regular speakers in Kenya, Tanzania, Uganda, Rwanda, Burundi, Parts of DRC, Malawi and Somalia. But even with this nearly all databases are developed in a specific nation's official language which is predominantly English or French. The reason for this can be attributed to the fact that these official languages also double as the primary training languages and therefore database developers tend to favour their usage in database schema development. On the other hand, casual users tend to prefer using local languages or Kiswahili, the business languages among the communities. Thus a practical solution ought to have a cross-lingual solution capacity to cater for these differing language usages. It is for this reason that studies in this area are undertaken.

In summary though closely related to conventional semantic ontologies and other specialized fields' ontologies such as gene ontology, relational database ontologies differ due to their manipulation requirements as explained above. Further relational databases store data from all domains and also store in a language independent manner, thereby motivating the move towards domain-independent ontology assisted access to relational databases through any given natural language.

2.5 Trends in Reviewed Approaches

An analysis of literature from section 2.3 reveals a definable general trend of application of methodologies to the QA task illustrated in figure 2.11

The figure highlights the trend of preferred approaches to QA processing depending on the degree of structuredness of the data source. It has been observed that highly unstructured sources such as web pages, text documents and other similar sources tend to favor semantic parsing and semantic role labeling. Methods applied range from highly annotated machine learning techniques (Mooney, 2007) to purely unsupervised techniques such as that by Domingos and Poon, (2009). Semantic role labeling has previously been used in QA problem as reported in various works such as in Jurafsky and Gildea, (2002). The current trend is to move from semantic role labeling to supervised machine learning but current efforts are pushing towards unsupervised learning methods (Domingos & Poon, 2009). Another school of thought is logic based mapping where research has been done on phrase-based and token-based methods. Examples include the graph-matching approach by Popescu, Etzioni, and Kautz (2003), token-based approach for Kiswahili (Muchemi L. , 2008) and tiscover's English NL interface (Dittenbach & Berger, 2003) among others. Template mapping and graph-based mapping have shown better results as opposed to syntactic based approaches (Muchemi & Narin'yani, 2007). Structured entry sources such as yellow pages and information on templates such as hotels, universities and airports services rely on named entity as main method of information extraction (IE) task. The IE task is however extended to include QA abilities. Recent efforts have been directed to the creation of ontologies (Munir, Odeh, & McClatchey, 2008). The power of ontologies lies in their capacity to provide context for semantics. In specialized fields such as bioinformatics, QA queries are processed from domain specific ontologies such as the GO gene ontology ((Ontology, 2012); (Gene Ontology Consortium, 2001)). Ontologies enable semantic description of data and in inference.

From this perspective it can be deduced that the direction to which a generalizable natural language database access solution should be sought is in the area of ontology concept mapping.

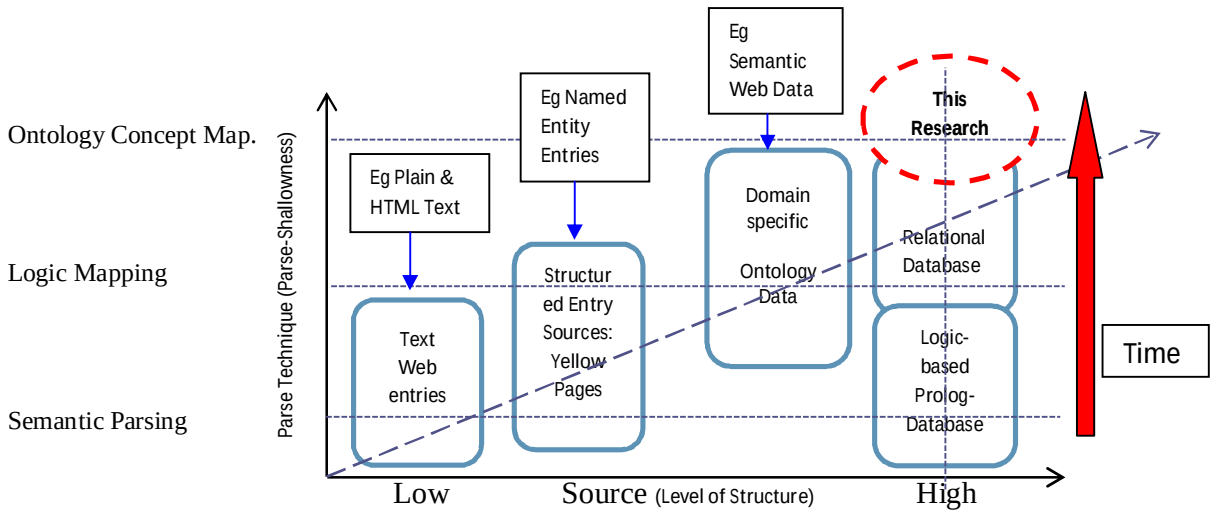


Fig. 2.11 Methodologies of QA Processing Depending on Source

This research aims at making a contribution in the area marked 'This Research' which envisages a domain and language independent ontology concept mapping architecture for natural language access to relational databases. The sections that follow critically review, analyze and evaluate processes and algorithms applicable to the mapping architecture. An assessment of the suitability of these processes and proposals for modification to suit relational database problem is also provided.

2.6 Towards Domain and Language Independent OCM Approach to Database Access

Not much research has been reported towards the design of domain and language independent approaches in ontology assisted natural language access to relational database. As analysed from the literature above there are major gaps or potential improvement areas in the OCM approach for relational database access. This research therefore dedicated its efforts towards development of concepts, models and algorithms required for realization of a language and domain independent approach for database access using natural language. The state-of-the-art for all components and modules required in the OCM approach as conceptualized in this research is reviewed, with a view of identifying the best practice and gaps that would be addressed by the research.

2.6.1 Conceptual Framework

The conceptual model shown in figure 2.12 is modelled on the generic QA model discussed in section 2.1. It however clearly separates processing and representation methods for the NLQ and database schema and highlights the central roles the matching and SPaRQL generation functions play.

In the envisaged model, a natural language query is processed through operations such as normalization, tokenization, lemmatization, stemming and part of speech tagging in the module labelled 'NLQ Processing' in figure 2.12. This is further followed by phrase formation, chunking, collocation and terms discovery in the same module. The resulting elements of natural query processing should be stored in a proposed specially designed schema, referred to in this research as a feature space model (FSM), which holds the elements in a manner that is usable for mapping purposes. The FSM is found within the module labelled 'NLQ Representation' in figure 2.12

On the other end schema processing is carried out through conversion of the relational database into an ontology through a mapping process where tables, columns and row values are mapped onto ontologies classes, data type properties and instances of the ontology respectively. The mapping process occurs in the module labelled 'Schema Processing'.

In this work it is proposed that an additional processing layer be added to this module and this would cater for the need for understanding the usually abbreviated or concatenated object names as well as field names. The products of this additional process would be stored in a proposed gazetteer within the module labelled in figure 2.12 as 'schema representation'. This work seeks to make further contributions in the mapping process indicated in figure 2.12 as 'matching function'. It is envisaged that methods can be devised to unearth implicit concepts within a query and within an ontology. This helps improve recall and accuracy values. For instance descriptions found on database objects such as row descriptions can be mapped to the ontology as labels and assist in implicit concepts discovery.

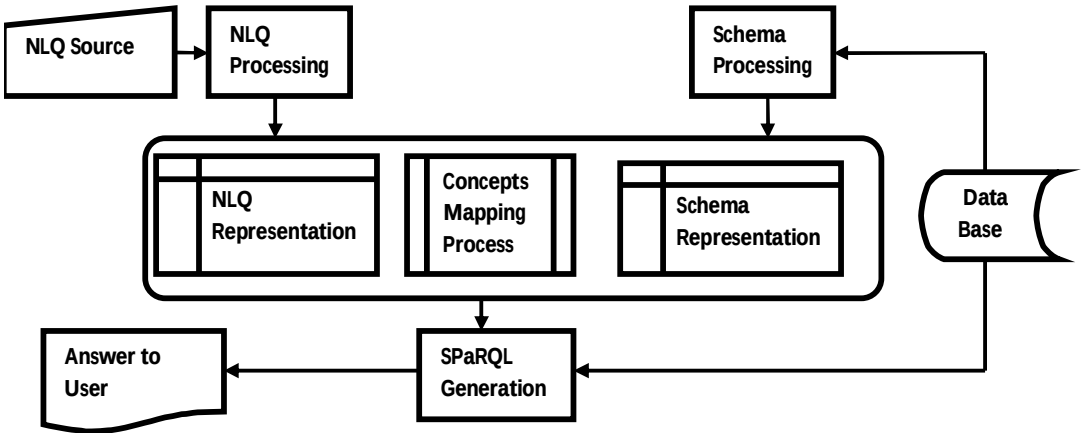


Fig. 2.12 The OCM Conceptual Model

Once the mapping process is complete, the process of structured query generation starts. This is carried out by the module labeled 'SPaRQL Generation Function'. SPaRQL is generated via a series of functions. The generated SPaRQL query is applied to an ontology with the help of an ontology reasoner (such as protégé reasoner[10]) and the desired answer generated.

The sections that follow provide a detailed component description of the elements in the conceptual model.

2.6.2 NLQ Processing Task

A natural language query is received from a source such as a text editor and is delivered to the NLQ processor. The NLQ processor is envisaged to have several modules which perform distinct roles. For instance most reviewed systems such as AquaLog (Vanessa et al., 2004), Querix (Esther et al. 2006), NLP Reduce (Kaufmann, et al. 2007), PANTO (Wang, Xiong, Zhou, & Yu, 2007), QuestIO (Tablan et al., 2008) and FREyA (Damljanovic, et al. 2010) among others have a normalizer and a tokenizer whose objective is to standardize all input texts and prepare them for further processing.

[10] A plug-in Reasoner for protégé tool For example The Pellet Reasoner Plug-in, version 1.0, makes Pellet 2 available in Protégé 4,

The next step involves NLQ processing. The main objective is to obtain tokens that represent concepts that would have correspondences in the ontology. In many systems such as NLP Reduce, Querix and QuestIO this task is reduced to identification of nouns and therefore the normalized token has to undergo part of speech labeling. In other systems syntactic knowledge has been used to achieve a variety of things within this NLQ processing module. For example FREyA (an acronym for 'Feedback, Refinement and Extended Vocabulary Aggregation') combines syntactic parsing with the knowledge encoded in ontologies in order to reduce the customization effort during porting from one domain to the other (Damljanovic et al. 2010). It achieves this through enhancing user interaction. On the other hand some systems such as PANTO use language dependent-syntactic processing to provide parse trees that enhance recall. As pointed out in 2.4 this work differs from reported works highlighted in the literature as it seeks to find processing methods that are language independent and that are geared towards matching ontologies generated from relational databases whose processing requirement is different from standard nomenclature ontologies.

In order to investigate this aspect there is need to apply theories of language that are universal. Some of the applicable linguistic theories include x-bar theory (Chomsky, 1970), transformational and generative theories (Chomsky, Syntactic Structures, 1957) among others. The underlying theory preferred in linguistic analysis of queries is Transformational theory advanced in Chomsky (1957). Transformational theory is preferred in this study because it has previously been used for query formulation process in MULDER (Kwok, Etzioni, & Weld, 2001), a question-answering system. In this theory it is stipulated that a sentence has a deep structure form (DSF) which can be transformed through transformation rules into several surface forms. For example the DSF of sentence 'close door' can be transformed into the surface structure forms 'close the door', 'you close the door', 'the door should be closed by you' etc. Deep structure forms (DSF) versus surface structure forms (SSF) has been most studied and applied in showing equivalence of sentences.

The research questions for the analysis are formulated as,

1. *Can deep structure forms (DSF) of a query be used in deducing the interrogative properties of a query?*
2. *What types of relationships exist between DSF of queries and SPaRQL queries?* This involves discovering linguistic *patterns* that can be discovered and exploited in development of templates

that can be used in mapping natural language queries into structured query language specifically SPaRQL?

3. *Does the choice of NL affect the answers obtained from the above two questions?*

As envisaged in the conceptual model NLQs are normalized, tokenized, lemmatized, stemmed and tagged with parts of speech. This is further followed by phrase formation, chunking, collocation and terms discovery in the same module. A challenge arises in the design of a schema that can hold these elements in a generic domain and language independent manner. In the systems reviewed such as PANTO (Wang et al., 2007), QuestIo (Tablan, 2008) among others, the products of NLP are mainly nouns and nominal phrases and subsequent processing is limited to these. In a system that expands the collection of what constitute concepts a more elaborate schema is expected. The design of that envisaged schema also referred to hereafter as the feature space model (FSM) forms a component of the research undertaken in this work and is described in section 3.5.

2.6.3 Schema Processing and Information Representation

Research geared towards methods and tools for successful automatic conversion of relational databases into ontologies has been intense resulting to several successful models. Wu et al. (2007) in their paper for a semantic-based search for the traditional Chinese medicine community have presented methodologies of how ontologies can be linked into databases (Wu, Chen, Cui, & Yin, 2007). Hu and Qu (2008) on their paper 'Discovery of mappings between ontology and RDBMS' have provided strong theoretical backgrounds to the process of conversion of relational tables to ontology elements (Hu & Qu, 2008).

Information from the database is mapped onto ontology constructs as shown in the example illustrated in figure 2.13. Concepts from the ontology are identified and if they match those in the NLQ they are selected as potential constructs of the subsequent SPaRQL query. Identifying concepts from a domain specific ontology is easy because the lexicon of the ontology is controlled and only string matching would be required. However in relational databases there is no controlled vocabulary in writing table and column names and therefore the resulting ontology will not have a controlled lexicon. Subsequently a challenge is encountered in the decoding of schema information specifically tables' and fields' names. The text used to label these names is usually a concatenation, acronyms or abbreviations. This challenge has not been tackled in the reviewed literature and therefore it is proposed for further research within this study. It is proposed that schema processing should be

extended from the current state of the art which involves basic mapping as explained in section 2.4 to also include identifying the various components within the labels, deducing or guessing their meaning and assigning them to specific database concepts. This challenge of parsing a database schema into suitable ontology concepts organized into a suitable formalism remains under-studied and is a key component addressed in this research.

A. Database (Ontology) Definition - OntologyMyNorthwind

```
<rdf:RDF xmlns="http://www.owl-ontologies.com/OntologyMyNorthwind.owl#"
   xml:base="http://www.owl-ontologies.com/OntologyMyNorthwind.owl"
   xmlns:dbs="http://www.dbs.cs.uni-duesseldorf.de/RDF/relational.owl#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
```

B. Table (Relation/Class) definition - employees

```
<owl:Class rdf:about="&db;employees">
    <db:hasPrimaryKeyFields rdf:datatype="&xsd;string">EmployeeID</db:hasPrimaryKeyFields>
    <db:isBridgeTable rdf:datatype="&xsd;boolean">false</db:isBridgeTable>
  </owl:Class>
```

C. Columns (Properties) Definition - FirstName

```
<owl:DatatypeProperty rdf:about="&db;employees.FirstName">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:domain rdf:resource="&db;employees"/>
    <db:hasOrigColumnName rdf:datatype="&xsd;string">FirstName</db:hasOrigColumnName>
    <rdfs:range rdf:resource="&xsd;string"/>
  </owl:DatatypeProperty>
```

D. Row Values (Instances) Definition - Lawrence

```
<db:employees rdf:about="&db;employees_Instance_1">
    <db:employees.EmployeeID rdf:datatype="&xsd;int">1</db:employees.EmployeeID>
    <db:employees.FirstName rdf:datatype="&xsd;string">Lawrence</db:employees.FirstName>
    <db:employees.HireDate rdf:datatype="&xsd;date">2010-04-18</db:employees.HireDate>
    .....
  </db:employees>
```

Fig. 2.13 Example to Illustrate Schema Information Representation

Another challenge that needs to be surmounted for database access problem is the design of a suitable schema for holding information on concepts extracted from ontology of a particular relational database. If a database is large this can be a real problem due to requirements of searching

time and memory (Munir et al., 2008). In a related task such as text information extraction for question answering, Danica et al. (2009), defines an ontology-based gazetteer named 'OntoRoot' for the system FREyA (Feedback, Refinement and Extended Vocabulary Aggregation) (Damljanovic, 2010). Tablan et al. (2008) presents the Questio architecture which incorporates a dynamic gazetteer named 'Ontology Resource Root Gazetteer' which contains all set of lemmas which are generated from text as triples containing concepts. In PANTO nominal phrases in the parse trees are extracted as pairs to form an intermediate representation called 'Query Triples'. Then, by utilizing knowledge in the ontology, PANTO maps 'Query Triples' to 'Ontology Triples' during run time and this takes considerable time to compute. This research then sought to establish the structure of a schema (or gazetteer) that is suitable for database problem. A dynamic gazetteer model in line with that developed by for Questio (Tablan, Damljanovic, & Bontchev, 2008) was selected because it allows dynamic holding of information and also allows holding of minimally processed (lemmas from ontology) concepts from the ontology.

2.6.4 The Matching Function

2.6.4.1 Overview of the Mapping Problem

Once the schemata structures of FSM and gazetteer are determined, the next process is to design the mapping function with a goal of enhancing accuracy and recall of the NL to SPaRQL mapping. NLQ and ontology are processed and the retrieved tokens stored in appropriate schemata. This process is illustrated in figure 2.14

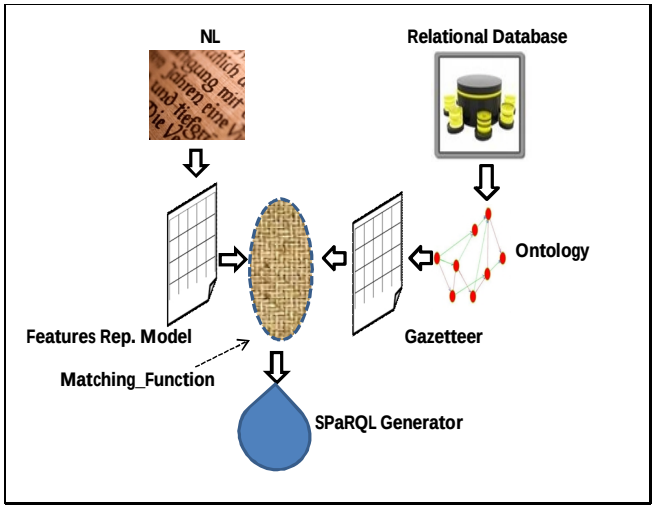


Fig. 2.14 Matching Function in OCM Approach

The concepts that match from the two representation schemata form the backbone of the generated SPaRQL query.

The challenge that arises from the mapping problem is that equivalent concepts in NLQ and ontology elements may be represented by different strings and therefore concept matching goes beyond simple string matching. Moreover accuracy and recall of the NL to SPaRQL mapping process must be enhanced. Strategies that map the concepts from NLQ to those from ontology must be formulated.

The strategies that inform design of these mapping algorithms are discussed next.

2.6.4.2 Strategies from Ontology Matching Models

By far the most preferred method of identifying the matching concepts is a simple lexical-level, keyword-based matching method with lemmatization. In this strategy each word in the FSM is matched against every word in the gazetteer through minimizing Levenshtein distance (number of operations needed to transform one string into the other, where an operation is an insertion, deletion, or substitution of a single character). To improve results stemming is carried out to ensure that all words derived from the same root map onto that root regardless of the prefixes and suffixes. Research is active in this area with researchers trying different models in order to improve on recall

and accuracy level. Research efforts in this area are motivated by models in closely related areas such as ontology matching and document retrieval techniques. Ontology matching, also known as ontology alignment, is the process of determining correspondences between ontologies. It establishes similarities in concepts and relationships in different ontologies within the same domain and is an important step for integrating overlapping domains of knowledge. Ontology matching strategies fall under six categories namely lexical-based, semantic-based, constraint-based, instance-based, structure-based and graph-based matching **(Keshavarz & Lee, 2012)**. The first four strategies operate at the concept level and may be extended to apply to the mapping problem under investigation. One such effort is by Gao et al. (2007) in which they report a constraint-based method for semantic mapping from natural language questions to OWL. They model the matching function as constraint satisfaction problem. They summarize their approach as follows:

> "…… we have proposed the basic ideas and formulation of a constraint-based method for semantic mapping from a natural language question to the elements in OWL. In this method, we first decompose questions into a set of variables by means of syntactical and semantic analysis, and then formulate their underlying constraints, e.g., associated knowledge, into different quantitative functions. Thereafter, we can make use of an optimization-based objective function to find sound substitutes in the OWL knowledge representation for the question variables."
>
> (Gao, Liu, Zhong, & Chen, 2007)

Although Gao et al. (2007) report that this constraint-based method improves precision by 15% of the current preferred method (lexical-level, keyword-based matching method with lemmatization) the key limitation remains in the problem of simplification of an NLQ into a set of variables. Further the practicality of obtaining the function constraints and assigning them weights based on associated knowledge of the question is a setback to this strategy. This model is work in progress and Gao et al. (2007) have stated that more investigation needs to be carried out in-depth on how to represent associated knowledge and how to systematically derive and formulate the corresponding constraints for the purposes of question understanding and semantic mapping.

2.6.4.3 Strategies from Document Retrieval Models

Research efforts motivated by document retrieval strategies revolve around the four theoretical models namely Boolean model, Vector Space model, Probabilistic model, and Language model which are the dominant models **(Liddy, 2005)**. These models are summarized next.

a) The Boolean Model ((Salton, 1971) and (Salton G F. E., 1983))

The Boolean Model is a classical information/document retrieval model which is the oldest and the most widely used. It uses a set of words (d), referred to as indexing terms, which are derived from a document and can be present in a query or absent,(1 or 0). The model also uses a representation of a query (q) as a set and a similarity function $R(q,d_i)$. The query "Give me the names of employees living in Nairobi or Kampala" is formulated in well-formed formula (*wff*) as " employeeName $\wedge$ (city $\rightarrow$ Nairobi $\vee$ Kampala). Matching is achieved through binary matching function where only documents containing phrases with employeeName $\wedge$ (city $\rightarrow$ Nairobi) or employeeName $\wedge$ (city $\rightarrow$ Kampala) are considered relevant.

The most commonly cited matching function is given as:

$$\mathbf{R(q,d_i)} = \frac{\mathbf{2|d \cap q|}}{\mathbf{|d| + |q|}}$$

Where R is the similarity function, q is a set that represents words in a query, d is a set of words derived from an entire document and also referred to as indexing terms and d_i therefore an individual member of that set.

b) Vector Space model (Salton G W. A., 1975)

In this information/document retrieval model each term within a document is represented by its weight. The term weight is computed by considering how often it appears within the document (term frequency (tf)) and also the inverse of how often it appears in a collection (inverse document frequency (idf_t)). It is assumed that if a term appears too often within the collection, the less important it is. The weight of a term w(t, d) is computed by obtaining the product of tf and idf_t in what is called tf-idf_t .

$w(t, d) = tf\text{-}idf_t = tf \ X \ idf_t$

A document can then be represented by a vector of these weights for each term.

$V(d) = (w(t_1,d),w(t_2,d),...,w(t_n,d))$

These weights are then assembled into a vector. The vectors are normalized so that unequal length of vectors between different documents can be taken care of. The query terms and document terms are

represented using such normalized vectors and the similarity between the two vectors computed from the dot product of the two vectors. Thus **score(q,d) = *v*(q). *v*(d).** Matching is thus achieved by computing the highest score.

c) Probabilistic Model (Robertson S, 1976)**:**

A Document Retrieval System based on the Probabilistic Model calculates the probability of relevance for each term in a document based on that term's frequency in a set of known relevant documents. The probability of the relevance of the term against another set of non-relevant documents is also determined. The objective is to determine whether if a new document when subjected to probability check with respect to a query term, the probability will tend towards the relevant document's probability, if the query is relevant to the new document, or towards the probability of non-relevant document, if the new document is not relevant to a particular query. In the absence of initial relevance probabilities, a prior probability can be determined by counting the number of documents in which the term appears and the number of documents in which it does not appear. This initial estimate is then adjusted based on users' response to controlled experiments. This model assumes that probabilities are based on a binary condition of relevance and therefore the matching is binary.

d) Language Model (Ponte J, 1998)

In this type of modelling each document has its own language model. Language modelling is the task of estimating the probability distribution of linguistic units such as words and phrases in documents. The probability distribution itself is referred to as a language model. Language modelling involves generating the probability distribution for each document. Queries are thought of as being generated by a document language model. Ponte et al. (1998), state that they infer a language model for each document and rank the document according to the estimate of producing the query. Liddy, (2005) expounds this by stating that the operative question is: "Which document(s) would produce this query?" The documents in the collection are evaluated and ranked based on the probability of their language model generating that query. Language model predicts probability of query production and not probability of relevance of the document. It is therefore referred to as the query-likelihood retrieval model.

The above six models form a sound basis upon which one can study the best matching model between the gazetteer and the FSM in the context of OCM for database access. Although

information/document retrieval problem is different from database access problem significant parallels can be drawn between these two problems as shown in figure 2.15.

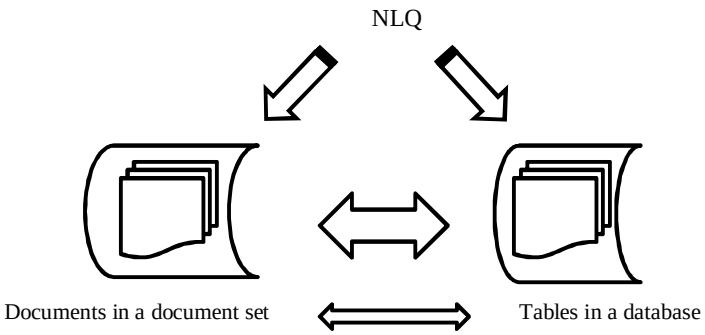


Fig. 2.15 Analogy of Document Retrieval Problem and Database Inform. Retrieval Problem

NL statements are used in both problems and the requirement for NLP is similar. Both require tokenization, morphological analysis and syntactic analysis. What is different is the source of information. Whereas the source in document retrieval problem is documents organized as document sets, the database problem is equivalent in that information is organized in tables grouped as databases. A collection of documents would be equivalent to a collection of tables (database) while an individual document would be equivalent to an individual table. The analogy is illustrated in figure 2.15.

Another similarity is in the information access goal where just like in the document retrieval problem, the goal of database access problem is to obtain relevant information from an already existing storage. For example to solve the document retrieval problem using the Boolean model, the degree of similarity between the words in the query and those in the document title is calculated. Only documents containing phrases that match with those in the query are considered relevant. If a match is found the document is retrieved. In an analogous manner, the degree of similarity between words in the query and those in a table's name are calculated. If a match is found between words in the table's title and the query's tokens, then that table is considered a candidate for selection and it forms a basis for the structured query in this case SPaRQL. For those retrieved tables further matching between the query tokens and the attributes (or row values) of the table is performed and the one where a match is found is selected and forms the structured query. As illustrated in figure 2.14.

With this analogy in mind one is compelled to look at the applicability and efficiency of these models. In the boolean, vector space, probabilistic and language models both the query and table's contents are represented as two sets of strings and similarity or relevance parameter calculated as earlier explained. From the tables' titles and the query set strings, the respective degree of similarity or relevance (depending on the model selected) is used to select the table. The column name is also selected in a similar manner that is, by calculating the similarity or relevance of a column to the query set. These models perform poorly when it comes to rows selection because for information in a particular row to be retrieved with a higher degree of precision, it is not the number of times the tokens within a query appear within a particular table or the degree of similarity as perceived in these models but it is the presence of both the table name and column name that would yield a more accurate answer. This then leads us to ask whether a lexical-level keyword matching would be a better model.

From the observation that recall and accuracy of NLQ-SPaRQL mapping models suffer due to different lexical representations of similar concepts at the NLQ and ontology level and that the lexical-level, keyword-based matching method that is the current state of the art does not adequately address this challenge, this research investigates enhancements to this algorithm. The algorithm is enhanced through techniques borrowed from ontology matching strategies specifically semantic-based strategy. Semantic matching strategy combines integration of lexicon-based matching and meaning of the words. Issues relating to performance of the mapping function are studied in this work too.

2.6.5 Structured Query (SPaRQL) Generation

As explained in section 2.6.4 and illustrated in figure 2.14, the matching function gives as its output a list of concepts which are present both in the query and in the ontology. The various concepts as generated by the matching function form a set of strings which is not ordered. The query generator's function is to organize these concepts into a structured query. The task therefore is to select the various strings (representing different concepts) and assemble them into a structured query.

The process of converting query- and ontology-representations into formal query language such as SQL, RDQL, SPaRQL or SERQL has been approached differently by various researchers. One such approach is to compile a grammar that is used in generation of queries. Bernstein et al. (2006) while presenting their system 'Querying the Semantic Web with Ginseng' which is a guided input natural

language search engine have developed 'Ginseng grammar' which describes both the parse rules of the English queries, as entered by a user, and the query composition elements of the RDQL queries (Bernstein, Kaufmann, & Kiefer, 2006). This grammar uses the BNF (Backus Normal Form) notation which represents the syntax for expressing context free grammar.

Another school of thought involves the use of 'interpretations' and 'transformers' as used in Tablan et al. (2008). Interpretations are containers that are used for holding information while transformers are algorithms for concatenating information held in a container with other strings generated by the matching function so that a new container the with compounded information is generated. The complexity of information is recursively incremented to more complex information that yields the longest possible structured query. The following quote from Tablan et al. (2008) demonstrates the spirit of this approach,

> *"At the beginning of the process, the list of candidate interpretations is initialised with a simple interpretation containing the input text and the annotations created in the previous steps. In each iteration, the interpretations currently in the candidate list are transformed into more detailed ones. …….. All candidate interpretations are scored according to a set of metrics …… and, in order to keep the number of alternatives under control, candidates that score too low can be eliminated"*
>
> (Tablan, Damljanovic, & Bontchev, 2008)

These approaches have their merits and demerits. For example the first approach requires authoring of grammar once the domain is changed or the language of usage is changed say from English to Kiswahili. This is because it is this grammar that controls how NL is going to be parsed and how they are going to be formed into a structured query. This method impedes the attainment of domain-independence which is one of the tenets of this research. As such this method is not a selected approach in this research.

The second approach requires the use of scoring matrix which requires scoring on chunk (similarity score), property (specificity score), and domain and range scores. Similarity score compares the lexical similarities and is calculated using the Levenshtein distance. It indicates the minimum number of operations needed to transform one string into the other, where an operation is an insertion, deletion, or substitution of a single character. Specificity score is determined by the distance of a property from its farthermost super-property. This means properties that are more specific score higher than their super-properties. Similarly domain and range score also referred to as distance score by Tablan et al. (2008) tries to infer an implicit specificity of a property based on the

level of classes that are used as its domain and range. An example to illustrate distance score calculation is shown in figure 2.16.

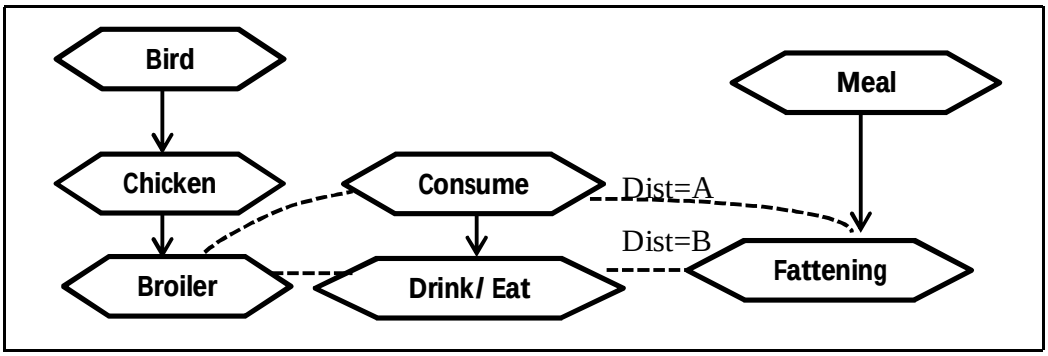


Fig. 2.16 Determination of Specifity Score

In this example the concepts triple 'Broiler eats fattening-food' scores more than 'Broiler consumes fattening-food' because the inverse of length indicated 'Dist=B' is greater than the inverse of that indicated 'Dist=A'. Hence according to specifity scoring the two statements score differently and are not semantically similar. Property score and domain-and-range score assume that properties, domains and ranges occur in a hierarchy within the ontology. In database access problem the score for the two distances 'A' and 'B' should be equal because statements such as 'Broiler eats fattening-food' and 'Broiler consumes fattening-food' are semantically similar. Hence while specifity scores are necessary in ontologies created for a specific domain such as the GATE ontology used in Tablan et al. (2008) experiments, the same is not true for relational database ontologies as shown in the example above.

From literature it is observed that if a relational database table has foreign key references to other tables, these are replaced by instance pointers when the database is converted into an ontology. However during the SPaRQL query generation a challenge was noted as this pointer information is hardly utilized during query generation. Foreign keys are important in ensuring data integrity in relational databases, therefore a mechanism for enforcing this integrity in the OCM approach need to be designed.

In summary the approaches used for the general QA problem are not readily applicable to the database problem as explained above. This therefore leaves us with a question that needs to be

answered and that is, 'what is the most appropriate query generation method for ontologies generated by relational databases?'

2.7 Evaluating Performance of the Architectural Model

Smart, (2008) has identified a criterion that is used to evaluate QA systems (Smart, 2008). The criterion may be used to evaluate performance as well as categorizing the systems. It has ten key areas assessed through the following questions,

1. **Support for different ontologies (Domain independence):** Can the system execute queries against various ontologies?
2. **Recall:** What percentage of the natural language queries entered by a user can be translated (correctly) into a corresponding semantic query?
3. **Precision:** How accurate is the system in terms of correctly translating the user query into the target query language? Does the system always retrieve the right kind of information requested by the user?
4. **Language Independence:** Degree of language independence. Is the architectural model affected by switching of language?
5. **Extent of NLP processing:** What kinds of NLP technologies are used as part of the query system?
6. **Extent of user interaction:** How much user interaction is permitted by the tool? Does the tool involve the user in resolving semantically ambiguous user input?
7. **Target query language:** What semantic query languages, e.g. SPARQL, are generated by the system?
8. **Usability:** How well does the tool perform in usability studies?
9. **Training requirements:** What is the training overhead associated with the tool? What kinds of users is the tool targeted towards, i.e. what is the target user community?

This research delivers an architectural model that forms a basis for developing a QA system for accessing a relational database, thus the focus is architecture-evaluation as opposed to the full system evaluation. It would be reasonable to review the above criterion and reduce it to have elements that are architecture-evaluation specific. These include the first four elements in the above criterion, namely domain independence, recall, precision and language independence. A review of experimental procedures for evaluating architectures' performance reveals that precision and recall

are the most favoured empirical measures for architecture evaluation. While evaluating 'Krisp' (Kate & Mooney, 2007), a 10-fold cross validation was used to measure performance in terms of precision and recall. Popescu et al. (2003) while evaluating 'PRECISE' also measured precision and recall. In addition they evaluated their architectural model using a parameter described as 'distribution drift'. This means testing using data that is totally different from what was used to train or tune the database. The motivation for this is that precision and recall dramatically fall when data changes with time from what was used to train. Since the drift described is related to deterioration of performance with change of data away from training data, this drift is renamed here to 'training drift'. Training drift only affects machine learning approaches and thus it is not considered in this work.

It is envisaged for adoption an evaluation framework that takes into consideration all architecture-related aspects derived from the Smart (2008) criterion that is, precision, recall, domain independence and language independence. In addition accuracy, F-score (F-score is harmonic mean of precision and recall) and support for cross-lingual databases would be useful additions to the evaluation criteria.

2.8 Summary

This chapter has highlighted the challenges that need to be addressed in order to solve the access to relational database problem using natural language. In particular it has been shown that an ontology concept mapping (OCM) approach can be used to solve this problem. A generic conceptual framework that handles most of the challenges encountered during processing such as language and domain dependence has been arrived at. It has also been shown that some important issues that need to be addressed by this architecture include capacity to handle the strict formalities and conditions set by relational databases, capacity to handle cross-lingual databases, language independence and high portability.

In order to actualize the conceptual framework into a concrete architecture and develop a prototype capable of giving indicative performance, some important algorithms need to be designed and developed. These include, parsing database schema information, matching algorithm within the OCM model, natural Language Query processing algorithm and SPaRQL generation algorithm. The design and implementation procedures of these components are detailed in chapter 3.

Chapter 3: OCM DESIGN METHODOLOGY

3.0 Preamble

Through synthesis of the literature in the previous chapter, it has been articulated what research has been done and what needs to be further done. The main methodologies and research techniques previously applied in each specific aspect of the problem have been reviewed and critically analyzed so that appropriate methods are applied. This chapter therefore provides the details of various research strategies and specific research actions geared towards the design of a language and domain independent ontology-based model for NL-based database access

A comprehensive analysis of methods and techniques for data gathering, experimental procedures, pre-design results and analysis are provided. The OCM design process, the architectural model and algorithms that facilitate the functioning of the model are presented in this chapter. Further the prototype development process and resources selected have been discussed in detail.

3.1 Overview of Issues to be Tackled

The OCM conceptual model was introduced and discussed in detail in section 2.6. This is a high level design that seeks to address the problem of natural language access to relational databases through an ontology-based approach. The problem has several sub-components that have been described in detail in chapter two and are re-illustrated in figure 3.1 for ease of reference.

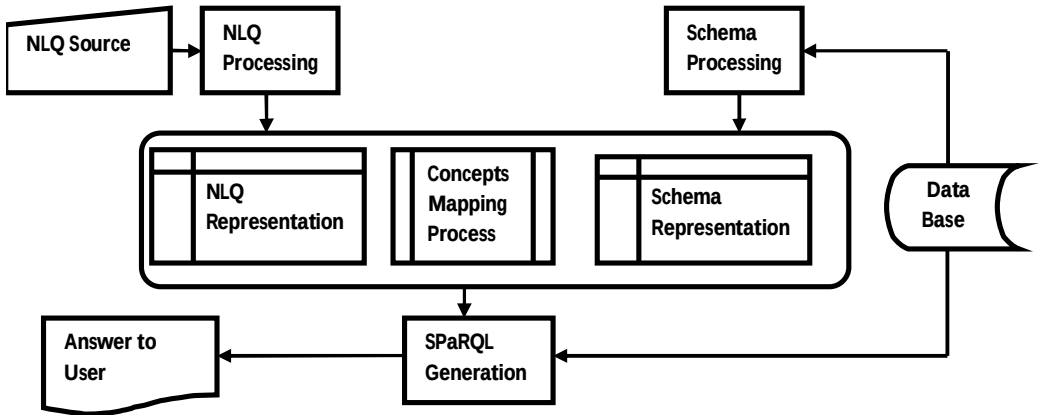


Fig. 3.1 Components of NL Access Problem Illustrated as a Conceptual Model

From the conceptual framework the process flow is abstracted and is illustrated in figure 3.2. As reviewed from literature, several tasks in the ontology-based relational database access problem are yet to be realized for a generic model.

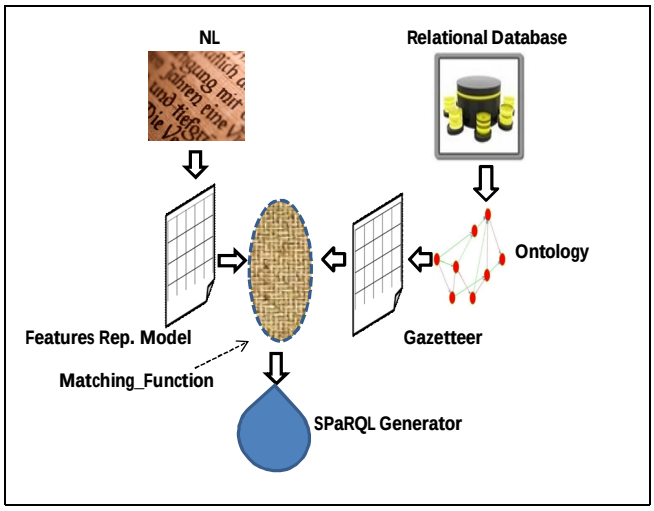


Fig. 3.2 The Solution Overview

The issues that need to be tackled before realization of the OCM model were analyzed in chapter 2 and are restated as follows,

1. Concepts Discovery
    a. There is need for extending the current state of the art procedures for explicit concepts discovery so that the techniques can be generalizable to any language. The use of universal language theories to tackle the language independence problem is explored.
    b. Current state-of-the-art methods rely on nouns and nominal phrases for concept identification. Concepts are however more diverse than this as identified by Krishnamurthy & Mitchell, (2011). There is need to expand the landscape to include other categories as identified in Sewangi (2001) and Ohly (1982) among others.
    c. In order to enhance the language understanding capacity of the model, there is need to design heuristics for implicit concepts discovery.
2. Decoding of ontology data and Semantic Tagging: Relational databases have no controlled vocabulary for naming tables and columns and therefore the derived ontologies would also

not have a controlled or a pre-determined lexicon. Subsequently a challenge is encountered in the decoding of database schema information specifically, names of tables and columns (fields) names. There is need for extending schema processing algorithms from the current state of the art which involves basic mapping as explained in section 2.4 to also include identification of the various components within the labels, deducing their meaning and assigning them to specific database concepts. This task is viewed as a semantic assignment task of the ontology data.

3. Schemata Designs: Another challenge that needs to be surmounted for the database access problem is the design of a suitable schema for holding information about concepts extracted from ontologies generated from relational databases. Further, in a system that expands the collection of what constitutes 'concepts' as envisaged in item 1 above, an elaborate schema for holding information from NLQ is to be designed too. The designs of these envisaged schemata form an aspect of the issues addressed in this work. The structures of these meaning representation schemes are critical because they influence the choice of the mapping algorithm. This research then sought to establish the generic structure of the two schemata suitable for the relational database problem.
4. Mapping Algorithm: There is need to enhance the lexical-level, keyword-based matching method that is the current state of the art because it does not adequately address recall challenges arising from different lexical representations of similar concepts at the NLQ and ontology levels (Punyakanok, Roth, & Yin, 2004). As analysed from literature, a methodology that borrows from ontology matching strategies, specifically the semantic-based strategy is adopted. The semantic matching strategy combines integration of lexicon-based matching with the meaning of the words.
5. Structured Query Generator Function: The various 'concepts' generated by the matching function form a set of unordered strings. The query generator's task is to organize these 'concepts' into a structured query. The design of this algorithm is a research problem tackled in this chapter.
6. Cross-lingual Access: Most NL database access problems have to grapple with the challenge of *cross-lingual* interaction. *Cross-linguality* refers to the phenomenon of using a given language to query a database whose schema is authored in a different language. The abbreviations and concatenations of words forming the object and field names are done in a

different language from the one used to query. Cross-lingual querying therefore occurs where the database definition language is not necessarily the one that is used for querying the database a challenge commonly found in multi-lingual regions.

7. Domain Independence: There is need for design of an architecture that is domain independent. It should be easily portable across domains and across different databases within the same domain.
8. Performance evaluation of the model

The above issues formed the basis for this research. Each of the above tasks was tackled through a specific research procedure as described in the sections that follow. The Performance evaluation methodology is covered in chapter four.

3.2 Research Design Synopsis

The following is a synopsis of the research design applied for each of the tasks listed in section 3.1. A detailed report of each research activity is given in the respective sections that follow.

Multiple cases study research design was adopted for informing the designs on concepts discovery envisaged in item 1 of section 3.1 above. In this research, rules of surface to deep structure transformation in sentences, as provided in Transformational theory by Chomsky (1957), was extended and used to study the structure of natural language queries.

In order to enhance the language understanding capacity of the model as envisaged in item 1 c of section 3.1, an exploratory study was carried out on data collected from the case studies and heuristics designed and tested on a prototype for implicit concepts discovery.

Finally in order to expand the collection of what constitutes discoverable ‘concepts’ from an input, findings from Krishnamurthy & Mitchell (2011), Sewangi (2001) and Ohly (1982) were experimentally tested on a prototype and performance measurements taken. The task generally involved building functions of regular expressions of terms, collocations and other linguistic patterns as described in these sources and extending the current concept identification algorithms through these functions. This was an experimental design involving test and comparison experiments.

Design of algorithms for decoding of ontology data and semantic tagging envisaged in item 2 of section 3.1 was informed by analysis of results obtained from field survey and other published

sources. This was an exploratory research involving field data collection and survey and was complemented by an experimental study involving prototype measurements on performance. Design of the two schemas, mapping algorithm and query generation function (items 3, 4 and 5 in section 3.1) was guided by literature survey and analysis coupled with experimental studies while the study on designing a mechanism for handling cross-lingual issues (item 6) was done through experimental design involving prototype performance measurements.

Evaluations were conducted through development of a prototype and taking performance measurements. This is however detailed in chapter 4 of this thesis.

3.3 Research Design for Concepts Discovery Tasks

As pointed out in section 3.2, this research aspect was broken down into three components. These included i) investigations into the development of procedures for concepts discovery through query reduction techniques, ii) expansion of the landscape of what constitutes discoverable 'concepts' from an input and iii) enhancing language understanding capacity through heuristics design for implicit concepts discovery. The studies were carried out through case studies design aimed at theory testing and an exploratory study on collected data for heuristics design for the first two components respectively, and prototype-based experimental design.

3.3.1 Case Studies Design

In order to establish rigor, credibility, transferability, dependability and confirmability, the five point case study research design strategy (Yin, 1994) was adopted. Accordingly the case study design had the following five components,

1. Research question(s),
2. Propositions based on some criteria,
3. Unit(s) of analysis that must provide rigor,
4. Determination of how the data is linked to the propositions
5. Criteria to interpret the findings.

Further to this, a protocol is required to carry out the tasks. A protocol serves as a framework of operation and includes all the necessary elements in the proper conduct of research. This also

enhances clarity and repeatability. In this work the protocol published by the University of Massachusetts at Amherst (Zucker, 2009) was adopted. The protocol is summarized in figure 3.3.

- Purpose and rationale for case study
  - Significance of the phenomena of interest
  - Research questions
- Describe the full case
- Design based on the unit of analysis and research purpose
- Data collection and management techniques
  - Field methods
  - Transcribed notes and interviews
  - Mapping of major concepts
  - Building typologies
  - Member checking
- Focus the analysis built on themes linked to purpose and unit of analysis
- Analyze findings based on the purpose, rationale, and research questions
  - Case perspective
  - Disciplinary perspective
  - Cross-case comparison
  - Write up the case from an emic perspective
  - Biography, autobiography, narratives
- Establishing rigor
  - Credibility
  - Transferability
  - Dependability
  - Confirmability

Fig. 3.3 Protocol Adopted for Carrying out Case Studies

Section 3.3.2 provides an account of the actual case studies carried out following the adopted protocol and results obtained.

3.3.2 Purpose and Rationale for Case Studies (Characterizing Linguistic Features of user Inputs)

Concepts of Transformational theory by Chomsky (1957) providing rules for transforming surface structure forms (SSF) to deep structure forms (DSF) in sentences was used to study the structure of

natural language queries. The theory was tested in the context of 'query-reduction' where it was studied whether if a query is represented using a SSF, it can be mapped onto its equivalent DSF while retaining the same interrogative (information elicitation) properties. If this was found to be the case, the transformation process of a query would be significantly simplified because any query would require only to be transformed to its minimal form (DSF) before being converted into SPaRQL. The relationship between DSF and SPaRQL was also studied. Data collected from the field and other published sources was used in the case studies.

3.3.3 Research Questions for the Linguistic-based Case Studies

The study was guided by the following three questions,

- *Can deep structure forms (DSF) of a query be used in deducing the interrogative properties of a NL query?*
- *What type of relationship exists between DSF and SPaRQL queries and is it language and domain independent?*
- *Are the processes for conversion of SSF to DSF in NL queries language and domain independent?*

3.3.4 Description of the Cases

The research involved investigations from five different case scenarios. Languages selected for this case study in this research were English and Kiswahili because these are the official languages in Kenya and most prevalent in the East African region. The case scenarios are briefly described.

3.3.4.1 Case 1: Kiswahili Queries

Documented research on Kiswahili questions posed to databases is not publicly available. A field survey was therefore necessary to obtain objective Kiswahili queries. In order to get a representative sample of Kiswahili queries a farmers' group, which is a potential user of an NLQ database access system, was selected. Questionnaires were used to solicit potential queries from farmers that would be seeking information from a simulated database containing professional solutions. The set of obtained queries was compared against the theoretically expected query formats (KU, 2011), (Kamusi Project, 2013) with a purpose of establishing how representative the obtained queries were to general Kiswahili queries. The query samples were observed to be representative of general

Kiswahili queries with most of the expected question types being present. These included 'who', 'where', 'what', 'when', 'how', 'which', 'enumeratives', 'superlative-based', 'comparative-based', 'yes/no' disjunctive (choice) and 'list/show/give/find/describe' types of questions.

Sampling Method and Sample Size for Poultry Case

Chain referral method as described in Mugenda, A. and Mugenda, O. (2003) was used in selecting the sample frame. Chain referral is a non-probabilistic sampling technique suitable in this case because respondents must have certain characteristics such as all being farmers of a particular product, and must have a common need for specialized solutions in that particular domain. The solutions could be provided by a human specialist or a specialized database containing solutions. Poultry farmers are likely beneficiaries of products modeled on findings of this research and were therefore selected.

The sample selected was an active poultry farmers' project in Makongeni estate, a sub-urban area of Thika town in Thika district. The participants rear chicken for commercial purposes and therefore were likely to pose queries related to chicken farming to experts. The area was selected because it met the prerequisite conditions such as all participants were regular users of Kiswahili and so were likely to query the database in Kiswahili and the participants use specialist knowledge such as veterinary services at a commercial scale and therefore have a regular need for interacting. These conditions were established via a pre-study survey.

In chain referral, a type of purposive sampling method, sample sizes are determined on the basis of 'theoretical saturation' - that is the point in data collection when new data no longer bring additional insights to the research questions. Purposive sampling is therefore most successful when data review and analysis are done in conjunction with data collection (FHI, 2012). In this research, analysis was performed after every ten questionnaires. However, the saturation point was difficult to determine empirically and was rather subjective, therefore the questionnaires were limited by practical reasons to 50.

Data Collection

A survey questionnaire was the preferred tool because typical text inputs were required for study before the prototype was actually designed and developed. The questionnaire was designed so that

farmers were required to write short questions. Each questionnaire had twenty five information request areas which required the respondent to pose questions to a hypothetical system acting as a veterinary doctor. Six hundred and twenty five questions were collected. The following examples show some of the typical queries that were collected,

1. *Ametoka nchi ipi?* (What is its country of origin?)
2. *Utapata wapi soko ya kuku za nyama?* (Where will you get a market for broilers?)
3. *Nafaa kuwapa kuku maji kiasi kipi?* (How much water should I give the chicken?)
4. *Kuku anayepigwa na wengine anafaa kutengwa?* (Should we separate a chicken that is beaten by others?)
5. *[Nipe orodha ya] wanunuzi bora*. ([Give me a list of] best buyers). The part enclosed in square brackets was not explicitly stated but is necessary to complete the query.
6. *Nitabebewa vifaranga na nani?* (who will transport the chicken?)
7. *Nitapata vifaranga lini?* (When will I get the chicks?)
8. *Vyumba vyafaa kujengwa vipi?* (How should houses be built?)
9. *Vifaa vipi vya kutumiwa kupima?* (Which instruments are used to measure temperature?)
10. *Kuku ipi hutaga mayai mengi kuliko ya kienyeji?* (Which chicken lays more eggs than local ones?)
11. *Jogoo yupi ana uzito kuliko wengine wote?* (Which is the heaviest broiler)

These eleven questions represent the following types of queries namely ‘what’, ‘where’, ‘enumerative’, ‘yes/no’, ‘list/show/give/find/describe’, ‘who’, ‘when’, ‘how’, ‘which’, ‘comparative’ and ‘superlative’ respectively. More typical questions given by the farmers are found in Appendix 1 of this report. The responses to the questionnaire reflected a full spectrum of typically expected query types and were therefore taken to reflect an exhaustive range of text inputs to a poultry farmers system.

3.3.4.2 Case 2: UoN MSc Coordinator

The School of Computing and Informatics of the University of Nairobi runs four MSc degree programs which are coordinated from a central office of the MSc coordinator. The coordinator is responsible for handling students’ queries. The queries were contained in various e-mails and information gathered via a web interface that is maintained by the coordinator. The data collected

was from the domain of students management, and therefore provided a domain variation with case 1. The sample set of respondents was made diverse by increasing the number of students whose queries were picked and therefore information collected from these two sources was random. All questions collected were in English and this also provided a language contrast to case 1 which was in Kiswahili.

Data collected over a period of five years was available. Three hundred and ten questions were extracted and used for analysis. Some typical queries for 'what', 'when', 'enumerative', ''list/show/give/find/describe', 'where', 'yes/no' and 'give' types of queries respectively are given below,

1. Please share with me email address of the lecturer in charge of ICS 645 Natural language interface this semester.
2. When is this month's MSc proposal presentation scheduled?
3. How many students can access the mailing address mscis_07@students.uonbi.ac.ke?
4. I would like to pursue a master's degree in CS, please let me know the prerequisites for course.
5. Where is the venue for MSc proposal presentation scheduled for 14th August
6. Is the deadline for MSc marked scripts still 28th may 2007.
7. Kindly assist me with a tentative program for this year

More samples of the queries are available in appendix 1.

3.3.4.3 Case 3: Queries to Microsoft's NorthwindDB used to Test ELF

Originally created by Bootra (2004) to evaluate ELF natural language query interface to query Microsoft northwind database at Virginia Commonwealth University, the questions were collected electronically from random sources. This data is publicly available at Bootra (2004) or at ELF software home page. All questions collected were in English and this provides good comparative data with case 2.

3.3.4.4 Case 4 and 5: Questions to Microsoft's NorthwindDB used to Test ELF

Cases 4 and 5 are queries for computer jobs and restaurant searching that were originally collected electronically by Tang Lappoon in his PhD work at Texas State University under the supervision of

Raymond Mooney (Tang & Mooney, 2001) and have widely been quoted in literature as benchmark questions for studying natural language interface models. These were selected because they provide a good comparative basis with many other published works. They contain over 500 and 250 questions respectively.

A total of 1805 NL queries were available for analysis. Table 3.1 summarizes these query sets.

Table 3.1: Query sets Used

	Name of Query-set	**No of Questions**	**Description**	**Original Source**
1	Farmers Queries	625	Poultry farmers queries	Muchemi, (2008)
2	UoN MSc Coordinator	310	Questions by UoN MSc students to coordinator	Coordinator e-mails
3	ELF Queries to MS NorthwindDB	120	Originally created by Bootra to evaluated ELF on Microsoft northwind-db (at Virginia Commonwealth University	(Bootra, 2004)
4	Computer Jobs	500	Database and queries for computer jobs used originally by Tang under Ray Mooney for PhD work at Texas State University	Recreated from Tang & Mooney, 2001
5	Restaurant	250	Same as above but for restaurant selection	Tang & Mooney, 2001
	Total	1805		

These questions were used to characterize linguistic features for NL inputs. They were also used in performance evaluation experiments described in chapter four.

3.3.5 Analysis Overview

The underlying theory that was used in linguistic analysis of queries is transformational-generative grammar, brought forth by Noam Chomsky (Chomsky, 1957). Transformational theory was selected in this study because it has previously been used for the query formulation process in MULDER (Kwok, Etzioni, & Weld, 2001), a NL question-answering system. The three components of the original transformational-generative grammar included phrase structure rules, transformational rules and morphophonemic rules (Zellig, 1951). Phrase structure trees help analyze phrase structure rules. Sentence diagrams help analyze S-V-O-modifiers arrangement in a sentence or a set of similar sentences with different S-V-O-modifiers arrangement. A sentence 'has a DSF which is transformed through transformation rules into several surface forms. Morphophonemic rules help model transformations in spoken language. Massamba, Kihore, & Hokororo (1999) have studied the Kiswahili transformations and documented them in their book *Sarufi Miundo ya Kiswahili Sanifu*.

In this study, the theory was tested in the context of 'query-reduction' where it was studied whether if a query is represented using a surface form, it can be mapped onto its equivalent DSF while retaining the same interrogative (information elicitation) properties. If this was found to be the case, the transformation process of a query would be significantly simplified because any query would only require to be transformed to its minimal form (DSF) before being converted into SPaRQL. The relationship between DSF and SPaRQL was also studied. Data collected from field and other published sources was used in the case studies. The five sets of queries as described in table 3.1 were used for studies under the following themes,

- Transfer of semantics (interrogative properties) from NL to DSF in kernelization process,
- Type of relationship existing between DSF of queries and SPaRQL queries,
- Dependence of kernelization process and DSF-SPaRQL conversion on Natural language.
- Prevalence of various transformation rules on collected data,
- Word count of concepts and implications on selection of optimal phrases' length

3.3.6 Kernelization Procedure

A kernel statement is a statement expressed in the simple, active, affirmative and declarative form and is produced directly by phrase structure rules. Other statements are produced from the kernel statements through transformation or combination with other kernel statements.

Sentence diagramming is a basic linguistic analysis tool introduced in the 19th century by Kellog and Reed in their book *Higher Lessons in English* (Kellog & Reed, 1877). They are visualization aids of how different parts of a sentence fit together, with special emphasis to S-V-O and their modifiers. The subject goes to one slot, the verb the other and the object the last slot. Modifiers emanate from these slots and are drawn according to the type of modifier. Sentence-diagrams have therefore been used to study transformational rules in this work.

The first step in the kernelization process involves identifying the main actor or agent. This is the subject of the statement. In the example illustrated in figure 3.4 (b) and (c) (*Ametoka nchi ipi?* What is its country of origin) the subject is the pronoun *'A-'* ('It'), representing the chicken. The second step is to identify what is being affected by the subject or will receive the action that is the object, in this case (*nchi)* 'country'.

The third step is to identify the verb (predicate). A verb identifies the action or required relationship between the subject and the object. In the example shown in figure 3.4, the predicate is *toka* (origin) because it shows the relationship between chicken and country. The fourth step is to identify the modifiers of the subject, object and verb and these are usually indicated by adjectives, prepositional phrases (relationship between two objects e.g. cup *is under* the table), adverbial phrases (e.g. *early in* the morning), verb phrases such as infinitives (*to sleep*), while modifiers to the verbs are usually indicated by adverbs (e.g. walked *slowly*). Modifiers carry great semantic value because they affect the type of answer expected by the interrogator.

At times queries do possess more than one object. In this case the objects are classified as direct or indirect objects. During the kernelization process this aspect is taken care of by allowing multiple objects which relate directly or indirectly with the subject via a predicate. Consider the following sentence observed from the sample query set,

'Ni-ta-i-patia maji kiasi kipi?' How much water will I give it?

The kernelization process produces the subject *ni* (I) and two objects *maji* (water) and *i* (it) where the object '*i*' is a direct object and the object 'it' (referring to the chicken) is an indirect object. The modifier *kiasi kip*i (how much) indicates that a quantity is required (an enumerative type of a query).

The Sentence diagramming technique was used to decompose sentences to minimal form. These minimal forms are equivalent to DSF envisaged in Chomsky, (1957). Sentence diagramming involves the following seven steps,

I. Identify the subject by answering the question, "Who? or What?" is the actor,
II. Identify the object (optional) through answering the question, "Whom? or What?" will receive or be affected by the agent,
III. Identify the predicates (verbs) usually answered by the question "What action is taking place, or what happened in the query?" It is a verb or state of being (for example am, is, are, was, were),
IV. Identify articles (a/an/the) or possessives (my, your, his, hers, its, their, Kamau's etc.)
V. Identify adjectives (words that describe or limit a noun or pronoun) by answering questions such as, "Which one? How many? What kind? What size? What color? etc.
VI. Identify adverbs (words that modify verbs, adjectives or other adverbs) by answering the questions, "How? When? Where? How much? Why?" etc.
VII. Identify prepositional phrases. These are groups of words that begin with a preposition and end with a noun or pronoun which is the object of the preposition e.g. between them.

The identified words are then inserted in a sentence diagram where the subject, object and verb are drawn above the base line and the articles, adjectives, adverbs and prepositional phrases are drawn below the base line as illustrated in Fig 3.4 (a). Fig 3.4 (b) and (c) show the sentence diagram for the sentence '*Ametoka nchi ipi?* (What is its country of origin?)'. The square brackets indicate that the object is implied by the interrogator.

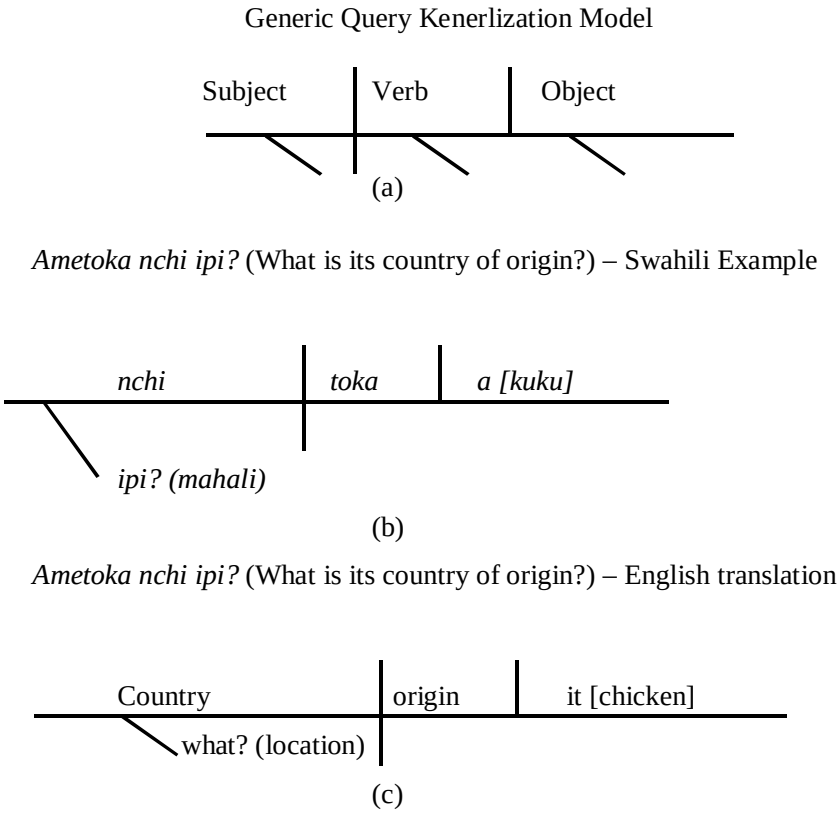


Fig. 3.4 Sentence Diagramming Technique

More illustrative examples are found in appendix 2

3.3.7 Sampling Technique

The query sets were large and required sampling. A stratified random sampling approach was used to select queries from each query set that were subjected to kernelization. The query sets were separated into two population groups, English and Kiswahili. English population group comprised of four query sets namely UoN MSc coordinator, ELF queries to Northwind, computer job searching and restaurants, with a total population of 1180 queries while Kiswahili had a population of 625 queries. Each population in a given group was divided into twelve strata (divisions) each strata containing a unique query type, for example 'when' type. These query types were the most prevalent in the collected query sets and also the most frequently used interrogatives (KU, 2011) and hence their selection. The twelve strata identified were 'what', 'where', 'enumerative', 'yes/no',

'list/show/give/find/describe', 'who', 'when', 'how', 'which', 'comparative', 'superlative' and disjunctive (choice) types. The number of occurrences for each query type (stratum) was recorded for the English and Kiswahili population respectively.

A random sample from each population was taken in a number proportional to the stratum's size of that given population. A total of 50 queries were picked from each population and the diagramming technique applied. Samples of queries that were diagrammed are found in appendix 2. The following formula was used to obtain the number of questions selected for kernelization analysis.

# of Queries Diagrammed = (# of Occurrences of stratum members/Total population) x Desired size

Where the desired size was 50

In order to select the number of queries that were analyzed for rules of transformation, a stratified random sampling approach similar to the one used in the kernelization study described above was used. Eight strata were used and these included 'Imperative', 'Agent deletion', 'Passive', 'Deletion of excessive elements', 'Coordination', 'Addition of elements', 'Negation', and 'Question' transformations.

3.3.8 Results and Analysis of Cases Study Findings

This section describes the results obtained from the five query sets. The sampled queries were subjected to kernelization through the diagramming technique as explained in section 3.3.6.

3.3.8.1 Types of Transformations Noted

Linguistic theory on transformational-generative grammar advanced by Zellig (1951) describes transformational-generative grammar as a rule system formalized with mathematical precision that generates the grammatical sentences of the language that it characterizes, and assigns to each sentence a structural description (Zellig, 1951). Transformational-Generative grammar achieves these transformations without the need of any further information that is not represented explicitly in the sentence. According to the modified transformational theory by Chomsky (1970), transformational-generative grammar (or simply transformational grammar) transforms a "deep structure" (or kernel structure) into a "surface structure" and shows the relationship of such sentences (Chomsky, 1970).

The following transformations were noted in all the query sets. Rules 1 to 5 refer to common transformation rules while rules 6 to 8 refer to generative rules. Transformational rules allow sentences to be changed into identical ones through grammar movement while generative rules generate or create sentences.

1. Imperative transformation (IT) e.g. 'All?' instead of 'List all available jobs' or '*Wanunuzi bora'* (better buyers) instead of *'Nipe orodha ya wanunuzi bora'* (Give me a list of better buyers). The transformation was noted in both English and Kiswahili query sets.
2. Agent deletion transformation (DAT). This is manifested by the deletion of the doer of the action. For example *'(Kuku) Inataga kwa mda gani?* (The chicken) It lays after what duration'.
3. Passive transformation (PT). Transformation from active to passive tense changes the sentence from DSF to SSF. The active form represents the deep structure form e.g. *'kuku ilikunywa dawa'*(The chicken took medicine) while the passive form represents the surface form e.g. *'dawa ilinywewa na kuku'*(The medicine was taken by the chicken).
4. Deletion of excessive elements transformation (DET). This eliminates excessive words and avoids repetition. For example '*Jimbi anakula chakula na vifaranga vinakula chakula?'* becomes '*Jimbi na vifaranga wanakula chakula?* The cock is eating food and the chicks are eating food' becomes 'The cock and chicks are eating food.
5. Coordination transformation (CT). In the DSF two sentences are combined into one surface form sentence e.g. *'kuku zinakunywa dawa; kuku haziponi'* (The chicken have taken medicine; the chicken are sick) becomes *'kuku zina kunywa dawa lakini haziponi'* (the chicken have taken medicine but they are still sick).
6. Addition of elements (AET). This adds information such as adjectives and adverbs. It usually changes the answer provided. For example *'kuku wanakula chakula ya kienyeji?'* (chicken eat local food?) becomes *'kuku wanakula chakula cha kienyeji kingi'* (chicken eat a lot of local food?).
7. Negation transformation (NT). This transformation negates a sentence. For example *'Kuku zinataga mayai'* becomes *'kuku hazitagi mayai' (The chicken are laying eggs* becomes *the chicken are not laying eggs)*.

8. Question transformation (QT). This is distinguished by the tone of the last syllable which is higher, a question mark, inclusion of the term *'je'* in Kiswahili at the beginning of a sentence or by adding a confirmation query at the end of a statement. For example 'The brown cocks are big, aren't they? These transformations were common in English and Kiswahili query sets.

Other transformations such as reflexive transformation and emphatic transformation were not noted in the datasets although they are common transformation rules in generative grammar.

Tables 3.2 and 3.3 show results from some sample queries highlighting transformations and applicable transformation rules. **Bolded words** indicate the primary subject-verb-object data, *italicized word*s show supportive words (in most cases these are adjectives) while the <u>underlined words</u> show non-verb predicates especially state of being.

Table 3.2 Sample Kiswahili Queries Transformations

Studying Transformations in Swahili queries				
No.	**Query (Surface Structure form)**	**Deep Structure (with Concepts Highlighted)**	**Transf. Rule(s)**	**SPARQL-Triples Equivalents**
1	Ametoka nchi ipi?	**Atoka nchi** *ipi*	AET, QT	Kuku/toka/nchi
2	Inataga kwa mda gani?	**I**na **taga mda** *gani*	DAT, QT	Kuku/taga/mda_gani
3	Wakisha komaa nitauzaje?	**Wa***kikomaa* **ni**ta **uza** *aje*	DAT, AET, QT	Kuku/komaa/uza_bei
4	Nitaagiza vifaranga kupitia nani?	**Ni**ta **agiza** ~**faranga** *kupitia nani*	AET, QT	faranga/agiza/pitia_nani
5	Kuku wakigonjeka nitamwona nani?	**Kuku** waki**gonjeka** nitamw*ona nani*	AET, QT	Kuku/gonjeka/ona_nani
6	Nafaa kuwapa kuku maji kiasi kipi?	Nafaa **kuwapa kuku maji** *kiasi* kipi	QT	kuku/pa/maji_kiasi_kipi
7	Vinafaa kujengwa vikielekea jua au la?	**Vi**na **jengwa** *vikielekea* **jua**	DAT, AET, QT	jenga/elekea/jua
8	Ni chombo kipi kinafaa cha kuleta joto inayofaa?	**chombo** kipi kinafaa cha **kuleta joto**	AET, QT	Chombo_kipi/leta/joto_inafaa
9	Wakati gani mtu anafaa kujua joto limezidi?	**Wakati gani mtu anajua joto** *limezidi*	AET, QT	Mtu/jua_wakati/joto_limezidi
10	Ni baridi kiasi gani inatakikana?	**baridi** kiasi *gani ina***takikana**	AET, QT	Baridi/taka/kiasi_gani
11	Chombo kipi kinafaa kutumika?	**Chombo** *kipi* kinafaa ku**tumika**	QT	Chombo_kipi/faa/tumika
12	Nivyombo vipi vinafaa kwa usafi?	Ni**vyombo** *vipi* vinafaa kwa **usafi**	QT	Vyombo_vipi/faa/usafi
13	Kuku wanafaa kuachana katika ukuaji na "gap" gani?	**Kuku** wanafaa **kuachana** *katika ukuaji* na **"gap"** *gani*	AET, QT	Kuku/faa/achana_'gap'_gani
14	Kuku akikomaa anafaa kuwa na uzito kimo gani?	**Kuku** aki**komaa afaa uzito** *gani*	AET, QT	Kuku_komaa/faa/uzito_gani
15	Dawa huharibika kwa mda upi?	**Dawa huharibika** kwa **mda** *upi*	QT	Dawa/haribika/mda_upi
16	Kuku anayepigwa na wengine anafaa kutengwa?	**Kuku** *anayepigwa* na **wengine** *anafaa ku***tengwa**	DET, QT	Kuku/pigwa/kuku_tengwa
17	Ni dalili gani zilizo za kawaida kuku akiugua?	**dalili** *gani* **kuku akiugua**	AET, QT	Kuku/ugua/dalili_gani
18	Nafaa kutumia dawa gani?	*Nafaa* ku**tumia dawa** *gani*	QT	mimi/tumia/dawa_gani
19	Ni njia gani mwafaka ya kuzuia magonjwa?	**Njia** *gani ya* **kuzuia magonjwa**	QT	magonjwa/zuia/njia_gani
20	Unaweza kula kuku mgonjwa?	**U***naweza* **kula kuku** *mgonjwa*	AET, QT	Mtu/Kula/kuku_mgonjwa
21	Unajua aje kuku amefikisha wakati wake wa kuuzwa?	**U***najuaje* **kuku amefik**isha *kuuzwa*	AET, QT	Kuku/umri_kuuzwa/maelezo?
22	Wanunuzi bora?	**Wanunuzi** bora	IT, QT	Mimi/nipe/Majina_wanunuzi_bora
23	Je, ni chakula kipi unaweza patia kuku wa nyama na wa mayai?	**chakula** *kipi* **patia kuku** *wa nyama na wa mayai*	CT, QT, AET	kuku_wa_nyama /patia/ Chakula_kipi kuku_wa_mayai/patia/ Chakula_kipi
24	Kuku wa nyama anastahili kuwa na kilo ngapi kwa siku arobainne?	**Kuku** wa nyama **ana kilo** *ngapi* kwa *siku arobainne*	QT	Kuku_nyama/ uzito_upi siku_arobaine
25	Vyumba vyafaa kujengwa kwa nini?	**Vyumba kujengwa** kwa **nini**	QT, DAT	Vyumba/jengwa/vifaa_gani

Studying English queries transformations was carried out in a similar manner and here are some sample sentences,

Table 3.3 Sample English Queries Transformations

Studying Transformations				
No.	**Query (Surface Structure)**	**Deep structure Equivalent**	**Rule(s)**	**SPARQL Triples**
1	'All of it?'	'*All* **it**?'	IT, DAT	Jobs/type/all
2	'All the jobs please?'	'*All* **jobs**?'	IT, DAT	Jobs/type/all
3	'All?'	'*All*?'	IT, DAT	Jobs/type/all
4	'Any jobs available using database?'	'Any **jobs** available using *database*?'	AET	Jobs/type/any
5	'Type the jobs for a database specialist?'	'**List jobs** for *database specialist*?'	AET	Jobs/type/Db_specialist
6	'Are there Ada jobs outside Austin?	'**Are there** *Ada* **jobs** *outside Austin*?	AET	Jobs/type/ada_outAustin
7	'Are there any Autocad jobs open?'	'**Are there** *Autocad* **jobs** open?'	QT	Jobs/type/autocad
8	'Are there any computer jobs for the playstation?'	'**Are there** *computer* **jobs** for *playstation*?'	AET	Jobs/type/playstation
9	'Are there any jobs in the US with the title verification engineer?'	'**Are there jobs** *in the US* with the **title** *verification engineer?*'	CT	Jobs/type/verification_eng Jobs/located/usa
10	Are there any jobs in 'c++' that the salary is 50000?'	'**Are there jobs** *in 'c++'* that the *salary* is *50000?*'	AET, DET	Jobs/type/c++ Jobs/salary/50000
11	'Are there any jobs requiring a BScs for Boeing in Seattle?'	'**Are there jobs** *requiring a BScs for Boeing in Seattle?*'	AET	Jobs/require/Bsc Jobs/at/Boeing Jobs/in/Seattle
12	'Are there any jobs specializing in AI with JPL?'	'**Are there jobs** *in AI with JPL?*'	AET	Jobs/description/AI Jobs/in/JPL
13	'Are there jobs that do not require a degree in Houston?'	'**Are there jobs** *that do not require a degree in Houston?*'	NT	Jobs/type/No_degree Jobs/in/Houston
14	'Can you show me VB jobs with 50000 salary with databases and excel?'	'**Show me** *VB* **jobs** *with 50000 salary with databases and excel?*'	AET	Jobs/type/vb;db;excel Jobs/salary/50000
15	'Could a senior consulting engineer find work in Boston?'	'*Senior consulting* ***engineer*** **find work** *in Boston*?	AET	Jobs/type/consulting_eng Jobs/in/Boston
16	'Could i have some jobs using SQL with oracle?'	'Could I have **jobs** using *SQL with oracle?*'	AET	Jobs/type/SQL;Oracle
17	'Find all network administration jobs in Austin?'	'**Find** *all network administration* **jobs** *in Austin?*'	IT, AET	Jobs/type/admin Jobs/in/Austin
18	'Give me jobs for a games specialist?'	'**Give me jobs** *for a games specialist?*'	none	Jobs/type/games
19	'I sure do wish there were java assembly jobs out there 'can you help?'	'**Are there** *java assembly* **jobs** out there?'	AET	Jobs/type/java_assembly
20	'What jobs are there for a '3d' graphics specialist?'	'What **jobs are there** for a *'3d' graphics specialist?*'	AET	Jobs/type/3d_graphics
21	'Who might offer me 50000 for web development?'	'**Who** might **offer me *50000*** *for web development?*'	AET	Jobs/type/web_development Jobs/salary/50000
22	'What jobs in Houston are there that require a BSc with 1 year of experience?'	'**Jobs** *in Houston* **require** *a BSc with 1 year of experience?*'	AET	Jobs/require/Bsc;1 year Jobs/in/Houston
23	'Tell me what jobs there are?'	'**Tell me** what **jobs** there are?'	IT	Jobs/type/all
24	'What's available on vax and near Austin?'	'**What** on **vax** and **near Austin** is available**?**'	PT	Jobs/type/vax Jobs/in/near-houston
25	'What oracle jobs are there with compaq in houston using pc?'	'**What** *oracle* **jobs** are there with *compaq in houston using pc?*'	AET	Jobs/type/oracle;pc Jobs/in/Houston

KEY

IT = Imperative transformation (IT) e.g. 'All?' instead of 'List all available jobs';

DAT = Agent deletion transformation (DAT) This is effected by not showing who did the action.;

PT = Passive transformation (PT). Transformation from active to passive tense changes the sentence from SSF to DSF.

DET = Deletion of excessive elements transformation (DET). This eliminates excessive words and avoids repetition;

CT = Coordination transformation (CT). In the DSF two sentences are combined into one surface form sentence;

AET = Addition of elements (AET). This adds information such as adjectives and adverbs.

NT = Negation transformation (NT). This transformation negates a sentence.
QT = Question transformation (QT). This is distinguished by either the tone of the last syllable which is higher or a question mark

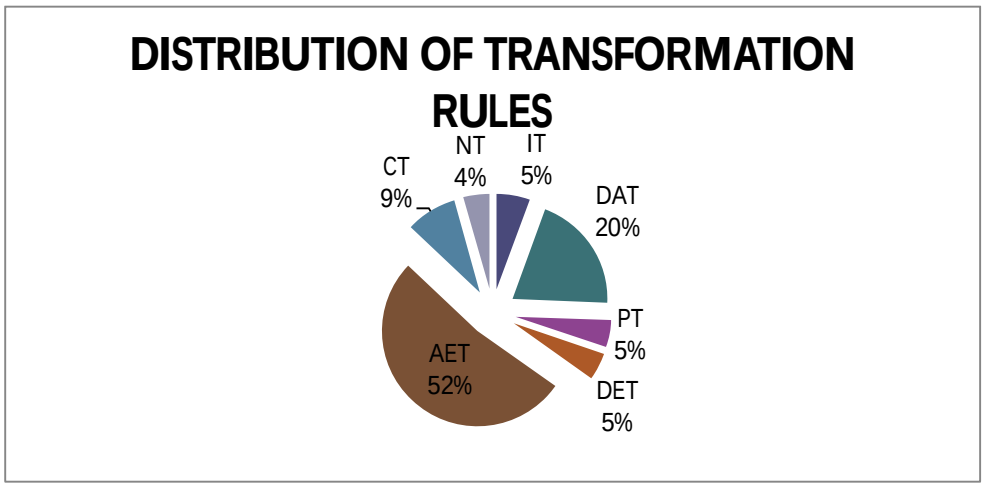
3.3.8.2 Prevalence of Various Transformation Rules

Unrestrained natural language text has very diverse surface representation, and this study cannot therefore claim to be comprehensive enough to create exhaustive rules that govern this phenomenon. However we can study frequency of occurrence of the transformation rules. The rules with a high frequency can be used in optimizing the additional set of rules required in comprehension of natural language queries. For example questions that have the word 'who' will indicate an interrogative state or a missing actor such as a subject or an object. This would then be guided by query transformation (QT) or agent deletion transformation (DAT) rules.

Table 3.4 shows the frequency of occurrence of such rules in randomly selected queries,

Table 3.4 Summary of Prevalence of Transformation Rules

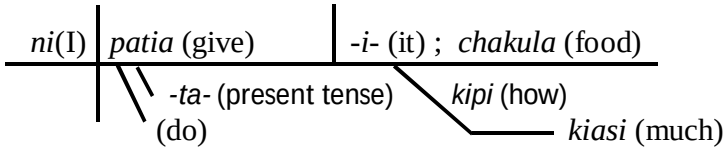
No.	Type of Transformation	Farmers	Computer jobs	ELF Queries to Microsoft Northwind_DB	UoN MSc Coordinator	Restaurant	Average % Prevalence
1	Imperative transformation	8	12	6	2	0	5.6
2	Agent deletion transformation	22	10	15	28	25	20
3	Passive transformation	4	8	7	2	2	4.6
4	Deletion of excessive elements	8	6	2	2	5	4.6
5	Addition of elements	48	56	51	58	48	52.2
6	Coordination transformation	6	5	10	4	18	8.6
7	Negation transformation	4	3	9	4	2	4.4
	TOTAL No. of Randomly Selected Questions	100		100	100	100	100



Note: AET= Addition of Elements Transf.; DAT= Agent Deletion Transf.; PT= Passive Transformation; DET= Deletion of Elements Transf.; CT= Coordination Transf.; NT= Negation Transf.; IT= Imperative Transf.

Fig 3.5 Distribution of Transformation Rules

As observed the transformation-generative rules that the alter semantics of a query (generative) comprised 56.7% of the total queries analyzed with 92.2% of these being addition of elements transformation rules. This means that 92.2% of this category (where semantics are altered) has the primary S-V-O structure modified through additional modifiers such as adjectives and adverbs which changes the type of answer expected. Figure 3.6 illustrates the kernelization of such a sample query *'nitaipatia chakula kiasi kipi?* (How much food shall I give it?).



'*ni-ta-i-patia chakula kiasi kipi?* (How much food do I give it?)

Fig 3.6 Modification of Object by an Interrogative

This query is interpreted as, "ni **(Sub)** *ta(***prefix***)* *–i-***(obj1)***-patia***(verb)** *chakula***(obj2)** *kiasi kipi?***(Obj modifier)** *{How much* **(obj modifier)** food***(Obj1)*** *do (***aux verb***)* I*(***Subj***)* give**(verb)** *it* **(obj2***)?}*

The answer to this would be say, *'nusu kilo'*(half kilo). If the modifiers were stripped the question would be,

Nitaipatia chakula? {Will I give it food?}. This would solicit for a 'yes' or 'no' answer.

3.3.9 Mapping NL Query Semantics to Kernelized Query (DSF) Semantics

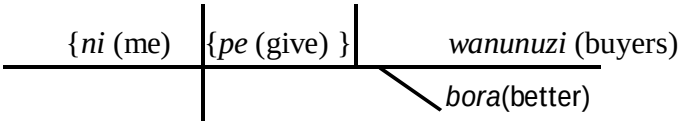
This section discusses the relationship between the semantics of NL queries and the semantics of DSF queries. To understand how the process of transferring meaning in a query occurs, analysis guided by generative-transformation rules was conducted.

A framework that explains this transfer was developed and presented in figure 3.13. The framework was tested for validity by applying queries of different types as detailed in section 3.3.9.3. Eight generative transformation rules described in section 3.3.8 were used in the semantics transfer analysis of the sampled queries. Stratified sampling method as described in 3.3.7 was used in the validation analysis where twelve query types identified in 3.3.7 were used for validity analysis.

3.3.9.1 Analysis of Semantics Transfer through Transformation Rules

RULE 1: IMPERATIVE TRANSFORMATION

An imperative transformation changes a sentence in its kernel form to an imperative surface form. When an imperative transformation is performed on queries, it was observed that the meaning (information solicitation) of the two queries is similar but the verb and subject have to be deduced. To create an imperative, the verb form is changed to its infinitive form but the word 'to' is excluded. For example the SSF query, *'Wanunuzi bora'* (better buyers) is in its imperative form. The associated DSF query would include an agent and a verb *'ni-pe* ('give me') to become *nipe wanunuzi bora*' (give me better buyers). This is illustrated in the kernel form shown in figure 3.7. Other examples from the collected dataset are found in appendix 1.



{Nipe} wanunuzi bora ({Give me} better buyers)

Nb. { } refer to missing elements in the original query

Fig. 3.7 Kernelization of an Imperative Transformation Query

From the analyzed data, this type of transformation affected about 5.6% of queries (see table 3.4). From figure 3.7 it observed that in order to obtain the meaning of this type of a query, the primary meaning bearing components (subject, verb and object or objects) must be identified from the query and any modifiers also taken into account. Implicit components must be deduced from the query. Meaning is transferred through relationships formed between these components. In the above illustrated case, the query is requesting for names (or other identifier attributes) of the 'buyers' who have an associated attribute 'better'. The meaning is therefore formed by the tripartite relation occurring between the object and two of its modifiers (the adjectives). The logical relation therefore is *'buyers-name-betterBuyer'* which is generalized as *"Object-Modifier1?-Modifier2"*. From this query it is observed that reference to the first person (me) does not have any impact on the answer provided because the interrogator is merely a recipient of the names generated. Likewise the verb 'give' or 'list' or other related verbs have no impact on the answer generated because the interrogator's motive is assumed to be known that is, seeking information.

RULE 2: AGENT DELETION TRANSFORMATION

Agent deletion transformation, where the action doer is deleted, was observed in about 20% of the cases. The basic information solicitation properties of the query were observed to remain unchanged. An example from the Kiswahili dataset *'inataga kwa mda gani?'* (in what period does the chicken lay?) shown in figure 3.8 illustrates this concept. In this case the agent *kuku* (chicken) is replaced by its pronoun *i-na* (it).

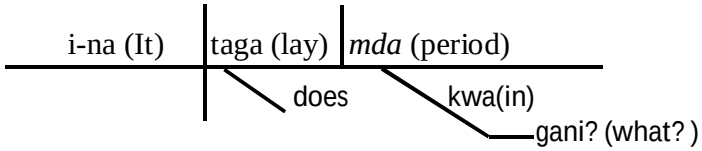
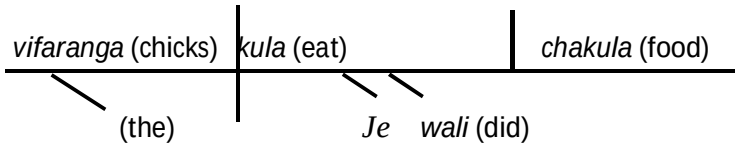


Fig. 3.8 Kernelization of an Agent Deletion Transformation Query

In this type of transformation, the meaning is carried by the primary meaning bearing components (subject, verb and object) and their modifiers, however the actor must be deduced from context. In database access problem the context is usually narrow and the pronoun can be easily deduced from the verb 'lay' which implies a 'subject' that lays and that would be a chicken. The relation for this meaning would therefore be "Chicken-Layegg-WhatPeriod?" which is generalizable to *"Subject-Verb- ObjectModifier"*

RULE 3: PASSIVE TRANSFORMATION RULE

The active form of a sentence represents the deep structure while the passive form represents the surface structure. This type of transformation known as passive transformation was observed in 4.6% of the collected sentences. An example of the SSF query *'Je, chakula kililiwa na vifaranga?'* (Was the food eaten by the chicks?) is transformed to the DSF form *'Je, vifaranga walikula chakula?'* (Did the chicks eat food?) is illustrated in figure 3.9. In the DSF 'the chicks' is the subject, 'eat' is the main verb, 'did' is auxiliary verb and 'food' the direct object.



'*Je, vifaranga walikula chakula?* (Did the chicks eat food)

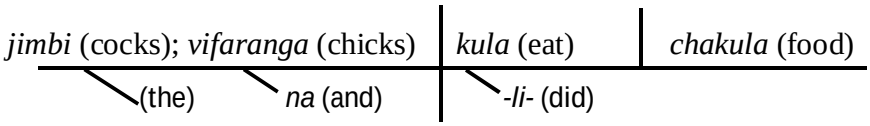
Fig. 3.9 Kernelization of and Passive Transformation Query

Once a query is turned to its DSF, the next step is to obtain the base meaning-bearing components which is done through extraction of phrases at the subject, verb, object levels and their modifiers.

The expected answer is an affirmation or denial of this DSF form which has the pattern 'Chicks-Eat-Food' which is generalizable to *"Subject-Verb-Object"*. The semantics are of the NL is therefore transferred through the primary components.

RULE 4: DELETION OF EXCESSIVE ELEMENTS TRANSFORMATION RULE

Deletion of excessive elements transformation is a type of transformation where excessive words are eliminated to avoid repetition. This was observed in 4.6% of the collected queries. An example from the collected query set is '*Jimbi walikula chakula na vifaranga walikula chakula?'*. This is transformed to '*Jimbi na vifaranga walikula chakula?'*. The transformed query with two components as its subject is illustrated in figure 3.10



'*Jimbi na vifaranga walikula chakula?* (Did the cocks and chicks eat food?)

Fig. 3.10 Kenerlization of Deletion of Excessive Elements Transformation Query

Deletion of excess components leads to a more concise surface structure, however at the DSF level this can be seen as two DSF representations with similar verb and object or objects but with different subjects.

In this type of query meaning is transferred through the base meaning-bearing components (subject, verb and object) of the two DSFs. The expected answer would be an affirmation or denial of the two DSFs. The patterns for this example are *"Subject-Verb-Object"*

RULE 5: COORDINATION TRANSFORMATION RULE

Another type of transformation rule that does not affect the semantics of a query is coordination transformation. This combines two or more sentences in the kernel form to one sentence in the surface form sentence. For example *'Kuku walikula chakula kimeoza kisha wakahara?'* (Did the chicken take rotten food then they diarrhead?). The most common coordination terms observed in the dataset included *'ilhari'* (but) *na* (and), *pia* (also), *wala* (although) and *kisha* (then). These formed 8.6% of the queries. In this type of a query, the statement is broken into two fragments to form two kernel sentences which together retain the original meaning.

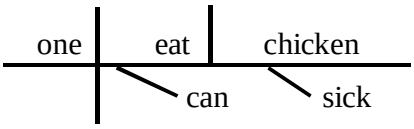
For example in the query, *'Kuku walikula chakula kimeoza kisha wakahara'* (The chicken took rotten food then they diarrhead), the two kernel statements are *'Kuku walikula chakula kimeoza' (*The chicken took rotten food) and *'Kuku walihara' (*The chicken then they diarrhead). These are broken down into their SVO components and their respective modifiers. Just like in the transformation discussed in rule 4 above meaning is transferred through the base meaning-bearing components (subject, verb and object) of the two DSFs. The expected answer is an affirmation or denial of the two DSFs. The patterns for this example are *"Subject-Verb-Object"*

RULE 6: ADDITION OF ELEMENTS TRANSFORMATION RULE

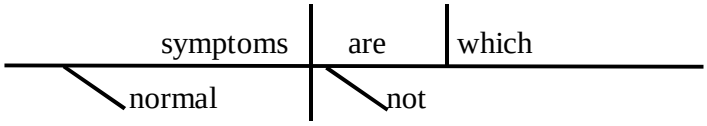
The addition of elements transformation AET transformation in which a DSF is transformed to other SSF through addition of modifiers was the most prevalent within the datasets and formed 52.2% of the analyzed queries. An example of AET is shown in figure 3.12 (a). As highlighted in section 3.3.8.2 the primary S-V-O structure is modified through additional modifiers such as adjectives and adverbs which changes the type of answer expected. In the examples shown in figure 3.6 and 3.12(a) (How much food do I give it? and 'Can one eat sick chicken') the object is modified. In other cases the subject is modified (see example in figure 3.12(b)). The patterns in these examples illustrate diverse patterns such as *"Verb-Object-ObjModifier"* and *"Subj-SubModifier-Object"*. Hence both primary components and their modifiers carry the semantics from NL to DSF.

RULE 7: NEGATION TRANSFORMATION RULE

In negation transformation, the verb is modified through negation. In this case the verb modifier is a critical element in understanding of the query and must therefore be accounted for in any subsequent meaning comprehension exercise. An example is illustrated in figure 3.12 (b).



(a) Example of AET Query: Can one eat sick chicken?



(b) Example of AET with NT Query: Which symptoms are not normal?

Fig. 3.12 Kernelization of Addition of Elements and Negation Transformation Queries

Another illustrative example that demonstrates this phenomenon is '*Je, haitakula chakula*? (Will it not eat food?) The DSF for this query is 'It will *NOT* eat food' where '*NOT*' modifies the verb. This is same for Kiswahili where the negation is actually prefixed to the verb. In another example where the query is supported by 'DO' this phenomenon of negating the verb is maintained. For example *'Mbona hukuongea ukweli?* (Why did you not talk the truth?). The support by the 'do' word does not cause a breakdown of the general rule of negating the verb.

RULE 8: QUERY TRANSFORMATION RULE

This type of query is stated as an ordinary declarative statement, however it is turned into a query by either of the following processes,

- Raised tone at the second last syllable for example '*maji haya ni **SA**fi*?'(This water is c**LEA**n?).
- In English the switching of the auxiliary verb and the subject is a more common usage, for example 'Is this water clean?)

- By adding a confirmation query at the end of a statement. For example *'Jogoo wa rangi ya kahawia ni wakubwa, au siyo?* 'The brown cocks are big, aren't they?
- Inclusion of the term 'je' in Kiswahili at the beginning of a sentence

3.3.9.2 Summarizing Query Semantics Transfer Process

In the above transformation processes, the base meaning-bearing components are the subject, verb and objects. In about 56.7% of the total queries analyzed the semantics were altered through modifying the attributes of the subject and the object. Modification of the verb was observed in negation transformation queries.

From this analysis we can therefore conclude that *for a deep structure query to carry the same interrogative properties as the original surface structure query, it must contain all the S-V-O terms as well as other modifying words and more so adjectives and adverbs.* The significance of this finding is that a surface to deep structure query conversion algorithm should extract both S-V-O and modifiers for it to be complete. Transformational-generative rules defined for many languages should be applied for these types of transformations. While these rules may not cover all the complexities of natural language free text, they are by large a useful modelling aid to obtain the basic meaning bearing components in a query. Further it was deduced that semantics is transferred through various combinations of the base primary meaning-bearing components and their modifiers. The patterns were however observed to be consistently triples.

The process of converting a question's surface structure form to the deep structure form and vice versa is affected by the underlying morphophonological processes. The quality of morphological processing is therefore affected negatively thereby impacting on the efficiency of extraction of roots from a highly agglutinated language such as Kiswahili. Morphophonological rules should therefore be taken into account to enhance the success rate of identifying roots of words. It is known that morphophonological rules mediate between phonological and morphological processes. Morphophological analysis provides rules that predict the regular sound changes occurring in the morphemes of a given a language. Various studies such as Iribe (2008), Port (1982) and Choge (2009) have made a review of Kiswahili morphophonological processes. They convert the underlying representations into the surface structure that is spoken and hence written. An example of such a process that affects Kiswahili is vowel harmonization. For example the interaction of the

vowels *'u'* and *'e'* results in semi vowel *'w'*. For example *'m**(u)-i**mbaji'* becomes *m**w-i**mbaji* (singer). In this case only the prefix is affected. In other cases the root is altered by the morphophonological processes. For example words *'m-ingi', 'ny-ingi','w-ingi'* (meaning 'a lot') has the root 'ingi'. This means that when obtaining the root, the process would ideally involve dropping the prefix before the *'–ingi'*. In plural the morphology is interfered with if the required prefix has an ending '–a' for example *wa-* and instead of saying *'wa-ingi'* we say *'w-**e**ngi.* The –a and –i in the prefix and root respectively interact to become –e-. This means that although the underlying structure would be *wa-ingi*, what appears as surface form is *wengi*. This change in expected morphology is explained through morphophonological processes. Investigation into other morphophonological rules found within the collected query set was beyond the scope of this research and hence standard morphophonological rules such as those in Iribe (2008), (Port, 1982) and (Choge, 2009) were assumed adequate for practical computational purposes. Morphophonological processes affect the quality of morphological processing and hence extraction of roots from a highly agglutinated language.

In summary the overall conversion processes from a linguistic point of view as analyzed above is summarized as shown in figure 3.13.

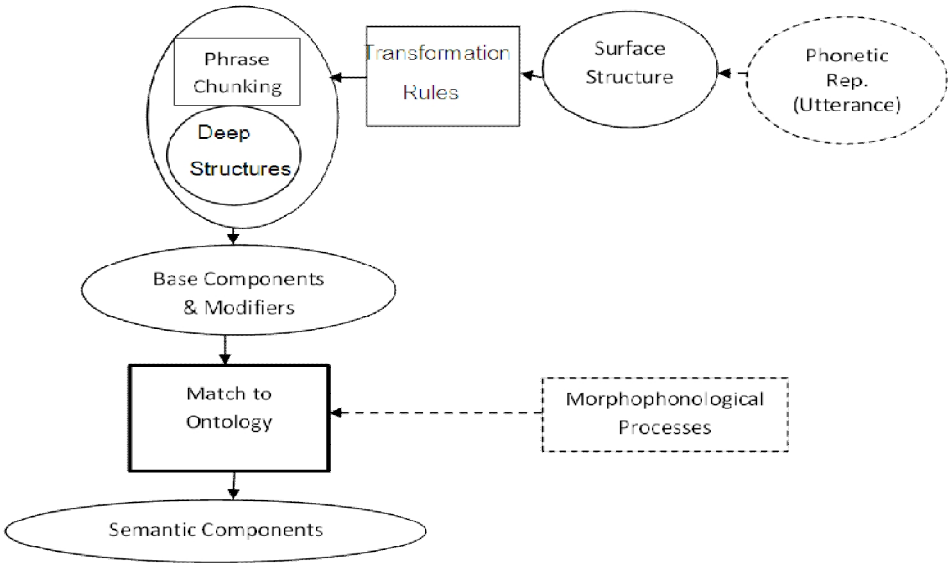


Fig 3.13 Query Semantics Transfer Model (QuSeT Model)

Understanding this process of meaning transfer is important because it informs the formulation of the structured query say in SPaRQL. The following is an example to illustrate the query semantics transfer model

Spoken: *Maji mengi yalinywewa na kuku?* (Was a lot of water taken by the chicken?)

Transformation-generative Rules (English)

The surface structure is an interrogative: 'Was a lot of water taken by the chicken?'

I. Transformed to Passive Declarative form: 'A lot of water was taken by the chicken'
$Aux – NP_2 – en – V – by – NP_1 \rightarrow : NP_2 – Aux. – en – V – by – NP_1$

II. Transformed to active simple declarative form: 'Chicken took a lot of water' which is in DSF'
$NP_2 – Aux. – en – V – by – NP_1 \rightarrow NP_1 – V – NP_2$
Where 'Aux' is the auxiliary and '-en-' indicates that the verb is modified by suffix 'en'.

Transformation Rules (Kiswahili)

The surface structure is an interrogative: *Maji Mengi yalinywewa na kuku? (*'Was a lot of water taken by the chicken?')

I. The interrogative is transformed to simple active declarative form
$NP_2 – V – na – NP_1 \rightarrow NP_1 – V – NP_2$ (*Kuku walikunywa maji mengi?)*
Where 'na' is a conjunction.

The base components (SVO) and their modifiers were found to exist as phrase-chunks within the query, therefore the base components of the DSF and their modifiers were extracted through the regular phrase chunking.

3.3.9.3 Qualitative Validation of the Query Semantics Transfer (QuSeT) Model

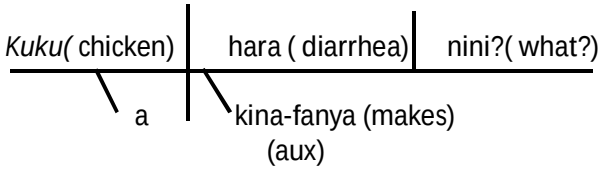
In the QuSeT validation analysis, twelve query types were used. These query types were identified in section 3.3.7 as 'what', 'where', 'enumerative', 'yes/no', 'list/show/give/find/describe', 'who', 'when', 'how', 'which', 'comparative', 'superlative' and disjunction (choice) types of queries. For each query type the following was analyzed,

- How kernelization occurs and if it conforms to the procedure described in the QuSeT,
- Types of elements that comprise meaning bearing components,
- Relationship between meaning bearing components within the DSF structure,
- Whether the semantic transfer process occurs without deviation to the procedure described in the QuSeT model.

1. WHAT Query TYPE

Figure 3.14 shows kernelization for sample query '*Nini kinafanya kuku kuhara?* (What makes a chicken diarrhea?). 'What' is used either as a pronoun for example as in 'what is this?' or as an adverb say as used in 'In what way did she go?' It is also used as an adjective.

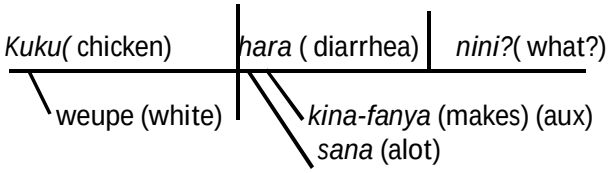
In this type of a query the subject and the verb carry the meaning bearing elements which the object is represented by the interrogative 'what'. If modifiers are added to the subject and verb or to either of them the modifiers must also be taken into account while understanding the semantics of the query as discussed in section 3.3.3.2.



Nini kinafanya kuku kuhara? (What makes a chicken diarrhea?)

Fig 3.14 Kernelization of a 'What-Query' Type

Figure 3.15 shows kernelization for sample query 'what makes white chicken diarrhea a lot'.



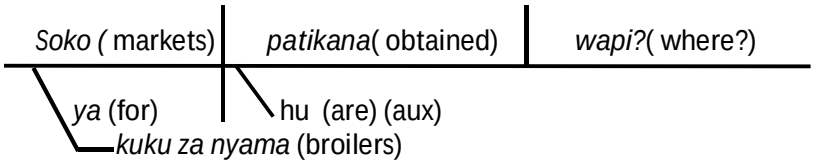
Nini kinafanya kuku weupe wahare sana? (What makes white chicken diarrhea a lot?)

Fig 3.15 Kernelization of a 'What-Query' Type with Modifiers

In this example, the query is converted to its base components and associated modifiers in form of phrases which are to be used for semantics mapping with the ontology mentions. This process of meaning transfer conforms to the framework presented in figure 3.13.

2. WHERE Query TYPE

Figure 3.16 shows kernelization for sample query '*Soko ya kuku za nyama hupatikana wapi?* (Where are markets for broilers obtained?).



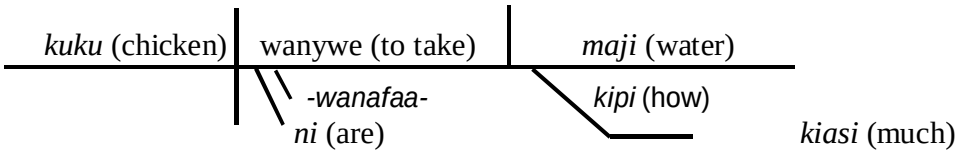
Soko ya kuku za nyama hupatikana wapi? (Where are markets for broilers obtained?)

Fig 3.16 Kernelization of a 'Where-Query' Type

'Where' is used either as a noun (e.g. where is the toilet?), as an adverb (e.g. where does this lead to) and as a conjunction (did we meet where there was an accident?). In general it was observed that the subject, verb, object and their modifiers carry the meaning of the query in whichever way the query is formed.

3. ENUMERATIVE Query TYPE

Figure 3.17 shows kernelization for sample query '*Je ni maji kiasi kipi kuku wanafaa wanywe?* (How much water are chicken to take?)'



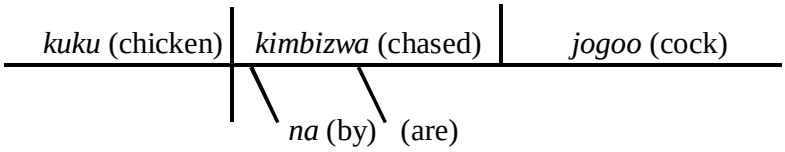
Je ni maji kiasi kipi kuku wanafaa wanywe? (How much water are chicken to take?)

Fig 3.17 Kernelization of an 'Enumerative-Query' Type

From these types of query it was observed that the subject, verb, object and their modifiers carry the meaning. The interrogative is the object modifier and its value signifies answer to the query.

4. YES/NO Query TYPE

Figure 3.18 shows kernelization for sample query '*Je kuku hukimbizwa na jogoo?* (Are chicken chased by cocks?)'. A DSF has no mood and hence is always a positive statement. However the negation modifies the DSF as shown in fig 3.18 below.



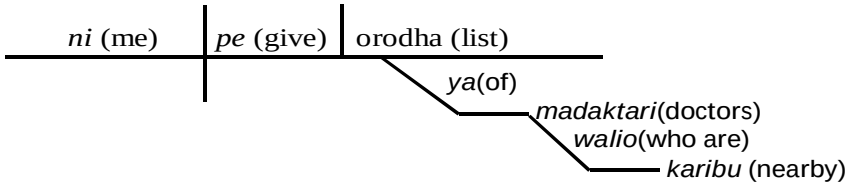
Je kuku hukimbzwa na jogoo? (Are chicken chased by cocks?)

Fig 3.18 Kernelization of a 'Yes/No-Query' Type

From this example it is observed that the subject, object and their modifiers are the main meaning bearing elements. In both Swahili and English the SVO word order is retained, however English 'yes/no' queries take an auxiliary verb at the beginning of the query. Kiswahili queries may contain the emphasis word *'je'* meaning an answer is expected.

5. GIVE/LIST Query TYPE

Figure 3.19 shows kernelization for sample query 'Nipe orodha ya madaktari walio karibu (Give me a list of the doctors who are nearby),



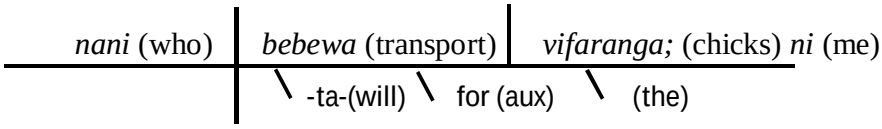
Nipe orodha ya madaktari walio karibu (Give me a list of the doctors who are nearby)

Fig 3.19 Kernelization of a 'Give/List-Query' Type

From this type of query, the meaning bearing terms are found at the object and its modifiers. The verb indicates the need to obtain the information while the object shows who gets the information. This type of a query can be answered by processed the object and its modifiers.

6. WHO Query TYPE

Figure 3.20 shows kernelization for sample query '*Nitabebewa vifaranga na nani?* (Who will transport the chicken for me?),



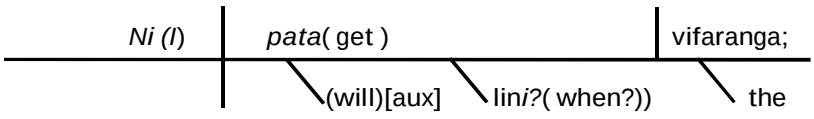
Nitabebewa vifaranga na nani? (who will transport the chicken for me?)

Fig 3.20 Kernelization of a 'Who-Query' Type

In this type of a query the interrogative 'who' represents the agent (doer) hence it is the subject of the query. 'the chicks' is the direct object (because it is taking the action directly) while 'me' is an indirect object because the action does not affect it directly. The meaning of the query is carried by the subject, verb, object and their modifiers where the interrogative is replaced by the actual subject in order for the answer to be obtained. It is important to note that a query must be in its passive form for it to be in DSF.

7. WHEN Query TYPE

Figure 3.21 shows kernelization for sample query '*Nitapata vifaranga lini?* (When will I get the chicks?)',



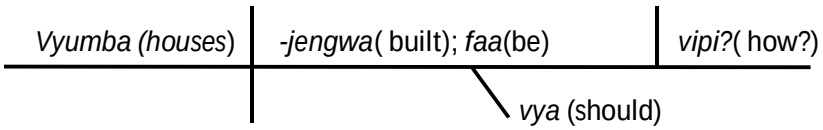
Nitapata vifaranga lini? (When will I get the chicks?)

Fig 3.21 Kernelization of a 'When-Query' Type

In this case, the interrogative 'when' is an adverb and modifies the verb. The answer could be 'when you finish the payment'. In other usage 'when' manifests as a pronoun (you need the report by when?), as a conjunction (Did she sit when she saw him come in?) or as a noun. The usage is so diverse that it is difficult to accurately use computational methods to distinguish them. In that case the phrase containing 'when' is treated as either a subject or object or as a modifier of any of the SVO.

8. HOW Query TYPE

Figure 3.22 shows kernelization for sample query '*Nitapata vifaranga lini?* (When will I get the chicks?)',



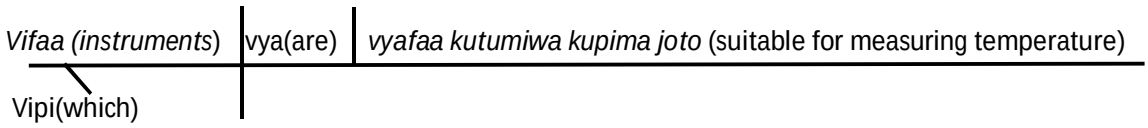
Vyumba vya-faa kujengwa vipi? (How should houses be built?)

Fig 3.22 Kernelization of a 'When-Query' Type

How appears as an adverb (in what manner), a conjunction (e.g. 'did he tell them how he had a situation?) or as a noun (e.g. do you know the how of getting there?). The computational treatment is similar to that of 'when' explained above.

9. WHICH Query TYPE

Figure 3.23 shows kernelization for sample query '*Nitapata vifaranga lini?* (When will I get the chicks?)',



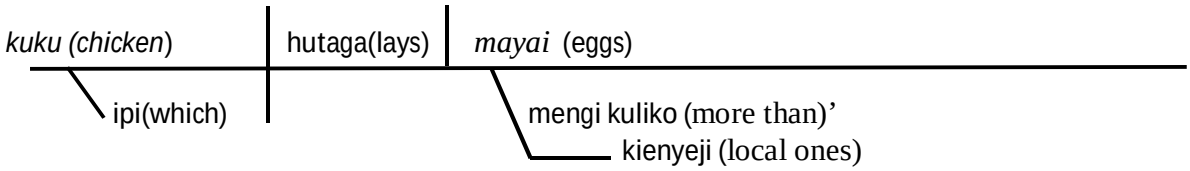
Vifaa vipi vyafaa kutumiwa kupima joto? (Which instruments are suitable for measuring temperature?)

Fig 3.23 Kernelization of a 'Which-Query' Type

'Which' acts like an adjective like in the example above or as a pronoun e.g. 'They wanted husbands which they got easily'. In first case it is treated as modifier while in the second example it replaces the subject.

10. COMPARATIVE Query TYPE

Figure 3.24 shows kernelization for sample query *'Kuku ipi hutaga mayai mengi kuliko ya kienyeji?* (Which chicken lays more eggs than local ones?)',



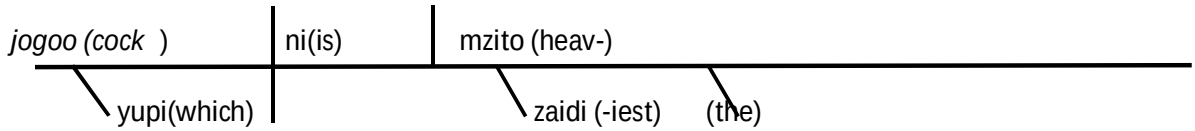
'Kuku ipi hutaga mayai mengi kuliko ya kienyeji? (Which chicken lays more eggs than local ones?)'

Fig 3.24 Kernelization of a 'Comparative' Type

Comparative words such as 'better' may be used in very diverse sense. For example as an adjective (e.g. Is this a better thesis?), as a verb (e.g. Will you better your handwriting?), as an adverb (Did she walk in a better way?) and as a noun (e.g. Is hers a better behavior?). The presence of a comparative in these types of queries prompts a 'yes/no' answer or a noun-phrase in place of the wh-interrogative. In these cases two kernel statements are required. The first DSF contains a subject, a verb (or auxiliary) and a modifier to either the subject or verb or auxiliary (for example 'Other| is| thesis-*good*') while the second contains the second subject, a similar verb or auxiliary to the first DSF, and the comparative modifier (for example 'This| is| thesis-*better*') where this modifier refers to the adjective or adverb contained in the second DSF. The expected answer is realized by testing for the truth of either given the contents of the underlying ontology. However when an interrogative, such as 'wh-' is combined with a comparative as in the example illustrated in figure 3.24, the object is modified by a phrase formed from the comparative. The interrogative modifies or replaces the subject depending on the type.

11. SUPERARATIVE Query TYPE

Figure 3.25 shows kernelization for sample query *'Jogoo yupi ni mzito zaidi?* (Which is the heaviest cock?)



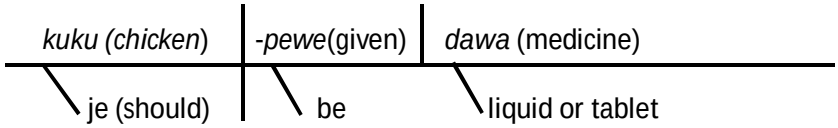
'Jogoo yupi ni mzito zaidi? (Which cock is the heaviest)

Fig 3.25 Kernelization of a 'Superative-Query' Type

Superlatives mainly appear as adjectives in queries (e.g. Will you wear your best cloth?). The DSF is written using the base components and their modifiers. In the example illustrated in figure 3.25 the object is modified with the superlative *'heaviest'*.

12. DISJUNCTIVE (CHOICE) Query TYPE

A study of these types of questions revealed that they deliver the meaning in a similar manner to disjunctive queries (or choice questions) e.g. *'Tuwape kuku dawa ya tembe au ya maji?'* (Do we give the chicken tablets or liquid medicine?). In this latter example a conversion to passive form is required for the query to be in DSF. The query then becomes, *'Kuku apewe dawa ya tembe au ya maji?'* (Should the chicken be given tablets or liquid medicine?). The kernelization then becomes,



'je kuku apewe dawa ya tembe au ya maji?' (Should the chicken be given tablets or liquid medicine?).

Fig 3.26 Kernelization of a 'Disjunctive-Query' Type

The disjunctive query behaves in a similar manner as the conjunctive query where the disjunction or conjunction separates objects, subjects or their modifiers. For the query to be answered the transformation lists two parallel kernel forms and tests for the truth of either using the contents of the underlying ontology

3.3.9.4 Quantitative Validation of the QuSeT Model

The QuSeT model was empirically tested for validity by applying two question sets from English- and Kiswahili-based query sets. Each set had a total of 25 questions drawn from the farming (Kiswahili) and UoN MSc Coordinator's (English) question sets respectively. A stratified sampling method similar to the one described in section 3.3.7 was used in building the two test sets. Twelve query types identified in 3.3.9.3 were used as the strata. These queries included 'what', 'where', 'enumerative', 'yes/no', 'list/show/give/find/describe', 'who', 'when', 'how', 'which', 'comparative', 'superlative' and disjunction (choice) types of queries.

The QuSeT model was built as a python module (see appendix 9). The questions from the test-sets were passed to the module and observations as to whether the module correctly identified the base words (or groups of words) and their modifiers noted. For the farmer's question set, 23 out of 25 questions were analyzed correctly, meaning all the base words and the modifiers were identified in at least 23 of the questions. This represented 92% accuracy. In the English query set, 24 questions were accurately analyzed, meaning that the accuracy was 96%. The mean accuracy of the QuSeT model was therefore determined as 94%. These accuracies are only but indicators of the efficiency of the QuSeT model.

3.3.9.5 Conclusions from Query Semantics Transfer Analysis

From analysis of the twelve query types it was concluded that,

- There exists a regular process in which the general semantics of a query is transferred from the surface structure to the base meaning-bearing components. For all query types analyzed there was conformity to the general transfer framework illustrated in figure 3.13,
- References to the first and second persons in a query do not bear direct reference to interrogative elements of the query. They represent the interrogator and the computer

respectively and in such a case the meaning should be deduced from the verb or object and their respective modifiers only. The subject is not considered as a meaning bearing phrase. In other cases, the verb is used to direct the action of the computer such as listing and in such a case only the objects or the subjects and their modifiers transfer the essence of the query.

- The interrogative (wh- word) substitutes either the subject or the object (as in 'who' and 'where' queries respectively) or modifies the subject, verb or object (as in 'which', 'when' and 'enumerative' queries respectively). The expected answer from the query is realized through substitution of the interrogative with a suitable meaning-bearing component from the ontology being queried. Other types of queries such as disjunctions and comparatives realize the expected answer by listing two parallel kernel forms and testing for the truth of either given the contents of the underlying ontology.
- Meaning-bearing components have a tri-partite relation which may be formed between,
    - the three primary components (subject, verb and object) or,
    - any two of these components and an interrogative or a modifier of either or
    - any of the primary components and its modifiers which may appear as a phrase such as a preposition

3.3.10 Relationship Between Meaning-bearing Elements of DSF SPaRQL and the Ontology

SPaRQL is a structured query language and data access protocol for the Semantic Web. SPaRQL is based on the Resource Description Framework (RDF) data model and therefore works for any data source that can be mapped onto RDF. SPaRQL, just like the RDF structure illustrated in figure 2.8, is built on triples, where a triple is a set consisting of three-elements:

```
?element1 ?element2  ?element3
```

The first element represents the database name, while the second represents the field name. The third element represents the row value, meaning the attribute's instance. An example of a SPaRQL query is provided in figure 3.27. The triple is observed in the 'WHERE' clause.

```
PREFIX north: <http://www.owl-ontologies.com/NewNorthwind#>
SELECT  ?SupplierID ?Name ?Region
WHERE { ?suppliers db:SupplierID ?SupplierID.
        ?suppliers db:CompanyName ?CompanyName.
        ?suppliers db:Region ?Region
FILTER(?Region = "central")}
```

Derived from the query: "Give me the names and identification of supplier from central region"

Fig. 3.27 An Example of a SPaRQL Query

The general structure of the SPaRQL is shown in figure 3.28

```
PREFIX alias_name: <http://www.URL#>
SELECT  ?Attribute1 ?Attribute2……?AttributeN
WHERE{ ?Element1 db: Element2 ? Element3.
     ? Element1 db: Element2 ? Element3.
     ?.............................
      ? Element1 db: Element2 ? Element3.
FILTER(?Attribute = Element3)}
```

Fig. 3.28 General Structure of the SPaRQL Query

In the general SPaRQL structure shown in figure 3.28, element 1 corresponds to the table name, element 2 the column name and element 3 to the row value if this is matched to relational database elements via an ontology. The row value, except the one in the filter line is a variable that is filled when the answer is provided by the system.

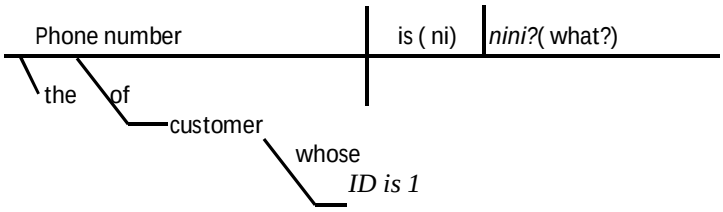
The sections that follow analyze the relationship between the various elements of a SPaRQL triple and the underlying ontology on the one hand, and DSF's base-components and modifiers on the other hand.

Consider a query where the interrogator is requesting for information about the telephone contact of a customer whose identification number is 1. This query may be stated as a 'WHAT-type or as a 'GIVE/LIST-Type' explained in section 3.3.9.3. The queries then are either,

Query One: 'What is the phone number of the customer whose ID is 1'

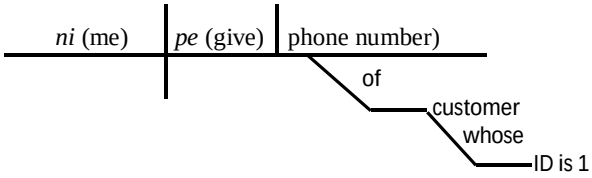
Query Two: 'Give me the phone number of the customer whose ID is 1'.

The kernelization of these statements yields the structures given in figure 3.29 and 3.30. The ontology from which the interrogator is requesting for information is shown in figure 3.31.



'What is the phone number of the customer whose ID is 1' or restated as ('The phone number of *the customer whose ID is 1* is WHAT')

Fig. 3.29 Kernelization of Illustrative Query One



Give me the phone number of the customer whose ID is 1'

Fig. 3.30 Kernelization of Illustrative Query Two

The base components and their modifiers are obtained from both the transformation and phrase structure formation rules as highlighted in the query semantics transfer framework illustrated in figure 3.13. The base components and the modifiers are the main semantic bearing elements. In the examples shown in figures 3.29 and 3.30 the meaning is carried by the two phrases contained within the query *'the phone number of the customer'* and *'customer whose id number is 1'*. As stated in section 3.3.9.4 references to first and second pronouns as well as the verb *'give'* do not have a direct effect on the answer being sought. This means that even if these elements are identified in the kernelization process as components of the DSF they are not considered as base-elements. They are thus dropped in the SPaRQL formation process.

Since there are two meaning bearing phrases, two triples are formed each having the following format: `?element1 ?element2  ?element3,`

These are,

```
?customer ?phone_number  ?Variable1
?customer ?id_number  ? Variable2
```

```
#DATATYPE PROPERTIES DEFINITION(DATABASE COLUMN NAMES)#
<owl:DatatypeProperty rdf:about="&db;customers.Phone">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <rdfs:domain rdf:resource="&db;customers"/>
        <db:hasOrigColumnName
rdf:datatype="&xsd;string">Phone</db:hasOrigColumnName>
        <rdfs:range rdf:resource="&xsd;string"/>
    </owl:DatatypeProperty>

<owl:DatatypeProperty rdf:about="&db;customers.CustomerID">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <rdfs:domain rdf:resource="&db;customers"/>
        <db:hasOrigColumnName
rdf:datatype="&xsd;string">CustomerID</db:hasOrigColumnName>
        <rdfs:range rdf:resource="&xsd;int"/>
    </owl:DatatypeProperty>
```

```
#INSTANCES DECLARATION (ROW-VALUES FOR EACH RECORD)#

<db:customers rdf:about="&db;customers_Instance_1">
        <db:customers.Address rdf:datatype="&xsd;string">Obere Str.
57</db:customers.Address>
        <db:customers.City
rdf:datatype="&xsd;string">Berlin</db:customers.City>
        <db:customers.CompanyName rdf:datatype="&xsd;string"
            >Alfreds Futterkiste</db:customers.CompanyName>
        <db:customers.ContactName
rdf:datatype="&xsd;string">MariaAnders</db:customers.ContactName>
        <db:customers.ContactTitle rdf:datatype="&xsd;string"
            >Sales Representative</db:customers.ContactTitle>
        <db:customers.Country
rdf:datatype="&xsd;string">germany</db:customers.Country>
        <db:customers.CustomerID
rdf:datatype="&xsd;int">1</db:customers.CustomerID>
        <db:customers.Fax rdf:datatype="&xsd;string">030-
0070000</db:customers.Fax>
        <db:customers.Phone rdf:datatype="&xsd;string">030-
0074321</db:customers.Phone>
        <db:customers.PostalCode
rdf:datatype="&xsd;string">12209</db:customers.PostalCode>
        <db:customers.Region
rdf:datatype="&xsd;string">stutgart</db:customers.Region>
    </db:customers>
```

Fig. 3.31 Segment of OWL-based RDF Ontology from Northwind Database

There is a mention of a specific row value (instance) and hence an addition `FILTER` clause is required. It was observed that when specific row values are mentioned, the user's intention is to constrain the number of records (rows) returned. The general syntax for this filter is as follows,

```
FILTER (?Variable = "value")
```

For the queries in figures 3.29 and 3.30 the filter is as follows,

```
FILTER (? id_number = "1")
```

In conclusion there is a guided mapping between the deep structure form of queries and SPaRQL triples and filters. Some noted conditions include dropping of first and second person pronouns, dropping 'give' or 'list' verbs and making deductions to attributes and referents that are not directly mentioned before SPaRQL processing commences. This conclusion is important because it guides the SPaRQL query formation algorithm.

3.3.11 Average Word Count of Concepts in Kiswahili Queries

The average word count analysis is important because it indicates the lower and possibly the upper bound number of words that typically express a concept. More importantly it indicates the optimal number of words typically expressing a concept and therefore guides any rule-based concept discovery process.

The observations shown in table 3.5 were made by manually isolating the concepts from selected farmers' queries and the number of words per concept counted. A total of fifty randomly selected questions were analyzed from the Kiswahili query set for farmers with a total of one hundred eighty four concepts observed.

For example in a sentence such as '*vifaranga ni bei gani*?', the underlined words represent concepts that are one word concepts while in '*kuku za nyama huuzwa wapi?*' the underlined words represent a three word concept. In rare occasions, sentences with four, five and six-word concepts such as the sentence *'masaa ishirini na nne'*, '*kifaranga wa kuku wa nyama anapatikana na pesa ngapi*?', 'vifaranga wa *kuku wa nyama wanaokuwa na kukosa kutembea …*' respectively were discovered.

Table 3.5 shows the average word count in concepts for Kiswahili queries.

Table 3.5 Average Word Count of Concepts in Kiswahili Queries

Number of words in the Concept	Number of Concepts Counted	Average % Prevalence
1	118	64.1
2	12	6.5
3	47	25.5
4	2	1.1
5	4	2.1
6	1	0.5
7	0	0
TOTAL No. of Concepts	**184**	**100**

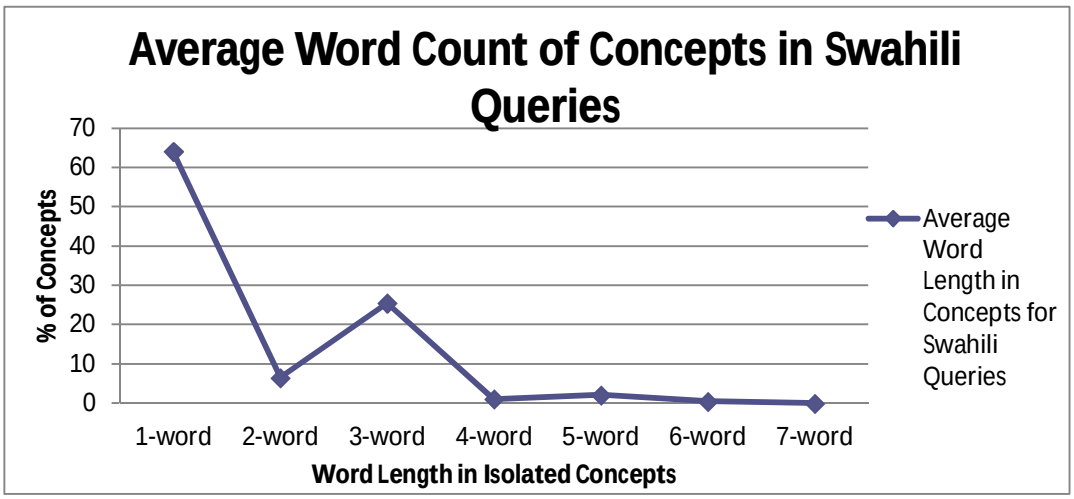


Fig 3.32 Average Word Count of Concepts in Kiswahili Queries

From figure 3.32, it can be observed that the most frequent concept lengths are one, two and three words. This means that patterns with at most three words are the most prevalent and a length of three words is thus optimal for a rule-based concept discovery process.

3.3.12 Independence of Kernelization and Triple Formation on Natural Language

The data applied in this analysis was from two languages and was independently collected from five case study sets. There were no significant differences in the results for Kiswahili or English query analysis as reported in sections 3.3.6 through 3.3.10. Section 3.3.6 highlighted the kernelization procedure which as described is language independent, while 3.3.8 gave an analysis of the prevalence of generative-transformation rules where again similar trends were noted for English and Kiswahili query sets. Section 3.3.9 outlined a framework of query semantics transfer which was a generic framework that fits into any language. The processes within the framework included generative-transformation, phrases formation and base-components and modifiers identification. In order to migrate from one language of querying to another, only the set of pre-defined language specific rules such as phrase structure rules and transformation rules need to be changed to match the querying language. Changing the set of rules is automatically achieved through a language recognition pre-process which causes the appropriate set of rules to be loaded. Further the SPaRQL triples formation process described in section 3.3.10 is totally language independent.

The use of *phrase structure rules, transformational rules and morphophonemic rules* is a practice that linguists have embraced since the days of Chomsky(1957) and Zelig (1951). Linguists have studied these rules over the years. These methods are context free and language independent. In other words they may be applied to any context and any language.

Transformational rules which are the backbone of this work are context free and have been developed for different languages. For example Kiswahili was studied and published by TUKI (Massamba, Kihore, & Hokororo, 1999). The process that is not practically universal (ie not language independent) is the way the rules are extracted. *This thesis does NOT claim this type of language independence but rather once the rules are manually extracted by linguists of a particular language, the rules may be computationally deployed to synthesize sentences in a language independent manner*. Since the rule sets extracted from the 5 case studies (for Swa and Eng) were

not necessarily exhaustive, the results do not seek to prove a theory but rather indicate the direction towards a language independent computational solution.

In general the transformational process flow from NLQ to SPaRQL is therefore language independent to the extent that rules have been pre-extracted.

3.4 Survey on Database Schema Authorship

Relational databases have no controlled vocabulary for naming tables and columns and therefore the resulting ontologies do not have a controlled lexicon. Subsequently a challenge is encountered in decoding database schema information, specifically names of tables and fields. An equivalent study for ontology elements in the field of ontology engineering revealed a common practice nomenclature for classes and properties. (Damljanovic, Tablan, & Bontcheva, 2008). The common practice involves the use of a dash or an underscore to separate names or abbreviations of names or the use of the 'camelCase' style for concatenating or separating names and abbreviations. Observations from the reviewed literature on ontologies derived from non-relational database sources, indicate that various ontology parsing algorithms such as reported in Tablan, Damljanovic, & Bontchev (2008) and Damljanovic, Agatonovic, & Cunningham (2010), use this common practice nomenclature while creating gazetteers. A gazetteer is a data-holding structure that contains concepts extracted from ontologies and can be viewed as an entity dictionary. When creating gazetteers for ontologies derived from relational databases these nomenclature assumptions may not hold because databases may have different nomenclature practices.

A survey was thus carried out to identify the common practice nomenclature for databases', tables' and column names. The findings from this study guided the creation of a general algorithm that handles many ontologies created from commonly available or legacy relational databases.

3.4.1 Study of Common-Practice Nomenclature of DB-Schema Objects

3.4.1.1 Purpose and Rationale for Common-Practice Nomenclature Study

The purpose of this segment of research was to establish if a common naming practice for relational databases exists and if so, then the answers to the following questions would be established based on collected data,

- *Is there a finite set of patterns that database schema authors' use in representing database schema object names?*
- *'How can we decipher the meaning of an 'intended concept' from the schema name?*
- *How can a general 'Concepts Re-construction Algorithm' be built from an ontology created from a relational database source?*

This study therefore sought to create a concepts reconstruction algorithm that would lead to the automation of gazetteer construction.

3.4.1.2 DB-Schema Objects Nomenclature Methodology Overview

The research methodology selected for this investigation was an exploratory study where data was collected through field surveys and a qualitative analysis technique applied.

The data that was specifically collected included,

- names of databases, tables and columns as authored by respondents,
- Existence of formal policies on naming procedures by different organizations,
- Existence of historical, common company-wide naming practices though not described as a formal policy
- Personal preferences for naming styles

The survey involved data collection from twelve training institutions and sixteen software development firms. Further, an internet based study of 320 randomly identified database schema object names was carried out to identify other nomenclatures used. Nine database management systems were also studied and profiled with respect to permitted rules for authoring objects and attributes names.

3.4.2 Sampling Method

Information that was to be obtained from various sources was considered confidential to the organizations providing the information and therefore an approach that guarantees confidentiality and confidence was preferred. The snowballing sampling method also known as chain referral sampling is suitable for hard-to-reach or hidden populations. It was selected because the chain referral aspect leads to building confidence in the interviewees. Questionnaire and interviews were

the preferred tools because of their simplicity and effectiveness. A total of eight questions were crafted into a questionnaire and database/application designers, developers and administrators were required to provide short answers. The questionnaire is found in Appendix 3 of this report.

3.4.3 Sample Frame and Size

In creating the sample frame two groups dealing with back-end services namely database-applications' developers and database development trainers were targeted. The two starting points of the chain referral sample frames were an application development firm and a university (database development) lecturer. Each grouping had an initial starting point which snowballed into other referral persons. A total of 28 units were interrogated. This being a purposive sampling method the size was determined on the basis of theoretical saturation, that is the point in data collection when new data no longer brings additional insights to the research questions. Since each questionnaire had a large number of schema objects to be analyzed (up to 6 database names, 12 table names and 16 column names), analysis was performed after every three questionnaires collected and the saturation point approximated. The internet-based survey involved a review of three hundred and twenty randomly selected database schema objects, a number which was limited by practical reasons.

3.4.4 Analysis Overview

In the first research question namely '*Is there a finite set of patterns that database schema authors use in representing database schema object names?'* analysis was done by way of discovering patterns used by various database schema authors. For each database schema object analyzed, a pattern was determined on how the author represented object names. Of interest were database, table and column names.

In the second research question namely *'How can we decipher the meaning of 'intended concepts' from the schema names?* analysis was done by way of recreating words from ontology representations and mapping them to lexical definitions. The process of recreating the words was recorded and later analyzed for presence of general patterns. For example the ontology representation *'dateOfBirth'* is reconstructed to *'date of birth'* and the process to do this requires the insertion of a space between lower case and upper case letters.

3.4.5 Results from Database Schema Authorship Studies

From the nine database management systems studied, it was observed that all allowed the use of ten commonly used nomenclature styles. Table 3.6 shows this summary. Table 3.7 shows the extent of usage for application development firms and DB development trainers.

3.4.5.1 Permitted Styles by Various Database Management Systems

The purpose of this profiling was to establish whether there are some string combinations that are not allowed by some database management systems. Nine commonly used database management systems were selected. These included the following,

1. MySQL
2. Microsoft SQL Server
3. Oracle
4. MS Access
5. SQLite
6. OpenOffice.org Base
7. IBM DB2 (Viper, Cobra and pureScale versions)
8. PostgreSQL
9. SmallSQL

Published literature for the respective software was studied and the results are shown in table 3.6.

Table 3.6 Permitted Objects Naming Styles (DB Servers)

	Pattern	Permission for usage by various DBMS (Tick= permit)									Comments
		1	2	3	4	5	6	7	8	9	
1	Under_score	✓	✓	✓	✓	✓	✓	✓	✓	✓	Allowed by all
2	camelCase	✓	✓	✓	✓	✓	✓	✓	✓	✓	Allowed by all
3	Da-sh	✓	✓	✓	✓	✓	✓	✓	✓	✓	Allowed by all
4	Abbreviations emp for employe	✓	✓	✓	✓	✓	✓	✓	✓	✓	Allowed by all
5	Pascal Casing	✓	✓	✓	✓	✓	✓	✓	✓	✓	Allowed by all
6	Finger_Breaking_Underscore	✓	✓	✓	✓	✓	✓	✓	✓	✓	Allowed by all
7	SCREEMING_UNDERSCORE	✓	✓	✓	✓	✓	✓	✓	✓	✓	Allowed by all
8	Acronyms eg ID, UI, IO	✓	✓	✓	✓	✓	✓	✓	✓	✓	Allowed by all
9	Dot eg hr.hire_date	✓	✓	✓	✓	✓	✓	✓	✓	✓	Allowed by all
10	“string like this”	✓	✓	✓	✓	✓	✓	✓	✓	✓	Allowed by all

The naming styles shown in table 3.6 are explained here below,

- Underscore concatenates two or more words or abbreviations using the underscore character. An example is sender_name,
- Abbreviation uses short form of names usually the consonants. An example is tbl as in tbl_name,
- Pascal style is one where every main word in a concatenation of words starts with upper case letter including the first word. For example CustomerAddress.
- Acronym style employs commonly or easily recognizable short forms for example ID in empID, RegNum).
- Dot style is where a period is inserted between two parts of a compound name. For example Tbl.location.
- Finger breaking style involves the combination of underscore and capitalization of the first letter of all main words as in the compound word Last_Name.
- Dash naming style involves the use of minus symbol between key words of a concatenated compound label. Example is chassis-num.
- String style uses the inverted commas usually to represent a string which has a blank between two words of a compound label.as in the example ‘first name’.
- Camel case is a naming style where main words in a compound label start with a capital letter except the first one. For example logbook.
- Scream nomenclature style involves the use of upper case letters only as in the example POSTCODE.

3.4.5.2 Results from Training and Development Firms

Thirty universities and training institutions carrying out database training and twenty software development companies were targeted. Information from twelve training institutions and sixteen software development companies was obtained and analyzed (see appendix 7 for names of these firms). The results are tabulated in Tables 3.7 and 3.8 respectively.

Table 3.7 Schema Objects Naming Techniques (Training Firms)

	Pattern	Extent of Usage by Training Firms (scale of 1-5; 5=most used, 1= least used; 0= never used)																
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	
1	Under_score	4	4	5	5	4	5	5	5	4	4	5	3	4	5	5	4	4.44
2	camelCase	3	3	2	2	3	3	3	3	3	2	2	2	3	2	2	2	2.5
3	Da-sh	0	0	1	0	1	0	1	0	1	0	0	0	0	1	0	0	0.31
4	Abbreviations	4	4	3	2	3	3	3	2	4	4	3	4	4	3	3	4	3.31
5	PascalCasing	5	4	5	4	5	4	4	5	4	5	4	5	5	4	4	5	4.5
6	Finger_Breaking_Underscore	3	1	2	2	2	3	2	2	2	1	2	3	3	2	3	2	2.19
7	SCREEMING_UNDERSCORE	0	0	1	0	1	0	0	0	0	0	0	1	0	1	1	1	0.38
8	Acronyms eg ID, UI, IO	4	4	5	4	5	4	4	5	4	4	4	4	3	4	3	3	4
9	Dot eg hr.hire_date	1	1	3	2	2	3	1	1	1	2	1	1	1	1	1	1	1.44
10	“string like this”	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0.06

The analyzed results which are presented in figure 3.33, revealed that underscore, abbreviations, Pascal and acronyms are the most frequently used styles while dot, finger and camel styles are moderately used. On the other hand dash, string and scream styles are seldom used by training institutions.

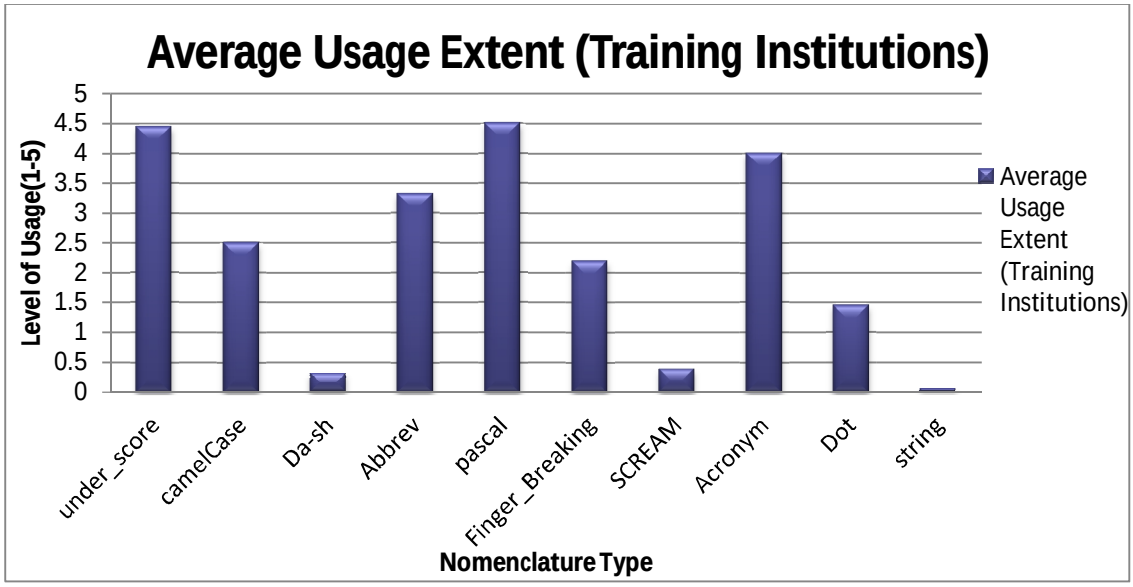


Fig 3.33 Average Usage of Nomenclature Type (Training Institutions)

Table 3.8 Schema Objects Naming Techniques (Software development Firms)

	Pattern	Extent of Usage by Development Firms (scale of 1-5; 5=most used, 1= least used; 0= never used)												
		1	2	3	4	5	6	7	8	9	10	11	12	Average
1	Under_score	5	5	5	5	5	3	5	5	4	5	5	5	4.8
2	camelCase	0	0	0	0	3	2	0	1	1	0	1	1	0.8
3	Da-sh	0	0	0	0	1	0	0	1	3	0	0	1	0.5
4	Abbreviations	1	2	0	4	5	4	5	1	1	5	3	3	2.8
5	PascalCasing	5	4	4	4	5	4	4	4	4	4	5	5	4.33
6	Finger_Breaking_Underscore	0	4	3	4	3	4	2	0	1	2	3	3	2.42
7	SCREEMING_UNDERSCORE	0	0	3	2	4	5	0	0	2	0	2	2	1.7
8	Acronyms eg ID, UI, IO	1	3	5	4	4	5	0	5	5	5	4	4	3.8
9	Dot eg hr.hire_date	1	3	3	4	1	0	0	0	3	3	2	1	1.8
10	“string like this”	1	0	0	0	0	1	0	0	2	0	1	0	0.4

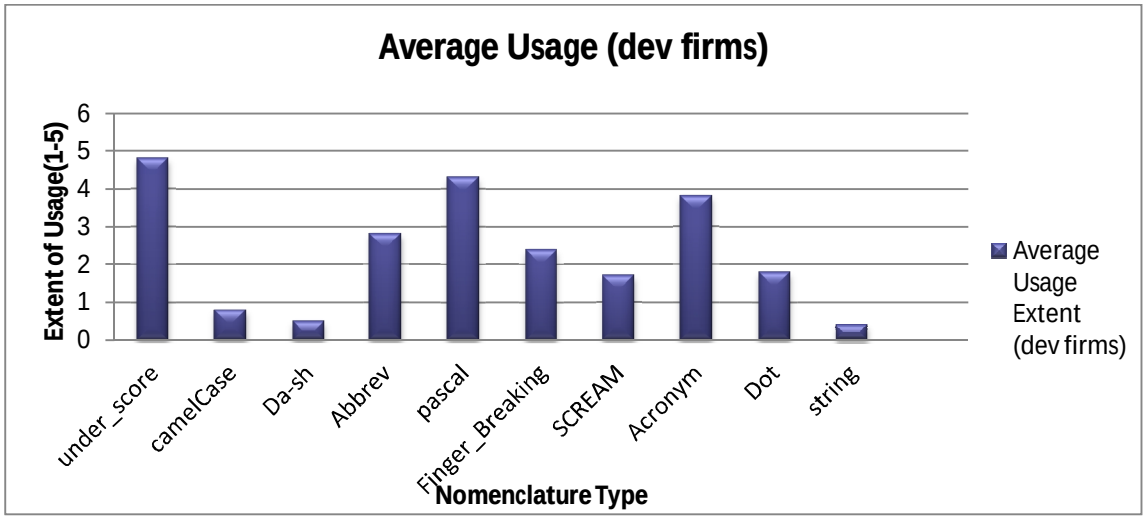


Fig 3.34 Average Usage of Nomenclature Type (Development Firms)

The results from the analysis presented in figure 3.34 showed that underscore, abbreviations, Pascal and acronyms are the most frequently used styles while dot, finger and scream styles are moderately used. On the other hand dash, string and camel styles are seldom used by development firms.

3.4.5.3 Results from Internet-based Survey

The internet based study involved analysis of 320 randomly collected database schema object names. The nomenclature styles were observed and the results tabulated in tabulated in table 3.9.

The data was analyzed and is presented in figure 3.35. The frequency of occurrence of various patterns from the internet based survey revealed that the underscore, abbreviation, dot and acronyms naming styles are the most frequent while dash and string are least used. Pascal, finger, camel and scream are moderately used.

Table 3.9 Results from Internet-based Survey

	Type Observed	Number Observed
1	Under_score	55
2	camelCase	22
3	Da-sh	5
4	Abbreviations	56
5	PascalCasing	34
6	Finger_Breaking_Underscore	24
7	SCREEMING_UNDERSCORE	28
8	Acronyms eg ID, UI, IO	47
9	Dot eg hr.hire_date	40
10	“string like this”	5
11	others	4
		320

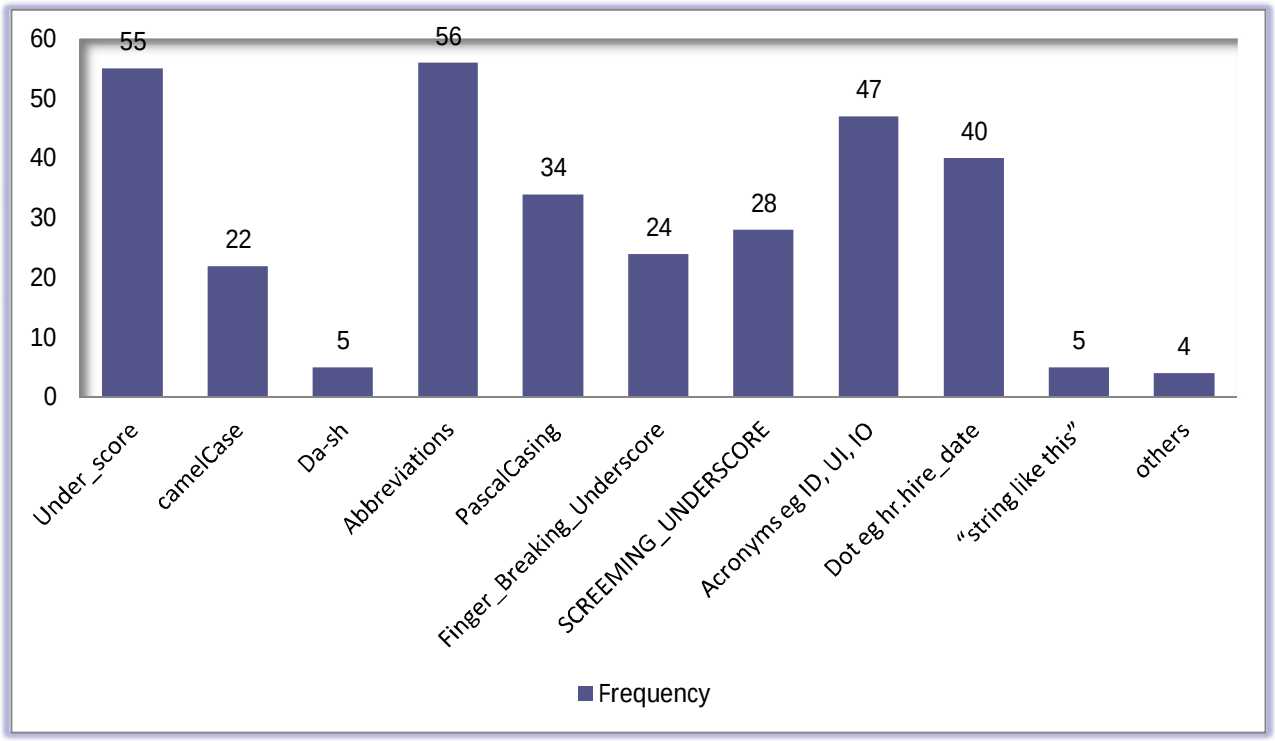


Fig 3.35 Frequency of Occurrence of Various Patterns in Internet based Survey

3.4.6 Analysis of Data from Database Authorship Surveys

3.4.6.1 Classification of Naming Styles

The analysis was done by formation of clusters. Three clusters emerged and these were; 'highly preferred', 'averagely preferred' and 'least preferred'. These are summarized in the table 3.10,

The 'highly preferred' cluster included underscore, abbreviation, Pascal and acronyms. The 'averagely preferred' cluster contained dot and finger breaking naming styles while the 'least preferred' cluster contained dash and string naming styles. Training institutions ranked camel case style as average while development companies seldom prefer this style. On the other hand development firms rated the scream nomenclature as average while training institutions avoided teaching this type. Further a study of 320 randomly identified database schema object names was reviewed to identify other styles used. The preference of naming style was found to closely resemble that of training and software development firms surveyed. An algorithm that handles decoding all the ten styles would be preferred but preference would be in the styles in the first cluster.

Table 3.10 Clustering of Preference Levels

	Category		
Cluster Type	**Training Institutions**	**Development Firms**	**Internet-based Firms**
Cluster 1: High	Underscore/Abbrev/Pascal/ Acronyms	Underscore/Abbrev/Pascal/ Acronyms	Underscore/Abbrev/Dot/ Acronyms
Cluster 2: Average	Dot/ Finger/Camel	Dot/Finger/ Scream	Pascal/Finger/Camel/Scream
Cluster 3: Least	Dash/String/Scream	Dash/String/Camel	Dash/String

3.4.6.2 Words Recreation Task

Analysis was also done to assess the relationship between the 'written form' and the 'meaning-bearing phrases' or recreated word form. This is analogous to answering the question,

'How can we decipher the meaning of 'intended concepts' from the corresponding schema names?

Column, table and database names were extracted from the questionnaires and several databases sourced from the internet. As explained in section 3.4 words were recreated from ontology representations. The lexicon obtained for each representation was then recreated into probable phrase chunks. The size of the chunks was dependent on lexicalizations obtained from the ontology representations. Majority of the chunks were found to be formed by one, two or three words, which is consistent with the findings of section 3.3.11 that found the most probable number of words typically expressing a concept to be one, two or three.

An example is given next for illustrating how word recreation was carried out. Consider the ontology representation *'titleOfCourtesy'*. This is reconstructed to *'title of courtesy'* and the process followed is that of inserting a space between lower case and upper case letters. The representation *'stud_ID'* is formed by two concatenated words *'student and Identification'*. The decoding process involves separation of the two probable lexeme abbreviations 'stud' and 'ID' and using a simple lexicon look-up mapping method to assign the meaning of the abbreviations. In this work a lexicon look-up approach was used because of the small size of the ontologies involved in the experiments. This task of recreating words and assigning them to probable phrase chunks was viewed as a semantic assignment task of the ontology data. The process of recreating the words was recorded and later analyzed for presence of general patterns which were coded into an algorithm discussed in

section 3.4.7. Words recreation process was found to be the reverse of the database objects naming methods identified in 3.4.5. For example *'stud_ID'* is recreated to the full form, *'student Identification'* whereas a table with a column intended for holding *'students' identities'* would be named in an abbreviated or shortened form say *'stud_ID'*.

3.4.6.3 General Observations from Questionnaires

Analysis was done on data collected for policy and personal preferences regarding the naming method of database objects. Several significant observations were made and are summarized as:

1. Each database developer has some form of policy on naming procedure. These policies are not necessarily formalized and are usually not published but are evident from the names observed.
2. Database developers rarely give names that do not have meaning. These meanings highly correlate with the intended concept.
3. In a few cases, an abbreviation related to the object type is included. For example 'tbl' in 'tbl-*tablename*' reflects that this is a table. This has an impact on deciphering the 'intended' concept.
4. Some authors over-abbreviated or used un-recognizable abbreviations.
5. Some acronyms required human intervention to decode.

3.4.7 Ontology Words Reconstruction Algorithm (OWoRA)

3.4.7.1 Description of OWoRA Algorithm

The purpose of this algorithm is to extract concepts from an ontology and provide a list of concepts in the form of phrase chunks. Figure 3.36 illustrates the algorithm.

```
1 function Words-Reconstruct():
2    function Elements-Scrapper(OWL ontology)→ List;
3       L := List;
4       S := Words in L;
5      if under-score in S → load function under();// Takes care of
         underscore & Finger_Breaking
6       if da-sh in S → load function dash();
7       if dot in S → load function dot();
8       function under(S): //similar for function dash()& function dot()except
          line 10 that is altered accordingly
9           for t in S:
10              if t is upper; //dash or dot
11              Index(t); //Find position of t
12              Split S at t → S1, S2, … Sn;
13          If si > 1;
14              stem S1, S2, … Sn → S1s, S2s, … Sns;
15          Call Semantic_assigner(Sis);// lexicon lookup
16              S1s, S2s, … Sns  → W1, W2, … Wn;
17          P → W1+W2+Wn // Phrase chunk
18          Return P;
19       if string in S → load function string();
20       function string(S):
21           for ' " ' in S;
22              convert string → list of S1, S2, … Sn;
23              stem S1, S2, … Sn → S1s, S2s, … Sns;
24              Call Semantic_assigner(Sis);// lexicon lookup
25              S1s, S2s, … Sns  → W1, W2, … Wn;
26              P → W1+W2+Wn // Phrase chunk
27              Return P;
28      Else
29         t:= Characters in S;
30              If  ti & ti+1 is upper S → load function acronym();
31                Call Semantic_assigner(Sis);// lexicon lookup
32                Return P;
33              Else → load function abbrev()
34                Call Semantic_assigner(Sis);// lexicon lookup
35                Return P;
36  end Words-Reconstruction.
```

Function retrieves schema elements from ontology and forms a List

Different functions handle various patterns found in the strings

Split compound Strings & do Stem

Identify lexicon & synonyms & form Associated concepts (Phrase chunks + other categories)

Return phrases

Fig 3.36 Ontology Words Reconstruction Algorithm (OWoRA)

This algorithm takes in the ontology derived from a relational database as its input and gives out a list of concepts derived from the ontology. The ontology elements which contain the abbreviations or acronyms of concepts are decoded and the underlying semantics discovered through the processes described in 3.4.6. Once the concepts are extracted they are assembled into a gazetteer whose construction details appear in section 3.5.2

3.4.7.2 Evaluation of the OWoRA Algorithm

The aim of this evaluation process was to experimentally determine the efficacy of the words reconstruction algorithm described in 3.4.7.1. Schemas from the Microsoft's Northwind_db as described in Table 4.2, and five other randomly selected databases published by Oracle (Oracle, 2008) and Vertica Systems (Vertica Systems, 2011), namely human_resource_management_db, order_entry_db, retail_management_db, phone_db and stock_exchange_db described in appendix 10 were used for analysis.

Each database was subjected to the words recovery algorithm and the total number of column names positively identified noted. This number was then expressed as a percentage of the total number of columns. Table 3.11 provides a summary of the obtained results,

Table 3.11 Evaluation Results of the Words Recovery Algorithm

Database Name	Number of Tables	Total Number of Columns	Number of Columns Identified	Number of Columns NOT identified	% Identified	Abbreviations NOT Recognized
HR_Db	7	36	36	0	100	none
Order_Entry_Db	6	35	33	2	92	NLS (appearing twice)
Retail_Management_Db	5	70	69	1	98	pos (point of sale)
Phone_Company_Db	8	54	46	8	85	Key (instead of ID; appears 8 times)
Stock_Exchange_Db	7	60	48	12	80	Key (instead of ID; appears 12 times)
Microsoft_Northwind_Db	8	72	72	0	100	none
				Mean Accuracy	**92.5**	

The average performance of this algorithm was found to be about 92.5%. Further to the above results, an analysis for establishing whether the reconstructed words had a correlation with the contents of respective columns was done.

It was found that in all the reconstructed words, there was a correlation between the semantics of the recreated words and the intended meaning (such as storing values with a meaning similar to the recreated words). For example a column with an original title *'DoB'* or *'dateOfBirth'* and which was decoded as 'date of birth' would contain various dates of birth for various records.

In general the OWoRA algorithm is suitable for reconstructing words from database schemas where users have no knowledge of the nomenclature of the database elements. It is however assumed that the nomenclature employed belongs to one of the ten most prevalent styles identified in this research. However in certain situations, the user of the database access software may have the opportunity to specify the nomenclature of the database elements or the database developer may explicitly provide the naming style adopted for a particular application. In these cases, the words extraction algorithm may be altered to a case-base structure where specific procedures, such as **`function`** `under(),dash(),dot()`among others as described in the `OwoRA` algorithm, are individually loaded depending on the nomenclature specified. This may serve to enhance the efficiency in such applications.

In summary this field study established that there is a finite set of patterns that database schema authors' use in representing database schema object names. It has been established that although most database management systems allow many different nomenclatures, only about ten categories are dominant which were grouped into three clusters as earlier described. The ten categories formed the basis for the word reconstruction algorithm described in figure 3.36.

It was also found that there is correlation between the semantics of the recreated words and the meaning of the ontology's written-form. It was also established that the accuracy of the of words recovery algorithm was 92.5 %. Failure to reach the 100% mark was as a result of the usage of unidentifiable acronyms and abbreviations. Further, some abbreviations do not relate to the word forms that would ordinarily be used to represent the words. These require human intervention while decoding and therefore making gazetteer formation a human assisted process. Although other reviewed gazetteer formation processes are human-assisted, their accuracy levels were not reported and therefore difficult to provide a comparative analysis.

Baseline algorithms are not readily available because most algorithms assume that the ontology contains entries that have full unabbreviated lexicon as opposed to concatenations of schema data in

relational databases. Other ontology inspired methods such as semantic web solutions do not suffer from this problem and ontology entries are matched directly to NL. See work reported in Kaufmann, Berstein & Fischer (2007), Munir, Odeh & McClatchey, (2008), Tablan, Damljanovic & Bontchev, (2008).

3.5 FSM and Gazetteer Design

The conceptual OCM model illustrated in figure 3.1 and 3.2 envisages a feature space model (FSM) and gazetteer model. The design of these key components was guided by literature analysis and was tested by building these into the OCM prototype. The position of the two schemas in the conversion process is shown in figure 3.37.

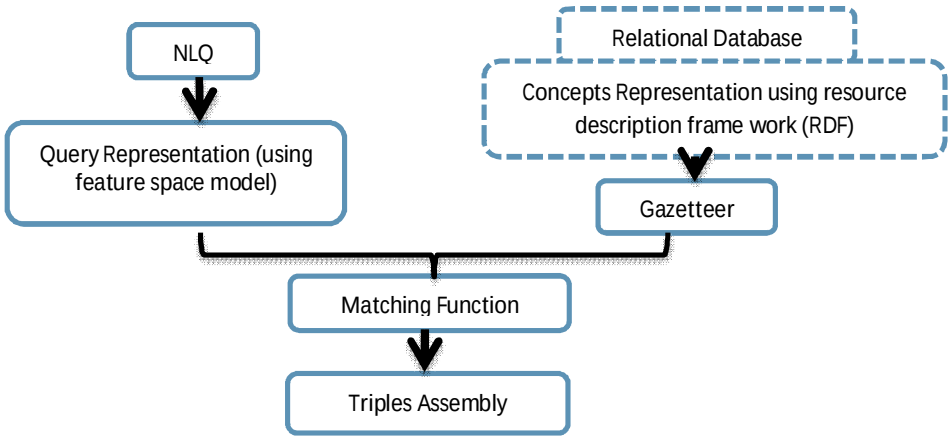


Fig. 3.37 Concepts Processing

Section 3.5.1 describes the structures of these two schemas, the feature space model and the gazetteer.

3.5.1 Feature Space Model (FSM)

As envisaged in the conceptual model NLQs are normalized, tokenized, lemmatized, stemmed and tagged with parts of speech. This is further followed by phrase formation, collocations and terms discovery in the same module. A challenge arises in the design of a schema that holds these elements in a domain and language independent manner. The main decision to be made was determination of the types of linguistic features to be stored and how they are to be stored (schema).

In designing the FSM a consideration was made of the initial, intermediate and final linguistic components that needed to be stored. From the NL query processing framework presented in figure 3.13 and the ontology elements processing algorithm presented in figure 3.36 the end products of both processes are meaning-bearing phrases that form the backbone of the SPaRQL query structure presented in section 3.3.10. Phrases from NLQ were stored in a feature space model while those from the ontology were stored in a gazetteer. Section 3.5.1 discusses the process used to design the feature space model while section 3.5.2 discusses the process of designing the gazetteer.

3.5.1.1 Experimental Investigation of Root versus Stem on Performance

FSM holds the phrase-chunks that need to match one or more of the gazetteer's phrase chunks so that a concepts' triple that is relevant to a user's request is formed. In this work matching of the FSM and gazetteer elements was through basic string matching. Phrases may be stripped to the stem or to the root level before a matching function is applied. The root of a word is the primary lexical unit and carries the significant semantic content whereas a stem is part of a word where affixes are attached to give different meanings. Selection of either of these methods results in different performance rates. Recall is a performance measure that indicates the number of cases that the system is able to answer positively and should have been answered, while precision is the number of the positive cases answered having been expressed as a percentage of the total numer of queries answered. It is importat to maximize F-score, which is the harmonic mean of the precision and recall so that usability can be increased.

A comparative experiment was therefore necessary to determine the approach with a better F-score and therefore recommend storage of the respective word form within the gazetteer and FSM. Two comparative experiments for each of the two case study languages were set up as follows.

Experiment A (Stemming Vs. Root – English Queries)

The OCM prototype whose detailed construction information is found in chapter 4 was constructed using the Lancaster stemmer (Paice, 1990) as the stemming tool. Another stemmer that was tested but not selected due to lower performance was the Porter's Stemmer (Porter, 1980). In the experiments a test set comprising 30 randomly selected questions in the UoN Masters Programs Coordinator's query set was used and the average recall value recorded.

A similar experiment was done but with the NLTK WordNet lemmatizer instead of the Lancaster stemmer. The average recall value was also recorded.

Table 3.12 shows a comparison between the Wordnet lemmatizer found in NLTK and Lancaster stemmer. The WordNet lemmatizer only removes affixes if the resulting word is in its dictionary. Lemmatizers are a good choice when compiling the vocabulary of some texts and or a list of valid lemmas. On the other hand stemmers truncate words according to some predefined algorithms such as those described in Paice (1990) or Porter (1980).

Table 3.12 Comparison of Stemmer and Lemmatizer

	Method	
	Lancaster Stemmer	WordNet Lemmatizer
Query (Original String is named Raw)	raw = """DENNIS: Listen, strange women lying in ponds distributing swords is no basis for a system of government. Supreme executive power derives from a mandate from the masses, not from some farcical aquatic ceremony."""	raw = """DENNIS: Listen, strange women lying in ponds distributing swords is no basis for a system of government. Supreme executive power derives from a mandate from the masses, not from some farcical aquatic ceremony."""

Python code procedures used	>>> tokens = nltk.word_tokenize(raw)
	>>> porter = nltk.PorterStemmer()
	>>> lancaster = nltk.LancasterStemmer()
	>>> [lancaster.stem(t) for t in tokens]
		>>> tokens = nltk.word_tokenize(raw)
		wnl = nltk.WordNetLemmatizer()
		>>> [wnl.lemmatize(t) for t in tokens]

Typical Output	['den', ':', 'list', ',', 'strange', 'wom', 'lying', 'in', 'pond', 'distribut', 'sword', 'is', 'no', 'bas', 'for', 'a', 'system', 'of', 'government.', 'suprem', 'execut', 'pow', 'der', 'from', 'a', 'mand', 'from', 'the', 'mass', ',', 'not', 'from', 'som', 'farc', 'aqu', 'ceremony', '.']	['DENNIS', ':', 'Listen', ',', 'strange', 'woman', 'lying', 'in', 'pond', 'distributing', 'sword', 'is', 'no', 'basis', 'for', 'a', 'system', 'of', 'government.', 'Supreme', 'executive', 'power', 'derives', 'from', 'a', 'mandate', 'from', 'the', 'mass', ',', 'not', 'from', 'some', 'farcical', 'aquatic', 'ceremony', '.']
Remarks	The stemmer truncates the words according to algorithm described in Paice(1990)	The WordNet lemmatizer only removes affixes if the resulting word is in its dictionary.

In both the experiments performance was calculated using the following formula,

$$Precison = tp/ tp+ fp$$

$$Recall = tp/ tp+ fn$$

$$Accuracy = tp+tn/tp+fp+tn+fn$$

$$F\text{-}score = 2(Precision\ X\ Recall)/(Precision + Recall)$$

Where *tp* and *fp* represent the true and false positives respectively and *tn* and *fn* represent the true and false negatives respectively

The results are presented in table 3.13.

Experiment B (Stemming Vs. Root – Kiswahili Queries)

Similar experiments as in A above were done but the query set was changed to 30 randomly picked Kiswahili questions from the farmers' query set. A Kiswahili lexical database (construction details provided in chapter 4) with stems was used and the average performance values recorded. The lemmas (root forms) database was then applied to the prototype and the same Kiswahili questions as above used.

Here are examples of the resulting output from the lemmatization and stemming processes for Kiswahili queries.

Surface form → *Kuku wakitetemeka ni wagonjwa?* (Are shivering chicken sick?)

Lexical form → [*kuku*] [*tetema*] [*ni*] [*mgonjwa*] ([are] [chicken] [shiver] [sick])

Stemmed form→ [*kuku*] [*tetem-*] [*ni*] [*-gonj-*] {prefixes: *wa-ki-* for *tetem-*; *-eka* for *tetem-*}

The average performance was determined. The results are presented in table 3.13.

Results and Analysis for Experiments on Root vs. Stem

Table 3.12 illustrates that performance values are higher when using stem formation process than when using root formation process regardless of the language used.

Table 3.13 Recall, Precision and F-Score Values for Root and Stem

Language	Method	Recall	Precision	F-Score
English	Stem	0.74	0.87	0.78
	Root	0.60	0.79	0.69
Swahili	Stem	0.68	0.83	0.77
	Root	0.65	0.78	0.68

The results indicated that for the model to have higher recall value, stem formation would be a better intermediate process than lemmatization.

As a result of this a decision of storing stemmed word forms was made, thus the FSM is designed to store stemmed word forms.

3.5.1.2 Storing Linguistic Components beyond Nouns

Concepts are more diverse than simple nouns and noun phrases as identified by Krishnamurthy & Mitchell, (2011). Studies for Kiswahili language have also revealed diverse patterns of term formations (Sewangi, 2001). Terms which include collocations represent concepts and therefore ought to be accounted for in the concepts discovery process and also storage. Furthermore nouns identification should include noun patterns such as those identified for Kiswahili by Ohly (1982) which include Norminalized verbs, Deverbative head with noun complement, Combination of nouns, Noun and adjectives, Nouns with –a connector and Nouns with –a connector and a nominalized verbs.

Most state-of-the art NL access methods to ontologies rely on conversion of queries to tokens from which the tokens are assembled into what is commonly known as bag-of-words. A bag-of-words means a collection of tokens created from either the ontology or the NLQ and the tokens do not relate to each other. This is evident in systems such as NLP-Reduce (Kaufmann, Berstein, & Fischer, 2007), Questio (Tablan, Damljanovic, & Bontchev, 2008), Freya (Damljanovic, Agatonovic, & Cunningham, 2010) among others. Tokens can be organized as phrases within a data structure after being extracted from texts through phrase-chunking procedures such as regular expressions. Considering that the NLQ and the ontology in this work are processed in methods stipulated in sections 3.3.9 and 3.4.7 which both result in phrase chunks, it would be desirable to store the phrase chunks as well.

Figure 3.38 shows a typical segment of python code that defines a noun-phrase chunk,

```
patterns = """
       NP: {<DT|PP\$>?<JJ>*<NN>}
     {<NNP>+}
     {<NN>+}
     """
NPChunker = nltk.RegexpParser(patterns) # create a chunk parser
# NP is a noun phrase; DT is determiner, PP is possessive, JJ is
adjective, NN is noun and NNP is a pronoun
```

Fig. 3.38 Python Code for describing patterns of Regular Expressions

The effect on the performance occassioned by introduction of phrase chunks into the FSM schema needed to be established. A comparative experimental investigation was suitable for determining the difference in performance between the two scenarios, namely bag-of-words and bag-of-words with phrases. The phrases that were stored included noun-phrase chunks, prepositional phrases and collocation terms (through patterns identified in Sewangi 2000 and found in appendix 5). The performance measures used were recall and accuracy. The experiments were set as follows,

Experiment A: Bag-of-Words vs. Concept Patterns –English Queries

The OCM prototype was constructed with FSM holding bag-of-words. A test set comprising of 30 randomly selected questions in the UoN Masters' Programs Coordinator's query set was used on the prototype and the average precision, recall and F-score values recorded.

Similar experiments were done where the NLTK Phrase-Chunker was used to generate phrase chunks which were stored along with the tokens previously stored as bag-of-words. A detailed description of how the NLTK was configured is given in section 4.2.2.3. The average precision, recall and F-score values from these set-ups were recorded and are summarized in table 3.13.

Experiment B: Bag-of-Words vs. Concept Patterns – Kiswahili Queries

Similar experiments as in A above were done but the query set was changed to 30 randomly selected Kiswahili questions from the farmers' query set. The NLTK tool and its RegExp libraries were used for these experiments. The average precision, recall and F-score values were determined when the FSM was designed to handle bag-of-words only and when extended to handle concepts in form of phrase chunks. The results were recorded and are summarized in table 3.14.

Results for Experiments on Bag-of-Words vs. Concept Patterns English and Kiswahili Queries

Table 3.14 shows that recall, precision and F-score values from the FSM built to hold both words and phrases is consistently higher regardless of the language used.

Table 3.14 Recall, Precision and F-score Values for Bag-of-Words and Concept Patterns

Language	FSM Contents	Recall	Precision	F-Score
English	Bag-of-Words	0.70	0.80	0.74
	Concept Patterns & Bag-of-Words	0.74	0.90	0.81
Swahili	Bag-of-Words	0.68	0.83	0.75
	Concept Patterns & Bag-of-Words	0.77	0.88	0.82

The results from these experiments indicated that the FSM built to hold both bag-of-words and concepts detected through patterns or regular phrase chunkers was a better choice for the OCM model.

3.5.1.3 Other Linguistic Components to be Stored

Analysis of the five query sets described in 3.3 showed that users occasionally use synonyms. Synonyms were observed for both objects and relationships (subject/object and verb). For example the predicates ‘I wish’, ‘I’d like’, ‘show me’, ‘are there’, ‘what is’, ‘who might offer’ etc. all refer to the same predicate, that is ‘list’. ‘Client’ and ‘customer’ objects or subjects refer to the same semantic category. Another observation made is the usage of hypernyms to refer to objects. For example ‘kuku’ (chicken) is a general term used to refer to ‘kuku wa mayai’ (layers) and ‘kuku wa nyama’ (broilers). If a query is posed on information on broilers, the general information on chicken should also be provided.

Finally enumeratives (e.g. how many), superlatives (e.g. heavier than) and time enquiry (e.g. when) where found abundant within the five query sets. Since these words have more implications in terms of processing requirement in order to obtain the meaning, it was found necessary to annotate them if found within a query. A two level annotation was used. A one ('1') represents superior while a zero ('0') represents an inferior. For example in the sentence 'John is younger than James' the word younger is annotated with a '0' and in the sentence 'Blood is thicker than water' thicker is annotated with '1'.

3.5.1.4 Structure of FSM

An illustrative example of the feature space model for the query, ***"Nipatie majina ya miji ambako wafanyikazi wanatoka?",*** *(Give me the names of cities where employees come from?)* and the target database are shown in figures 3.12 and 3.13 respectively.

			Word Processing							PhraseChunk		
Basic Linguistic Features	**Sequence #**		1	2	3	4	5	6	7	1	2	3
	Surface Form		Nipatie	majina	ya	miji	ambako	wafanyikazi	wanatoka	Phrase 1		
	Stem		**~pati~**	**~jina**	**ya**	**~ji**	**amba~**	**~fanyikazi**	**~toka~**	**See note 1**		
	Synonyms		...			eneo; mahali						
	Hypernyms[2] (general)		….		…		….	…				
	Hyponyms[2] (specific)		….	…..	…. .	…..	….	….				
Tags	**POS [3]**		VB	NN	JJ	NN	PN	NN	VB	-	-	-
Extras[4]	**Superlative**	**S**	0	0	0	0	0	0	0			
	Count	**C**	0	0	0	0	0	0	0			

NB:

1. Phrase chunk is formed from stems on the left
2. Hypernyms is a more general reference eg musical instrument" is a hypernym of "guitar"; Hyponyms is a more specific term eg Dog is a hyponym of animal.
3. POS tags denote: VB= Verb ; NN = Noun ; JJ = Adjective ; PN = Pronoun
4. Annotation of words in various categories eg word is superlative or not ('1' or '0'); Requests Counting or not (How many?)

Fig. 3.39 Feature Space Model for Query Representation

3.5.2 Gazetteer Formation Process

In traditional NLP applications, a gazetteer is a list of recognizable words or phrases that need to be compiled from a text. In this research the gazetteer was derived from an ontology derived from a relational database. As discussed in the literature, automatic discovery of mappings between ontology and RDBMS has been successful and many state-of-the-art tools developed. In this work the datamaster tool (Csongor, Martin, & Samson, 2009) was used. It was selected on the basis of its good performance and ease of integration with Protégé, the ontology building program that was used in this research.

Section 3.4.7 presented an algorithm that was used to reconstruct words from the ontology. Typically all concepts within a database that may be of interest to a user should be recognized and subsequently stored in a gazetteer. The gazetteer needs to hold information about the concepts and also facilitate the matching function that operates between the FSM described in section 3.5.1 and the gazetteer. Figure 3.40 illustrates processes leading to the formation of the gazetteer

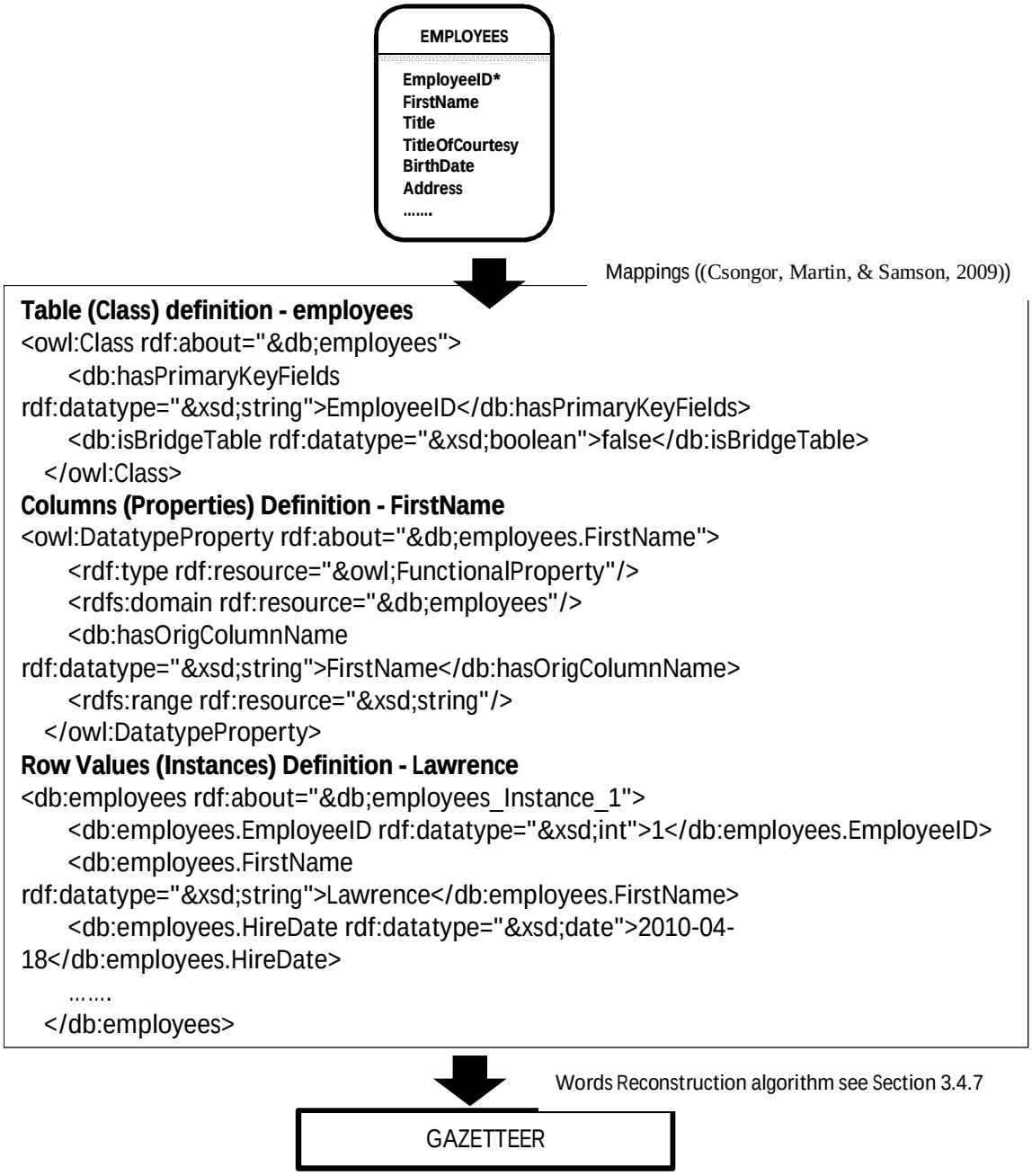


Fig. 3.40 Processes Leading to the Formation of Gazetteer

The relation (or set of relations in a multi-relation database) is automatically mapped to an ontology. The ontology elements are then extracted and converted into a gazetteer such as the one shown in Fig 3.41. The class, property and instance names are first normalized using algorithms explained in

section 3.47. In cross lingual querying, translation is done at the gazetteer stage and hence the translations are also contained in the gazetteer.

Ontology Concept	Normalized	Stem	Translation (Google Translate	Corrected Manual Translation	Stem of Translation	TYPE
Address	address	address	mitaani*	anuani	anuani	Property
BirthDate	birth date	Birth date	tarehe ya kuzaliwa	-	tarehe ya ~za~	Property
employeeId	employee identification	employ~ identif~	mfanyakazi kitambulisho	-	~fanyakazi ~tambu~	Property
employees	employees	employ~	wafanyakazi	-	~fanyakazi	Class
FirstName	first name	fist name	jina kwanza	-	jina la kwanz~	Property
title	title	title	haki miliki*	cheo	cheo	Property
TitleOfCourtesy	title of courtesy	title of courtes~	haki ya cheo*	cheo cha heshima	cheo cha heshim~	Property
Lawrence	Lawrence	Lawrence	Lawrence	-	Lawrence	Instance
.....	.....		.....	-		
customers	Customers	customer	wateja	-	~teja	Class
employees	employees	employ~	wafanyakazi	-	~fanyakazi	Class
City	city	cit~	mji	-	~ji	Property
CompanyName	company name	compan~ name	jina la kampuni	-	jina la kampuni	Property

Fig. 3.41 Structure of Gazetteer with Sample Data

3.6 The OCM Architectural Model

In this section the architectural model is presented. The purpose of the architectural model is to direct attention at an appropriate decomposition of the system without delving into details. In the architecture illustrated in figure 3.15. the system accepts the user input in the form of a full unrestrained sentence or key phrases and words. Raw text is subjected to linguistic processing that

involves tokenizing, stemming, POS tagging and phrase formation as earlier explained. On the other hand ontology elements relating to class and property names as well as instances are normalized and stemmed.

.

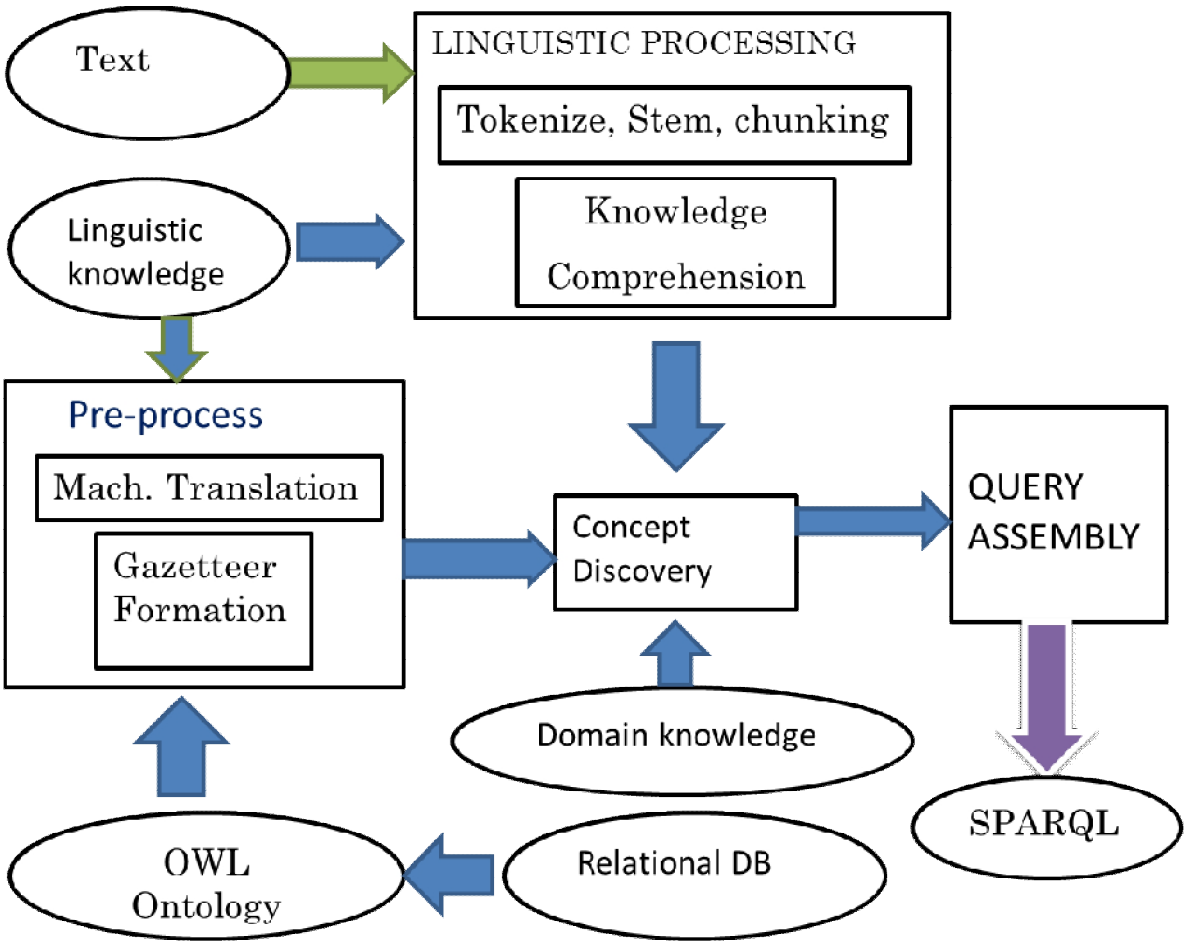


Fig. 3.42 Architecture for Ontology-based NL Access to DBs (ONLAD)

3.7 The Algorithms

3.7.1 Semantically Augmented Concepts Matching Approach(SACoMa)

The matching function takes the FSM and Gazetteer as input and generates a set of concepts not necessarily arranged in any order. This process is re-illustrated in figure 3.43.

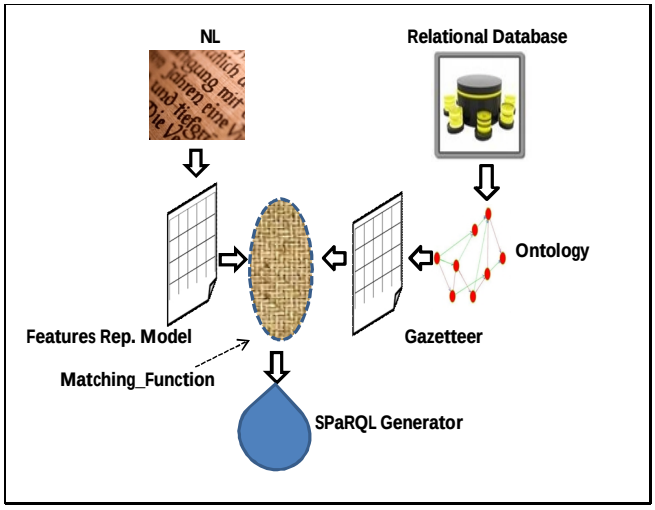


Fig. 3.43 Location of Matching Function in OCM Approach

The concepts that match from the two representation schemas form the backbone of the generated SPaRQL query. As established from experimentation, a lexical-level keyword-based matching method with lemmatization and improved by stemming was selected. The Levenshtein algorithm calculates the least number of edit operations that are necessary to modify one string to obtain another string. Levenshtein algorithm, also called edit-distance was selected for calculating distance between the two strings that is, the query concept and the ontology concept.

A zero edit distance means that only perfectly matching strings are identified. This means that the model returns few but accurate pairs thereby attaining high precision levels. Recall on the other hand is hampered. If the edit distance is increased, precision decreases but recall increases. The optimum edit gap needed to be established experimentally.

The Python implementation of Levenshtein distance calculator used in this research was adopted from **(Korokithakis, 2008)** and is shown in figure 3.44.

```
def levenshtein_distance(first, second):
    """Find the Levenshtein distance between two strings."""
    if len(first) > len(second):
        first, second = second, first
    if len(second) == 0:
        return len(first)
    first_length = len(first) + 1
    second_length = len(second) + 1
    distance_matrix=[[0] * second_length for x in range(first_length)]
    for i in range(first_length):
       distance_matrix[i][0] = i
    for j in range(second_length):
       distance_matrix[0][j]=j
    for i in xrange(1, first_length):
        for j in range(1, second_length):
            deletion = distance_matrix[i-1][j] + 1
            insertion = distance_matrix[i][j-1] + 1
            substitution = distance_matrix[i-1][j-1]
            if first[i-1] != second[j-1]:
                substitution += 1
            distance_matrix[i][j]=min(insertion,deletion,substitution)
    return distance_matrix[first_length-1][second_length-1]
```

Fig. 3.44 Python Implementation of Edit-Distance Calculation

As observed from data collected from the surveys reported in 3.3 equivalent concepts in NLQ and ontology elements are at times represented by different strings and therefore concept matching goes beyond simple string matching. The Levenshtein algorithm was enhanced through techniques borrowed from ontology matching strategies specifically semantic-based strategy as explained in section 2.6.4. The Semantic matching strategy combines integration of lexicon-based matching with the meaning of the words. This means that words identified to be semantically equivalent but having different surface forms are matched based on this fact.

For example the concept *'jimbi amekomaa'* (mature cock) is semantically equivalent to *'jogoo aliyekomaa'* (mature cockerel). This implies that *jimbi* should map to *jogoo* before Levenshtein mapping is applied. Semantic mapping was achieved through incorporation of a lexical database that included synonyms at the FSM before matching. For English queries Wordnet (Miller G. , 1995) was used while the Kiswahili lexical database whose construction is described in section 4.2.2.2 was used. This algorithm assumes that the lexical database is large enough and contains most synonyms of words.

3.7.2 Structured Query Generator Function

The various 'concepts' generated by the matching function form a set of unordered strings. The query generator's task is to organize these 'concepts' into a structured query.

Section 3.3.9.4 identified that meaning-bearing components have a tri-partite relation which may be formed between the three primary components (subject, verb and object) or, any two of these components and a modifier (or an interrogative of either) or any of the primary components and its modifiers which may appear as phrases such as a prepositions. If any of these meaning bearing components (which are stored in FSM) matches that in the gazetteer, then they qualify to be included in the SPaRQL query.

Section 3.3.10 described how triples of concepts are formed through the kernelization process. The query is reduced to a set of triples which collectively represent the original query meaning. The triples are of the format `?element1 ?element2 ?element3.` An illustrative example of the query *'What is the phone number of the customer whose ID is 1'* was provided. This resulted into two triples and a filter as shown below

```
?customer ?phone_number  ?Variable1
?customer ?id_number  ? Variable2
```

The first element in the set is the one that corresponds to the table name while the second relates to the column name. The gazetteer which contains an annotation of the type of concept, whether class, property or instance, provides for the identification and assignment of these elements.

These are the basic building blocks of the SPaRQL query. The query generator function organizes these into a full SPaRQL query by following templates such as the one shown in figure 3.45.

```
PREFIX alias_name: <http://www.URL#>
SELECT  ?Attribute1 ?Attribute2……?AttributeN
WHERE{ ?Subject db:Predicate ?Object.
     ?Subject db:Predicate ?Object.
     ?.............................
     ?Subject db:Predicate ?Object.

FILTER(?Attribute = object)}
```

Fig. 3.45 Template of Query Generator Function

The alias name (the word appearing after the key word 'PREFIX') is pulled directly from ontology descriptions. The attributes to be returned by the query (attributes appearing after SELECT keyword) are picked from the query's triple set, where specifically the middle element (`element2` in the triple) is selected.

The triples body (appearing within WHERE clause) are formed by heaping the triple sets ("`?element1 ?element2  ?element3`")

In section 3.3.10 it was stated that when specific row values are mentioned, the user's intention is to constrain the number of records (rows) returned. In SPaRQL this is achieved through the FILTER command. The general syntax for this is,

```
FILTER (?Variable = "value")
```

In the example query *'What is the phone number of the customer whose ID is 1'* there is a mention of a specific row value (instance) and hence a `FILTER` clause is required. The additional clause then becomes,

```
FILTER (? id_number = "1")
```

The full SPaRQL query is thus given as shown in figure 3.46.

```
PREFIX moon: <http://www.owl-ontologies.com/NewNorthwind#>
SELECT  ?orderID ?CustomerID ?CompanyName ?shipDate
WHERE{ ?orders db:OrderID ?OrderID.
      ?customers db:CustomerID ?CustomerID.
      ?orders db:CustomerID ?CustomerID.
      ?customers db:CompanyName ?CompanyName.
      ?orders db:OrderID ?orderID.
      ?orders db:ShippedDate ?shipDate
FILTER(?CustomerID = 1)}
```

Fig. 3.46 Example of a Generated SPaRQL Query

3.7.3 Discovering Implicit Concepts

In order to demonstrate how implicit concepts are discovered a representative sample database, 'Northwind database' by Microsoft group is presented in figure 3.47.

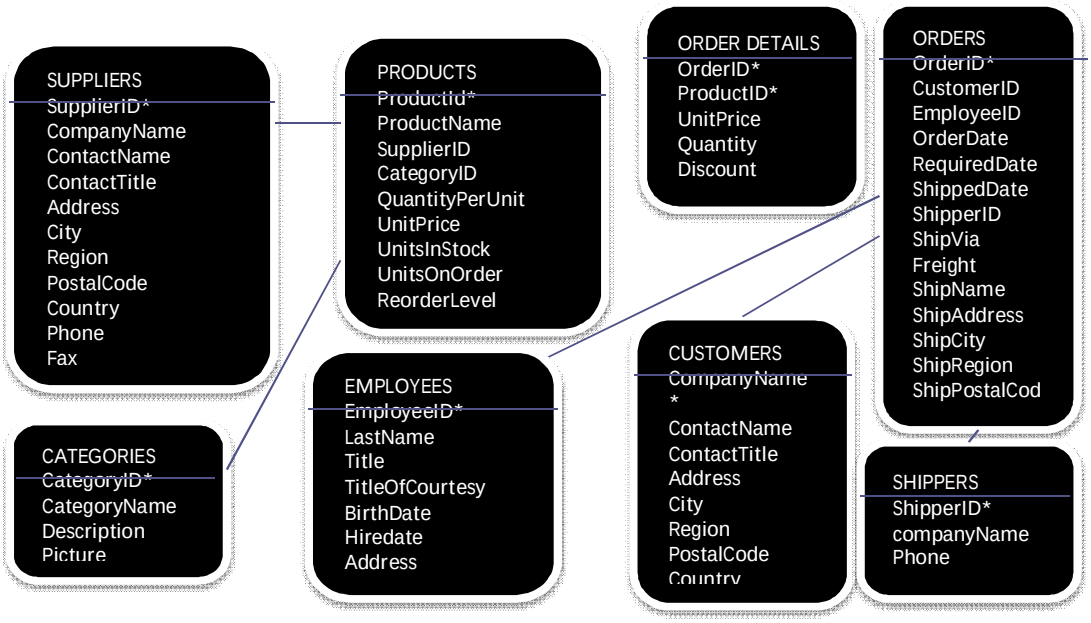


Fig. 3.47 Example from Microsoft Northwind Sample DB (Microsoft, 2004)

The above database shows a certain company's database where the company keeps an inventory of its customers, suppliers and the products they sell. Further the customers can make orders which can be transported by shippers whose information is also kept in the database. All information regarding to employees is also maintained.

The examples so far used are all explicit in that they have direct mentions of properties or classes. In other scenarios there is no direct mention but implied concepts in a rather implicit manner. The model provides for this implicit concepts-discovery by performing simple inference.

Consider this illustrative Example.

Sw: Bidhaa gani ambazo huja kwa chupa?

{En: Which products come in bottles?"}

In this case the following happens;

'*Bottles'* is stripped to *'bottle'* which in turn maps to **instance** '*bottled'* which is found within the ontology as an instance.

Since *Bottled* has been tagged in the Gazetteer as an instance of *categories* class through data type property *Description*, we discover two additional ontology concepts that is,

Categories class and

Description **property**

The triple becomes

?categories db: categories.description ?description

FILTER(?description = "bottled")

The *FILTER* is necessary for instantiating a class's property value and is applied where there is direct mention of an instance such as *'bottle'* in this example.

Interrogatives

Further the query has an interrogative of type "which" that suggests an identification problem. By default we return *instances* of classes with properties related to identification of the class; that is name, identification or both if present in that particular class.

Hence the triples would be:

```
?products db: products.ProductID ?ProductID.
?products db: products.ProductName?ProductName
```

3.7.4 Key Attributes (Foreign Key)

When dealing with two classes related via a foreign key a heuristic was developed. This is stated as follows, *"when two tables are involved in reply to a query, we introduce two triples one from each participating class and both having the common property"*.
In the example we have:

```
?products db: products.CategoryID ?CategoryID.
?categories db: products.CategoryId ?CategoryID.
```

This heuristic was validated by way of analyzing 20 queries that would require the answer to be generated from at least two tables. The heuristic was applied and the two possible triple sets generated manually. The two triples were then assessed for the type of answer they would jointly generate when applied to an OWL database and the results verified against the expected correct result. In all the cases selected for analysis, the heuristic was found to hold true. Some sample results are found in appendix 11.

3.7.5 Triples Assembly

Figure 3.48 shows the sets of triples obtained from the simple statement *"which products come in bottles?"*

```
{ ?products db: products.ProductID ?ProductID.
    ?products db: products.ProductName?ProductName.
    ?products db: products.CategoryID ?CategoryID.
    ?categories db: categories.CategoryId ?CategoryID.
    ?categories db: categories.Description ?Description.
 FILTER( ?Description = "bottled") }
```

Fig. 3.48 SPaRQL Query that handles Implicit Concepts and Foreign Keys

A further point to note is that in database ontologies it is preferable to use both class and property names so as to minimize ambiguity in case multiple classes are using similar property names as in the example customer's phone and supplier's phone.

3.7.6 Overall Algorithm

The processes described in this chapter were summarized into a high level algorithm which is presented in figure 3.49.

The OCM Algorithm

```
1. Assemble tokens list (words and phrases)
2. Comprehend ontology-strange terms (those without a lexical
   match in the ontology) that may be synonyms, hypernyms,
   hyponyms or even known jargon
3. Assemble List of Concepts (Tokens which match ontology
  elements)
- Explicit concepts
- Implicit concepts
- Concepts include matches to object and datatype properties,
  classes, instances, rdfs:labels, rdfs:comments  and special
  categories (superlatives and enumeratives).
4. Assemble Triples
- Determining participating relations along with Primary and
  Foreign keys
 -Identify User required properties and constraining instances
  and their related properties (Filters)
5. Assemble SPARQL Query
 - Progressively heap relevant triples until exhausted from
  user input
6. Execute the query on the Protégé reasoner.
```

Fig. 3.49 The Overall OCM Algorithm

3.8 Prototype and Resources Used

Availability of open source semantic web resources such as Protégé, Jena, and Sesame that edit and store ontologies as well as open source DB-ontology link tools such as Datamaster (Csongor, Martin, & Samson, 2009) of Stanford University have greatly informed the prototype development activities carried out in this research.

3.8.1 Prototype Overview

An overview of the main processes within the prototype and software used is illustrated in figure 3.50.

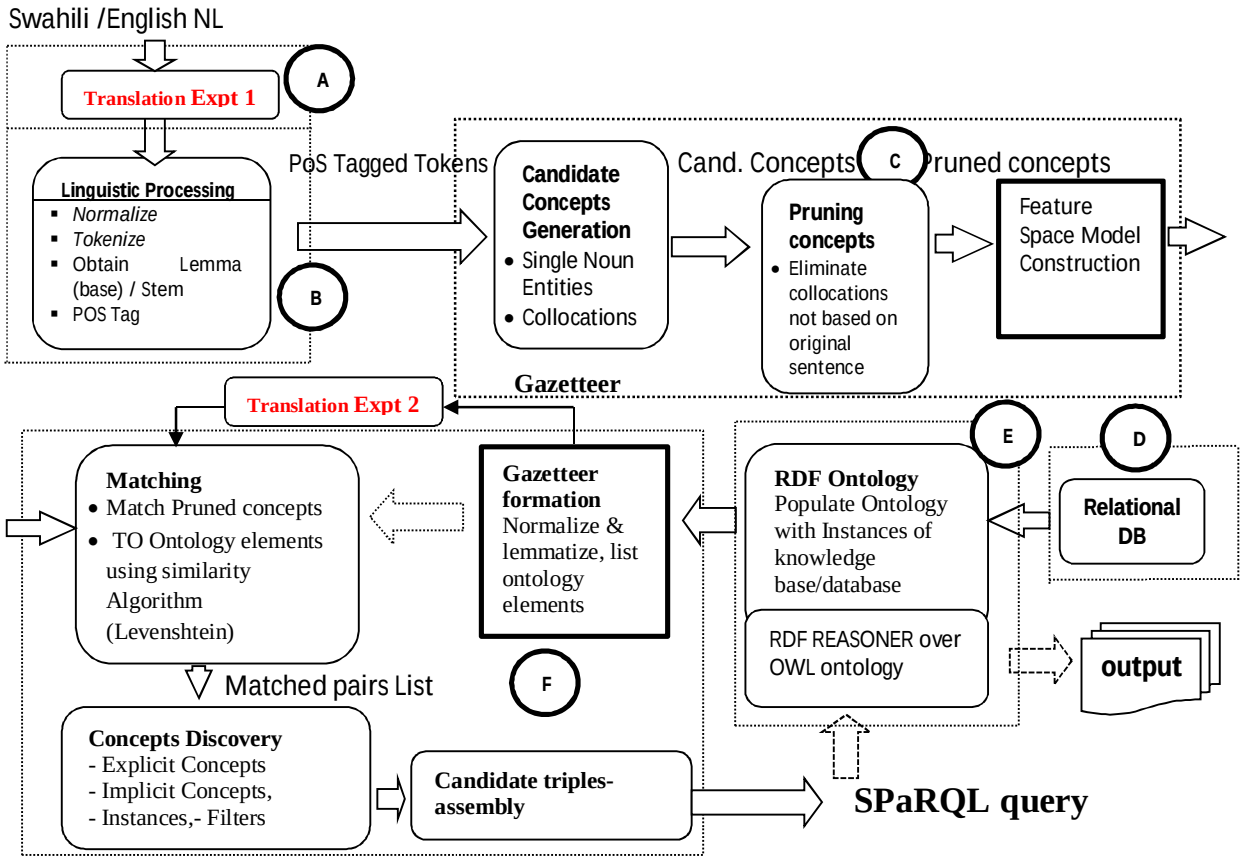


Fig. 3.50 Structural Design of Prototype

3.8.2 Resources

Figure 3.50 shows the main modules labeled A to F and whose purpose is described shortly.

3.8.2.1 Module A: Bilingual List for Cross-lingual Solution

In the model presented in section 3.6 and prototype shown in figure 3.50 it can be seen that if the language of querying is different from the language of naming database schema objects then translation is needed. This is the cross-lingual problem that the model has to grapple with. Translation may occur immediately after the NL input or during the gazetteer formation as a pre-process. Experiments were carried out to find which of these two stages gives better results (translation experiments on figure 3.50). Further three approaches to translation were experimented

on. These are dictionary-based methods in which Carabao[11] tool was used, transfer method in which Apertium[12] tool, was used and statistical machine translation in which Google Translate[13] was used. A Dictionary based method where a bilingual word list was created and used was selected. This decision was arrived at on the basis of less configuration work and better performance recorded during test runs.

3.8.2.2 Module B: Linguistic Processing

The linguistic processing module was developed as a composite of many python sub-modules and was implemented using the natural language tool kit, NLTK[14]. It has several sub-modules that perform various functions for English and Kiswahili language such as *normalizing, tokenizing, lemmatizing/stemming and POS Tagging, noun phrase chunking* and *noun entity/collocation identified*. The module has access to several linguistic resources such as WordNet (Miller G. , 1995) and the Lancaster Stemmer[15] (Paice, 1990) for English which easily integrate with NLTK.

For ease of performing the experiments the modules were organized in a pipeline referred to as the test bed. A language selector module was implemented as the start point of the pipeline.

a) Language Detection

Several methodologies for implementing the language detection module that can be integrated with NLTK's python code are available. These include stop-words frequency model, bi-gram and tri-gram frequency models among others. For a model whose expected input is a single sentence or just key words, the stop-words frequency model would not be suitable. On a trial bases the tri-gram model was found to perform better compared to bi-gram or other N-gram models and was therefore selected for implementation.

The procedure involved creating a list of trigrams from training texts for both English and Kiswahili. The probability distribution models for the trigrams (also known as language models) for the various languages (English and Kiswahili) were generated. The python code used was adopted from an open source forum on language identification in python by Cavar (2011) namely *lidtrainer.py*. The

[11] Carabao is a completely open machine translation toolkit. It allows creation of your own machine translation interlingua-modeled databases and using them on-the-fly. See http://www.uuucom.com/carabao-diy-machine-translation-kit-downloads-18340.html

[12] Apertium is a **free/open-source machine translation platform**, initially aimed at related-language pairs but recently expanded to deal with more divergent language pairs. See http://www.apertium.org/?id=whatisapertium

[13] Google Translate is a free translation service that provides instant translations between 64 different languages. See http://translate.google.com/

[14] *NLTK* is a platform for building Python programs to work with human language data. *And can be accessed at* http://nltk.org/

[15] For the code see http://nltk.googlecode.com/svn/trunk/doc/api/nltk.stem.lancaster-pysrc.html#LancasterStemmer

frequency of each trigram is the number of times it appears divided the total number of trigrams. The frequency is calculated for each trigram and then the trigrams are sorted according to the frequency in the training model. In order to determine the language model of the target text, the text is subjected to the module that determines the models and then calculates the distance between the two language models. The code that was used was adopted from Cavar (2011) and is called the *lid.py*. The distance between the two language models is obtained by getting the modulus of the difference between the probability of trigram in the training model and the target model and summing these up for all the trigrams in the target language model. The language model with the least distance is declared the identified language.

b) Normalizing and Tokenizing

Language detection is followed by a pre-process activity that involves normalizing and tokenizing with a view of standardizing all input texts to prepare them for further processing. Since normalizing and tokenizing are language independent a standard python routine was implemented. NLTK has a standard tokenizer but requires additional facilities for normalizing. The tool was enhanced by ensuring that it can recognize non-standard words like numbers, abbreviations and dates and convert them accordingly.

c) Lemmatization and Stemming

NLTK is integrated with several English Lemmatizing and Stemming tools. Two most used stemmers namely the Porter (Porter, Robertson, & Rijsbergen, 1980) and the Lancaster stemmers (Paice, 1990) were tried out. The Lancaster stemmer gave better results and therefore was selected. WordNet is the default lemmatizing tool for NLTK and was selected for the pipeline. WordNet lemmatizer uses the WordNet database to lookup lemmas. Lemmas differ from stems in that a lemma is a dictionary word, while a stem is a surface word less the affixes and not necessarily in the dictionary.

Kiswahili does not have readily available lemmatizing and stemming. As a result of this, a lemmatizer and a stemmer were developed for use in the prototype. Two widely quoted approaches in literature are data-driven methods such as the 'Memory-Based Kiswahili Morphological Analyzer' (MBSMA[16]) reported by De Pauw and Schryver (2008) and finite state transducers on two levels

[16] A demonstration system for the MBSMA-s system can be found on the AfLaT website http://aflat.org/?q=node/241).

such as SWATWOL (Hurskainen A. , 1992). These approaches perform morphological analysis and also give base form words that is, the lemmas. Another approach is the dictionary based approach. This approach has been used for the WordNet (Miller G. , 1995) lemmatizer where it uses a database to lookup for lemmas. For a limited experimental usage such as testing sample Kiswahili input sentences as is the case with the prototype being developed a look-up database was sufficient. Information from Helsinki Corpus of Kiswahili, HCS (Hurskainen A. , 2004) and the Kiswahili-English Dictionary by Institute of Kiswahili Research, University of Dar es Salaam (TUKI, 2000) which already contains Kiswahili lemma and lexical categories (POS) were used to construct a Kiswahili lexical database. HCS is a corpus with over twelve and half million words and was annotated using SALAMA (Hurskainen A. , 1999) which is a language manager that performs morphological analysis among other tasks. Only a small section of the HCS was used specifically 'Alasiri' genre whose origin was newspaper/magazine articles.

A section of the corpus is shown in figure 4.4.

```
<s>
        <w lemma="mshindi" type="N" msd="CAP 1/2-SG DER:verb (shinda)" trans="winner">Mshindi</w>
        <w lemma="wa" type="GEN-CON" msd="1/2-SG">wa</w>
        <w lemma="tatu" type="NUM" msd="NUM-INFL ORD" trans="third">tatu</w>
        <w lemma="katika" type="PREP" trans="in , at">katika</w>
        <w lemma="kinyang'anyiro" type="N" msd="7/8-SG DER:o" trans="stiff competition">kinyang'anyiro</w>
        <w lemma="hicho" type="PRON" msd="DEM :hV ASS-OBJ 7/8-SG" trans="this">hicho</w>
        <w lemma="ni" type="DEF-V:ni" trans="be">ni</w>
        <w lemma="taus" type="PROPNAME" msd="&amp;lt;CAP&gt; &amp;lt;Heur&gt; SG">Taus</w>
        <w lemma="Abdallah" type="PROPNAME" sem="AN HUM">Abdallah</w>
      <s>
```

Fig 3.51 Structure of HCS Showing Lemma, Part-of-speech Label, Translation among others

The structure of the Kiswahili-English dictionary (TUKI, 2000) is illustrated in figure 3.52. It shows how lemma, parts of speech the translation among others are represented in the dictionary. The electronic version was available.

bingwa *nm* & *kv ma-* [*a-/wa-*] 1 specialist, adept, consultant, expert: ~ *wa uchumi wa Afrika* specialist in African economy. 2 clever person, (michezo) champion, (sio rasmi) dab-hand: *Kijana huyu ni* ~ this youngman is very clever.

bingwa tapeli *nm ma-* [*a-/wa-*] quack, conman.

bin.i *kt* [*ele*] forge, counterfeit. (*tde*) **binia**, (*tden*) **biniana**, (*tdew*) **biniwa**; (*tdk*) **binika**; (*tds*) **binisha**.

binti pia **biti** *nm ma-* [*a-/wa-*] daughter, miss, girl, young lady: ~ *Juma* Juma's daughter.

binu.a *kt* [*ele*] protrude. (*tde*) **binulia**, (*tdew*)

Fig 3.52 Structure of Swa-Eng TUKI Dictionary Showing Lemma, Pos Labels and Translation

d) POS Tagging

The objective was to develop a part of speech tagger which handles the input statements. NLTK tool comes with several taggers. These taggers require to be trained on one of the corpora that comes with NLTK and that has part of speech tags. The unigram tagger for example tags each word by checking what the most frequent tag for the word is in a training corpus. There are approximately 38 sets of corpora that can be loaded onto NLTK and that one can choose from and includes such corpora as conll2000, brown, Stop words, NPS Chat, Universal Declaration of Human Rights Corpora among others. The main part of speech taggers integrate with NLTK include unigram, bigram, trigram, Regexp, affix, brill and hidden Markov model taggers.

Various taggers were tested on selected corpora. The Brill tagger having been tested on the brown corpus was selected for English inputs. The Brill tagger uses an initial unigram tagger and a set of templates usually ten. The training procedure involves importing the Brill tagger and running an initial tagger in this case the unigram tagger, and then improving the tagging by applying a list of transformation rules. These transformation rules are automatically learned from the training corpus, based on one or more rule templates. Training the Brill tagger is carried out via few steps illustrated in figure 3.53.

The tagger statistically computes the tag of each word, and then improves on the mistakes through the help of learning rules. In this way the Brill tagger successively transforms through rules an incorrect tagging of a word into a better one. Bird et al. (2008) explains that as with n-gram tagging, Brill tagger is a supervised learning method, since it requires annotated training data to figure out

whether the tagger's guess is a mistake or not. But unlike n-gram tagging, it does not count observations but compiles a list of transformational correction rules.

```
>>> brown_tagged_sents = brown.tagged_sents(categories='news') // obtain a set of tagged sentences from brown corpus in the 'news' genre

>>> size = int(len(brown_tagged_sents) * 0.9)// obtain total length of the tagged sentences

>>> train_sents = brown_tagged_sents[:size]// define that from start of tagged sentences to the 90% mark, the sentences will be used for training

>>> test_sents = brown_tagged_sents[size:]// define that from 90% mark to the end of tagged sentences will be used for training

>>> unigram_tagger = nltk.UnigramTagger(train_sents)// Define a unigram tagger and initialize with the pre-defined training sentences

>>> unigram_tagger.evaluate(test_sents)// Evaluate the unigram tagger… just to be sure that its working ok

>>> trainer = FastBrillTaggerTrainer(initial_tagger=unigram, templates=templates,
trace=3, deterministic=True)
>>> brill_tagger = trainer.train(train_sents, max_rules=10)// define the brill tagger // Train brill tagger and run it.
>>> print 'Accuracy: %4.1f%%' % (100.0 * nltk.tag.accuracy(brill_tagger, test_sents))
Accuracy: 82.6%
```

Fig 3.53 Training and Evaluation of Part of Speech Taggers

The Kiswahili POS tagger was also developed in a similar way as explained in figure 3.52 and preceding section. The Kiswahili corpus comprising of lemmas and part of speech tags was generated as a .txt file from the HCS corpus which is an xml file. The performance of the tagger was lower compared to the Data-Driven Part-of-Speech Tagger for Kiswahili (De Pauw, Schryver, & Wagacha, 2006) which is reported to have an accuracy of 98.6% against 82.6% for the combined Brill and unigram tagger. The Data-Driven Part-of-Speech Tagger was not available online nor other taggers such as the SWATWOL (Hurskainen A. , 1992) and Morfessor (Creutz, Lagus, Linden, & Virpioja, 2005), hence the decision to use the combined Brill and unigram tagger.

3.8.2.1 Module C: Concepts Generation

a) Candidate Concepts Generation (Concepts Modeling)

A term is a word or group of words used in a communicative setting to represent a concept within a domain. A term represents one concept within a domain. A term consisting of one or more words and can be categorized as a compound term or a term collocation. A Collocation is a sequence of

words or terms that co-occur more often than would be expected by chance. A phrase is used to refer to a building block of a sentence and so has a grammatical significance in a sentence. A multi-word phrase is a word group held tightly together by meaning relationships. A phrase is built around a head word (Noun, Verb, Adjective, Adverb, and Preposition) and may also have several modifiers in it. Modifiers are expressions that add details of meaning to the head word.

Phrase and terms generation from text relies on theories of chunking which are well established. The most dominant approaches include rule based methods and data-driven methods. Rule-based approaches use regular expressions while some of the successful machine learning methods include SVM-based chunkers such as YamCha[17], chunking using transformation-based learning such as the java oriented Greenwood's chunker[18] and the C++ oriented fnTBL[19] chunker and the NLTK chunkers. The default NLTK chunker is a classifier based chunker trained on the ACE corpus (Django Project, 2011). It recognizes noun phrases and named entities, such as locations, names, organizations, and only work well with an English tagger. NLTK also allows for definition and execution of either a machine learning based chunker or a regular expression chunker. Studies for Kiswahili text chunking have not been widely documented. The main challenge in machine learning chunking is in the creation of I-O-B tagged data if it does not exist, whereas the main challenge in regular expression based implementation is in the study and discovery of these patterns which is a manual task.

In the methodology applied for this work the NLTK machine learning chunker was selected for English texts on the basis of its performance and the fact that it can easily be linked to other python implemented modules. Data that is already PoS tagged and annotated with I-O-B tags for English is available. PoS and I-O-B tagged CoNLL 2000 corpus was used for training. The CoNLL 2000 corpus contains 270,000 words from the Wall Street Journal text and is already divided into training and testing portions.

In training the classifier-based machine learning chunking a four stage process was used.

[17] **YamCha** (Yet another multi-purpose Chunk annotator)is a generic, customizable, and open source text chunker oriented toward a lot of NLP tasks, such as POS tagging, Named Entity Recognition, base NP chunking, and Text Chunking and can be found at http://www.chasen.org/~taku/software/yamcha/

[18] GATE framework linkable chunker done at University of Sheffield. Details found at http://www.dcs.shef.ac.uk/~mark/index.html?http://www.dcs.shef.ac.uk/~mark/phd/software/chunker.html

[19] A fast and flexible implementation of Transformation-Based Learning in C++. Includes a POS tagger, but also NP chunking and general chunking models. Found at nlp.cs.jhu.edu/~rflorian/fntbl/

- The first stage involved a PoS tagger in which part of speech tags are assigned to a sentence. The result of this first stage is a list of tokens within a sentence that has PoS tags. The PoS tagged sentence forms the input of the next stage and that is chunk tagging.
- The PoS tagged sentences are further annotated with I-O-B tags by the help of a chunk-tagger. I-O-B tags specify if a particular token is inside, outside or at the beginning of a chunk. For example the sentence ' *Nipe alama za mwanafunzi aitwae Julius' (Give me the marks of a student called Julius)* can be broken down into several noun phrases such as 'alama za mwanafunzi'(*the marks of a student*). 'alama' can be labeled 'B' because it appears at the beginning of the noun phrase, 'za' appears inside the phrase and therefore is labeled 'I' while 'Nipe' is outside the phrase and is labeled 'O'. The various PoS tagged tokens within a sentence are annotated with the I-O-B tags by the chunk-tagger.
- The third stage was to convert these PoS and I-O-B tagged sentences into a chunk tree which is done by the chunk-parser.
- The fourth stage involved conversion of the chunk tree into actual chunks and this is done by an extractor. A tree traversal function for extracting NP-chunks in the parsed tree was defined and used to extract all n-gram chunks. Segments of the python code used for training and testing the English-text chunk parser and extractor are shown in appendix 4.

On the other hand Kiswahili does not have readily available I-O-B tagged data and an attempt to develop an I-O-B tagged corpus would require significant man-hours. However, Kiswahili has well documented regular patterns of noun-phrases which are presented by Ohly (1982) and recast by Sewangi (2001) as shown in figure 3.54 and term patterns. The phrase patterns are generalizable to many domains because they are common grammatical phrases. Another study done on patterns (Sewangi, 2001) sought to obtain term patterns in specific domains. It was demonstrated that term patterns are formed by words that function as members of a subcategory of the major categories. For example, the noun term and verbal noun act as subcategories of noun and verb major categories respectively. The study unearthed term patterns in two domains specifically health-care and literature domains. Though the patterns are domain specific one can obtain common templates that can be applied across domains.

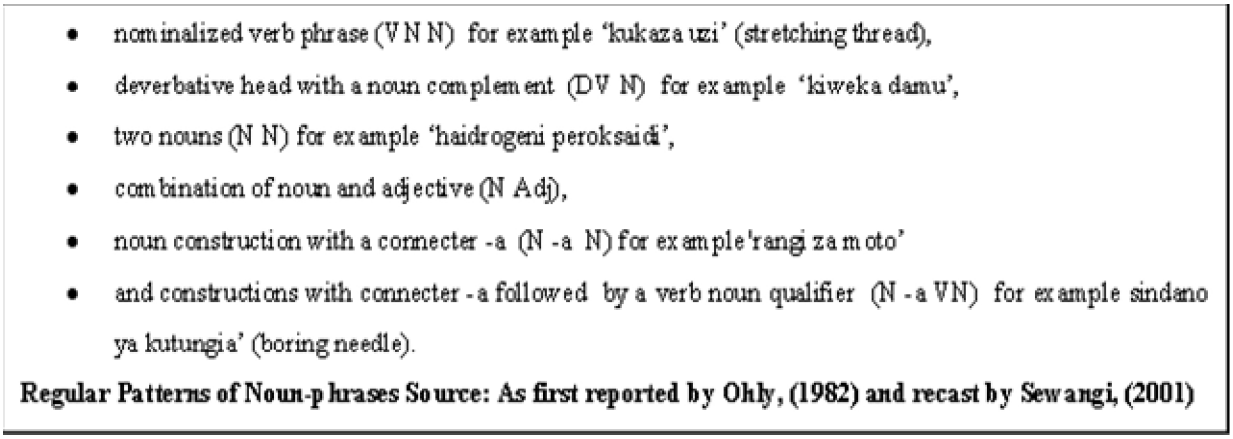


Fig. 3.54 Regular Patterns of Noun-phrases (Source: Sewangi, 2001)

The methodology applied in this study involved applying the common multi-word terms' regular expressions (see appendix 5). These templates and the Ohly phrase templates were used as regular expressions in phrase chunking. These were applied to the NLTK RegExp chunker as its regular expressions as shown in appendix 6.

b) Concepts Pruning and Feature Space Model Construction

The noun-phrases and terms generated from the section above are likely to be over generated especially due to the use of templates. Some phrases do not make semantic sense with respect to underlying concepts and therefore they need to be eliminated. The triples are pruned by eliminating triples not based on the composed semantic ontology of the database as these are less likely to yield results and ultimately assembled as SPARQL[20]. The Feature Space Model structure is described in section 3.3.2. It was implemented as an array using python.

3.8.2.2 Module D, E, F: Database, RDF Framework and OCM Tools

Module D was implemented using WampServer[21] that consists of php, MySQL, and apache while module E was implemented using Protégé[22], Datamaster and the protégé's native RDF Reasoner. Module F whose components design was explained in section 3.3.2 through 3.6.2 were implemented as python functions.

[20] SPARQL is a structured query language that can query RDF sources. See tutorial at http://www.w3.org/TR/rdf-SPaRQL-query/ and http://www.xml.com/pub/a/2005/11/16/introducing-SPaRQL-querying-semantic-web-tutorial.html?page=1

[21] *WampServer*: It allows you to create web applications with Apache2, PHP and a MySQL database. Alongside, PhpMyAdmin allows you to manage easily databases. See athttp://www.wampserver.com/en/

[22] A free open-source Java tool providing an extensible architecture for the creation of customized knowledge-based applications. See ***protege****.stanford.edu/*

3.9 Chapter Summary

This chapter has presented details of research activities leading to the design of the Ontology Concept Model (OCM). In particular it has detailed the activities of several case studies aiming at linguistic characterization of NLQs for both Kiswahili and English and a study of nomenclature trends of database elements'. Table 3.15 shows a summary of the research objectives, how each objective was addressed and the main components that informed the resulting OCM model that is to be evaluated in chapter 4.

Table 3.15 Summary of Objectives, Methods and Components Developed

Research Objective	**How Addressed**	**Main Components Developed**
Develop a suitable language and domain independ. methodology	Design of conceptual framework	The OCM Conceptual framework found in section 3.1.
Design an architectural model and algorithms	Case study for concepts discovery process	Explicit and implicit concepts algorithms and heuristics
	Modeling Query Semantics Transfer Process (NLQ → DSF →SPaRQL)	QuSeT model
	Modeling of Feature Space schema	FSM
	Case study for deciphering meanings from Schema Data	Common nomenclature patterns
	Modelling 'Concepts Re-construction'	OWoRA
	Design of data structures for schema data	Gazetteer
	Design of Concepts Mapping Algorithm	SaCOMA
	Design of Structured Query-Generator function	Structured Query-Generator function
	Design of MAIN Algorithms & Heuristics	The OCM Algorithm
	Assembly of Components to form OCM-based Architectural Model	Architecture for Ontology-ased NL-Access to DBs (ONLAD)
Evaluation	Development of Prototype	Test bed; OCM-based protype

In summary results from these two studies gave rise to two important contributions namely a semantics transfer (QuSeT) model based on generative-transformation grammar and an ontology words reconstruction algorithm (OWoRA). The semantics transfer framework exploits kernelization procedures to express an NLQ into its constituent chunk phrases and presents these as triples of semantic bearing components. It was also shown how these triples are modeled into SPaRQL queries

that access ontologies build from relational databases. The chapter also highlighted the procedures of designing two schemata, that is an FSM and a gazetteer as well as a Semantically Augmented Concepts Matching Approach (SACoMa) which are crucial components of the OCM. Other procedures presented included implicit concepts discovery, foreign keys handling, triples assembly, structured query generation and the overall OCM algorithm. The last section of the chapter dedicated itself to prototype development activities.

The next chapter explains the experiments that were used in evaluating the OCM model described in this chapter as well as the results.

Chapter 4: EVALUATION AND FINDINGS

4.0 Preamble

There is no standard framework for evaluating the performance of NL access to DB models at present. Various researchers have used different evaluation parameters and procedures as described in detail in section 2.7 of the literature review. The dominant quantitative parameters observed from literature included precision, recall, accuracy and F-Score while qualitative measures have varied from one research to the other with no single dominant parameter. In fact some researchers do not include qualitative parameters for evaluation. Further to this, the procedure to be followed in measuring these quantities has not been standardized. This chapter therefore highlights the procedures and parameters selected or designed for evaluating the OCM model.

4.1 Evaluation Framework (Parameters and Procedures)

An evaluation framework describes the environment, procedures and parameters used in determining the performance of a model. To evaluate the OCM model, an evaluation framework was designed after analysis of literature. The general process flow for the evaluation process applied is illustrated in figure 4.1.

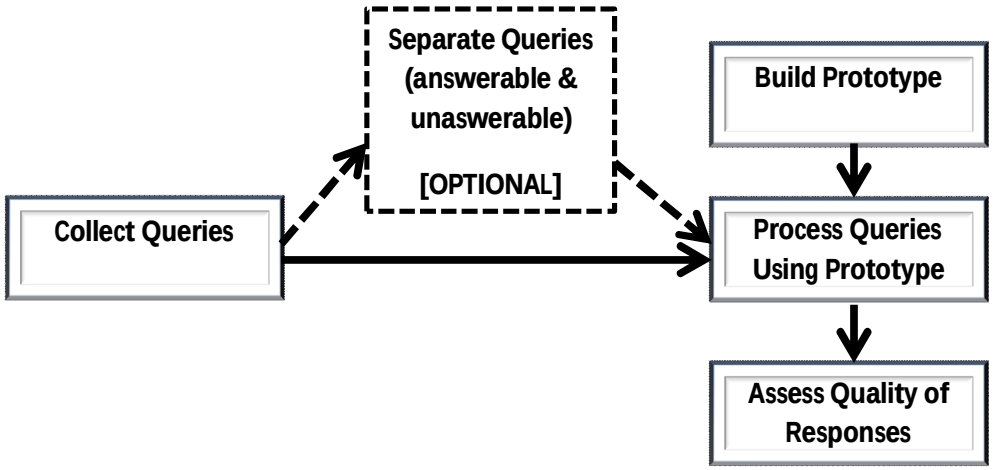


Fig. 4.1 General Evaluation Process Flow

From literature slight variations are observed in the five components of figure 4.1 depending on the preferences of the researcher. For example in query collection, NL questions may be collected from paper-based questionnaires like was the case in EXACT (Yates, Etzioni, & Weld, 2003), AquaLog

(Lopez, Pasin, & Motta, 2004) among others or generated electronically as in *Tiscover NL interface* (Dittenbach & Berger, 2003), *QuestIO* (Tablan, Damljanovic, & Bontchev, 2008). Others prefer to use existing queries such as Geo-queries and Restaurant queries by Tang and Mooney (2001). This research utilized five datasets for evaluation as explained in 3.3 where one set was collected manually, another one electronically and three others were obtained from existing query-sets, the aim being to provide a bench mark result.

The next task is to separate queries that may be answered by the system from those that cannot be answered due to lack of enough information or being out of context of the ontology. Popescu (2003) for example separated 'semantically tractable' questions from those that are not. Querix (Esther, Abraham, & Renato, 2006) separated what they termed as 'nonsense' questions from sensible ones. Yates et al. (2003) separated what they called 'non-goal oriented' questions from goal oriented. Some researchers however advocate for the use of raw question-sets as collected from users. Works such as *e-tourism NL interface* (Ruiz-Martınez, et al., 2009) and *Tiscover NL interface* (Dittenbach & Berger, 2003) used questions in their raw form to evaluate or perform analysis on questions. This research adopted the latter. This was motivated by the fact that users of these applications in the real world, submit all manner queries including the ones that yield no results. The model should provide the real performance under such circumstances. Since the queries within the query sets ware randomly collected, the true performance is therefore likely to be indicated.

Building of prototypes is an important step in the evaluation process. However tools used for implementing these prototypes vary, thereby giving non-standard testing environments. If similar tools and resources are used, the testing environment can be standardized. Section 4.2 provides an in-depth discussion on tools and resources selected for this work.

The last step in the evaluation framework involves assessing the quality of the output. Here variations occur with respect to what is being evaluated. For example PANTO (Wang, Xiong, Zhou, & Yu, 2007), *e-tourism NL interface* (Ruiz-Martınez, et al., 2009) among others manually inspected the quality of the structured query (SPaRQL) generated, while a large number of other researchers including QuestIO (Tablan, Damljanovic, & Bontchev, 2008), Querix (Esther, Abraham, & Renato, 2006), AquaLog (Lopez, Pasin, & Motta, 2004) among others preferred subjecting the generated structured query to the ontology. This research selected the latter because it gives a better reflection of the expected performance if the system was to function in an environment where a user is

querying an existing ontology. Moreover, there may exist errors unnoticeable by the eye if inspection of the structured queries is done manually. In assessing the quality of responses generated from the ontology, a given response can easily be classified as 'correct', 'wrong' and 'no-answer'. Figure 4.2 illustrates this classification of answers generated by a SPaRQL query from the ontology,

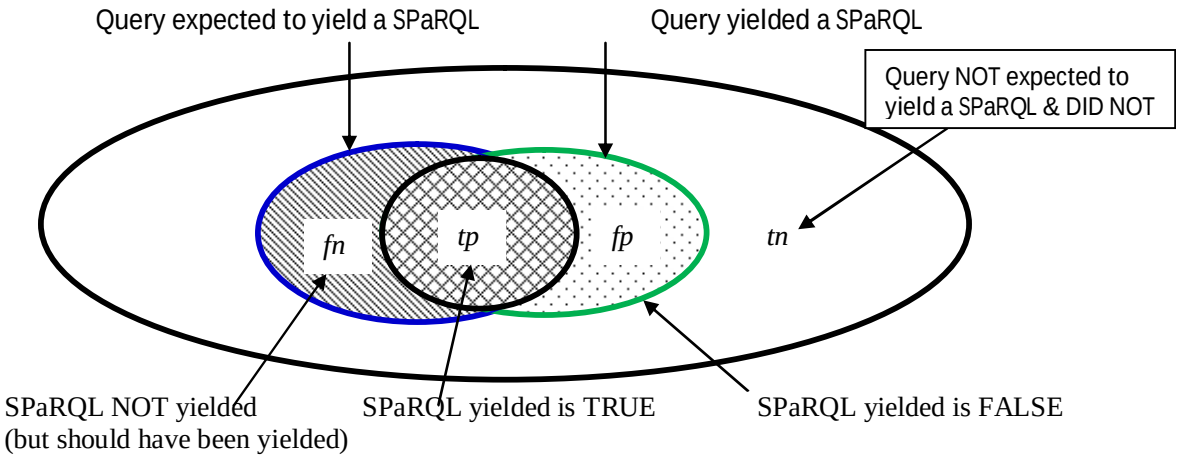


Fig. 4.2 Illustration of Categories Used in Evaluation

The recorded answers from each category formed the basis for calculating the accuracy, precision and recall values as explained in section 4.11.

The Evaluation framework used for this study had seven aspects as listed below,

a) Four quantitative measurements namely *Precision, Recall, Accuracy* and *F-score* and
b) Four qualitative measures namely *Domain independence, Language-independence, Support for Cross-linguality and Effect of Query Complexity on Model.*

4.1.1 Quantitative Parameters

From figure 4.2 it is observed that the model generates four categories of answers. True positives (tp) indicate cases where SPaRQL was generated and upon passing the query to the reasoner in the Protégé tool, a correct answer (as expected from the database point of view) was produced. False positives (fp) show cases where the SPaRQL generated an unexpected answer (not fulfilling the request expressed in NL). False negatives (fn) show cases where the system did not return any

SPaRQL although it was expected to, while true negatives (tn) indicate cases where the system did not produce a SPaRQL and was not expected to produce.

Precision is calculated as a ratio of the SPaRQL queries generated (and that yield right answers) to the total queries generated by the system. It therefore indicates the quality of the answers obtained from the system. The higher the precision, the better the performance is.

Recall is calculated as a ratio of the SPaRQL queries generated (and that yield right answers) to the total queries that should have been generated by the system. Recall indicates the extent to which our model generates true SPaRQL queries. The significance of recall is to show the range of questions the model is able to handle. Models with higher recall values are said to have better performance over those with lower recall value.

Another parameter used is accuracy. Accuracy is expressed as a percentage of sum of true positives and true negatives against the total number of queries passed to the system. Accuracy level shows the extent of the correct answers (tp and tn) users would obtain from a given query set. Accuracy therefore indicates the confidence one would have in a model given a specified query set.

Another parameter used is the harmonic mean of precision and recall, also known as F-score. It allows for expression of precision and recall as a single value. The higher the F-score, the better the performance. These parameters are summarized in table 4.1.

Table 4.1 Summary of Quantitative Parameters Used

Measurement Parameter	**Formula for Obtaining Parameter**	**Parameter Indicates**
Precision	*tp/ tp+ fp*	Quality of Answers Given
Recall	*tp/ tp+ fn*	Range of Questions Answered
Accuracy	*tp+tn/tp+fp+tn+fn*	User Confidence in the Model
F-score	*2(Precision X Recall)/(Precision + Recall)*	Model's Mean Performance

4.1.2 Qualitative Parameters

Domain Independence: This is the ability of a model to be ported from one area of application to another without the deterioration of the performance of the model. Domain independence is obtained from variance analysis where one studies if significant variations occur when the domain is successively changed. Accuracy, recall and precision are calculated for each domain area while holding the querying language constant.

Language Independence: This is the ability of a model to use two or more natural languages without the deterioration of the performance of the model. Language independence is obtained from variance analysis where one studies if significant variations occur when the language is successively changed. Accuracy, recall and precision are calculated for each language applied. Where multiple domains are involved the average value of each parameter is used while determining the variance value.

Support for Cross-Lingual Querying: This is the ability of a model to use a certain natural language to query a database given that the language used to author ontology or database schema objects is different from the querying language.

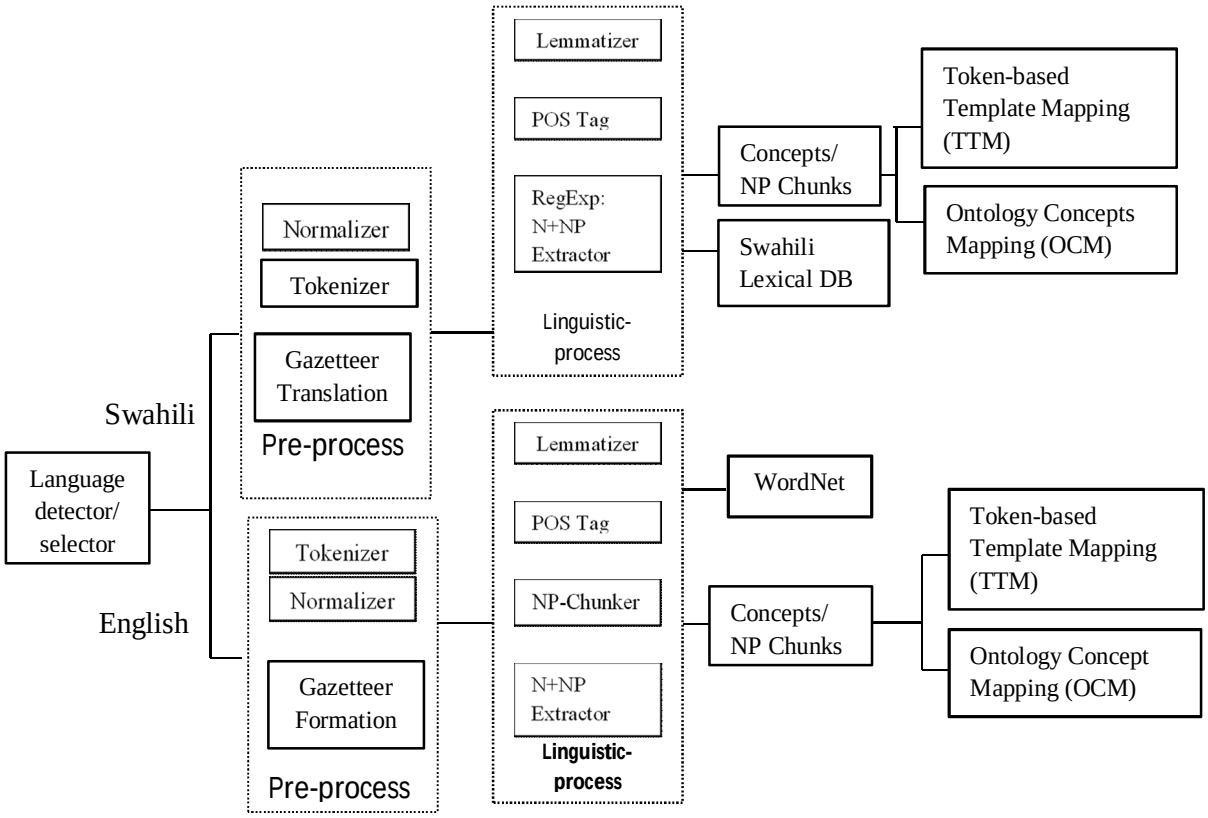
Effect of Query Complexity on the OCM model:

The degree of complexity of a query is assumed to be proportional to the number of concepts within a query. Most models deteriorate in performance with an increase in the number of concepts within a query.

The quantitative and qualitative parameters were experimentally obtained as described in the section 4.3.4.

4.2 Test-bed

The test bed that was developed and used for experimental purposes is shown in figure 4.3.



NB: TTM stands for Token-based Template Mapping (Muchemi L. , 2008) while OCM stands for Ontology Concepts Mapping.

Fig. 4.3 Test bed used in Prototypes Evaluation

The python code developed for this prototype is found in Appendix 8.

4.3 Evaluation Datasets

In the experiments carried out, two models, the OCM and the TTM (Muchemi L. , 2008) were applied across five databases. Table 4.2 outlines a summary of these databases. These databases were specifically used for the following reasons,

The farmers-db was mainly used to test Kiswahili aspects of the study while the UoN MSc coordinator's correspondences database was used to study English aspects of the research. The Northwind database which is shipped with Microsoft database server provides a well-known database which can provide some standard testing environment. It has also been used to test other database access models such as ELF (Bootra, 2004). The other two databases Restaurants-db and Job-search_db have been widely used by researchers in this area in evaluating their models. By using these databases to evaluate the OCM model, the results were easily bench-marked.

Table 4.2: Relational Databases Used in the Experiments

	Name of Database	No of Tables	Description
1	Chicken Farmers_db	8	Database created to mimic the one at Thika poultry farmers' project help desk as reported in Muchemi, (2008)
2	UoN MSc Coordinator_db	4	Database created to mimic students' records database at University of Nairobi.
3	Microsoft's Northwind_db	8	Standard database shipped with Microsoft's database server
4	Restaurants_db	7	Database whose schema is described in Tang & Mooney, (2001) and has been quoted widely in experiments[23]
5	Computer Jobs_db	4	Database whose schema is described in (Tang & Mooney, 2001) and has been quoted widely in experiments[24]

Each database was subjected to queries that had been collected through procedures described in section 3.3. The query sets were large and required sampling. Section 4.2.3 explains how sampling was done.

4.4 Queries Sampling Procedure

Each query set was treated as a separate population. Sampling was done from these query sets. A stratified random sampling approach was used to select queries from each population.

Each population (single query set) was divided into eight strata where each stratum contained queries of varying complexity. Complexity was defined in a similar manner as in Tablan *et al.* (2008) where the complexity of a question was assumed to increase with the number of concepts present in a query. This was determined by counting the number of meaning bearing components in

[23] See https://files.ifi.uzh.ch/ddis/oldweb/ddis/index.php%3Fid=519&print=1&no_cache=1.html for a reproduction
[24] See https://files.ifi.uzh.ch/ddis/oldweb/ddis/index.php%3Fid=519&print=1&no_cache=1.html for a reproduction

a query. A component was defined as a chunk of a phrase of any type, nouns, modifiers or collocation terms.

Table 4.3: Query Sets Used for Evaluation

Name of Query-set	Total No. of Questions in Set	No. of Queries Selected for Evaluation	Description	Original Source
Farmers Queries	625	200	Swahili queries based on poultry farmers case study.	Muchemi, (2008)
UoN MSc Coordinator Queries	310	200	English queries based on UoN MSc students' coordinator query set.	Coordinator e-mails
ELF Queries to MS NorthwindDB	120	120	English queries originally created by Bootra to evaluate ELF on Microsoft northwind-db at Virginia Commonwealth University	(Bootra, 2004)
Computer Jobs Queries	500	250	English database and queries for computer jobs used originally by Tang under Ray Mooney for PhD work at Texas State University	Recreated from Tang & Mooney, 2001
Restaurant Queries	250	200	Same as above but for restaurant selection	Tang & Mooney, 2001
Total	1805	970		

Diversity of queries within a single stratum was ensured through selection of queries of different types where the types were as defined in section 3.3.7. These included *'what'*, *'where'*, *'enumerative'*, *'yes/no'*, *'list/show/give/find/describe'*, *'who'*, *'when'*, *'how'*, *'which'*, *'comparative'*, *'superlative' and disjunctive (choice)* types. Figure 4.3 shows the number of queries selected per population.

4.5 Experimental Determination of Mean Performance of OCM Model

The test procedures involved subjecting the OCM and TTM models to a complete set of queries from a given query set. Each question was run against the models as shown in the test bed in figure 4.3. The models generated SPaRQL queries that were applied to the OWL ontology that had been formed from the respective relational database. Appendix 12 shows an example of an NLQ and its various transformations until a SPaRQL query is generated. It further shows results generated by the SPaRQL query upon application to the OWL ontology. Human evaluators examined the answers generated from the database and classified each one of them as 'true positive', 'false positive', 'true negative' or 'false negative'. Four human evaluators were used to perform the tests. The evaluators were recruited from undergraduate computer science students at the University of Nairobi. They were given basic training on handling input and output responses of the prototype.
This section presents the experimental procedures and an outline of the analysis carried out.

4.5.1 Results from Test-Sets

The OCM model was applied across five databases as described in Table 4.2 of section 4.3.2 and the respective query sets described in Table 3.1 of section 3.3.4.4 used. The specific questions in each query set are found in appendix 1. In order to make a direct comparison with other models, the parse tree template mapping model (TTM) described in Muchemi (2008) was plugged into the test bed and results from these query sets obtained and tabulated.

Evaluations were done with the value of Levenshtein gap, μ being 0 or 1, meaning perfect matching of strings within the gazetteer and the FSM or an allowance of one insertion, deletion or substitution of a single character respectively. The experiments were done in a comparative manner with the purpose of establishing the optimum OCM performance.

The following section highlights these results. They are organized according to the five test sets done where each test set represents results for OCM and TTM for a particular query set.

Tables 4.4 and 4.5 show a summary of results from first test set,

Table 4.4: Test Set 1- OCM - Kiswahili Queries (Poultry Farmers_db)

Swahili_Queries (Farmers_db)			
Experiment 1		**μ = 0**	**μ = 1**
	True Positives	118	116
	False Positives	13	32
	True Negatives	34	29
	False Negatives	35	23
	Total Queries	**200**	**200**
	Precision	0.90	0.78
	Recall	0.772	0.83
	Accuracy	0.76	0.73
	F-score	0.83	0.81

Table 4.5: Test Set 1-TTM- Kiswahili Queries (Poultry Farmers_db)

Swahili_Queries (Farmers_db)		
TTM		
	True Positives	87
	False Positives	42
	True Negatives	29
	False Negatives	42
	Total Queries	**200**
	Precision	0.67
	Recall	0.67
	Accuracy	0.58
	F-score	0.67

Tables 4.6 and 4.7 show a summary of results from the second test set,

Table 4.6: Test Set 2- OCM -English Queries (Microsoft's Northwind_db)

English_Queries (Microsoft_db)			
OCM		**μ = 0**	**μ = 1**
	True Positives	73	75
	False Positives	5	12
	True Negatives	5	7
	False Negatives	37	26
	Total Queries	**120**	**120**
	Precision	0.94	0.86
	Recall	0.66	0.74
	Accuracy	0.65	0.68
	F-score	0.78	0.79

Table 4.7: Test Set 2- TTM -English Queries (Microsoft's Northwind_db)

English_Queries (Microsoft_db)		
TTM		
	True Positives	61
	False Positives	28
	True Negatives	8
	False Negatives	23
	Total Queries	**120**
	Precision	0.68
	Recall	0.73
	Accuracy	0.57
	F-score	0.70

Tables 4.8 and 4.9 show a summary of results from the third test set,

Table 4.8: Test Set 3- OCM - English Queries (UoN MSc Coordinator_db)

English (UoN_MSc coordinator_db)			
OCM		**μ = 0**	**μ = 1**
	True Positives	110	103
	False Positives	14	16
	True Negatives	28	32
	False Negatives	48	49
	Total Queries	**200**	**200**
	Precision	0.88	0.87
	Recall	0.70	0.68
	Accuracy	0.69	0.68
	F-score	0.78	0.76

Table 4.9: Test Set 3- TTM - English Queries (UoN MSc Coordinator_db)

English (UoN_MSc coordinator_db))		
TTM		
	True Positives	90
	False Positives	29
	True Negatives	32
	False Negatives	49
	Total Queries	**200**
	Precision	0.76
	Recall	0.65
	Accuracy	0.61
	F-score	0.69

Table 4.10 and 4.11 shows a summary of results from the fourth test set,

Table 4.10: Test Set 4- OCM - English Queries (Restaurants_db)

English_Queries (Restaurants_db)			
OCM		μ = 0	μ = 1
	True Positives	98	99
	False Positives	11	20
	True Negatives	40	33
	False Negatives	51	48
	Total Queries	200	200
	Precision	0.90	0.83
	Recall	0.66	0.67
	Accuracy	0.69	0.66
	F-score	0.76	0.74

Table 4.11: Test Set 4- TTM - English Queries (Restaurants_db)

English (Restaurants_db)		
TTM		
	True Positives	85
	False Positives	34
	True Negatives	33
	False Negatives	48
	Total Queries	200
	Precision	0.71
	Recall	0.64
	Accuracy	0.59
	F-score	0.67

Tables 4.12 and 4.13 show a summary of results from the fifth test set,

Table 4.12: Test Set 5- OCM - English Queries (Computer_Jobs_db)

English_Queries (ComputerJobs_db)			
OCM		μ = 0	μ = 1
	True Positives	108	107
	False Positives	15	20
	True Negatives	27	28
	False Negatives	50	45
	Total Queries	**200**	**200**
	Precision	0.89	0.84
	Recall	0.68	0.70
	Accuracy	0.68	0.68
	F-score	0.77	0.77

Table 4.13: Test Set 5- TTM - English Queries (Computer_Jobs_db)

English (ComputerJobs_db)		
TTM		
	True Positives	88
	False Positives	39
	True Negatives	28
	False Negatives	45
	Total Queries	**200**
	Precision	0.69
	Recall	0.66
	Accuracy	0.58
	F-score	0.68

4.5.2 Discussion of Quantitative Evaluation Results

A summary of the results presented in section 4.5.1 is presented in Table 4.14.

Table 4.14 Summary of Results

	Levens gap)	**Exp. 1** Swa-Farmers	**Exp. 2** Northwind	**Exp. 3** MSc_Coord	**Exp. 4** Restaurant Search	**Exp. 5** Jobs	**Average**
Precision (%)	μ = 1	0.78	0.86	0.87	0.83	0.84	0.836
	μ = 0	0.90	0.94	0.89	0.90	0.88	0.902
	TTM	0.67	0.69	0.76	0.71	0.69	0.704
Recall (%)	μ = 1	0.84	0.74	0.68	0.67	0.70	0.726
	μ = 0	0.77	0.66	0.70	0.66	0.68	0.694
	TTM	0.67	0.73	0.65	0.64	0.66	0.67
Accuracy (%)	μ = 1	0.73	0.68	0.68	0.66	0.68	0.686
	μ = 0	0.76	0.65	0.69	0.69	0.68	0.694
	TTM	0.58	0.58	0.61	0.59	0.58	0.588
F-Score	μ = 1	0.81	0.79	0.76	0.74	0.77	0.774
	μ = 0	0.83	0.78	0.78	0.76	0.77	0.784
	TTM	0.67	0.71	0.70	0.67	0.68	0.686

The results indicate a model whose average precision at a Levenshtein distance μ, of 1 (within the matching function) is 0.84 and increases to 0.90 on decrease of μ to 0. Precision therefore increases with decrease of μ while recall decreases from 0.73 to 0.69 when the edit distance is changed from one to zero.

Since precision indicates the quality of the answers obtained from the system, it is true that the higher the precision, the better the quality of the answers received by the user. Thus based on

precision alone **μ** should be restricted to zero. As stated in section 4.1.1, recall is an indicator of the extent to which OCM generates SPaRQL queries given a wide range of question types and varying complexity. The significance of recall is to show the range of questions the model is able to handle. The higher the recall value the better performance, thus based on recall, μ should be set to one. Accuracy on the other hand increases slightly from 0.686 to 0.694, a difference that is not significant enough for a clear performance enhancement to be concluded. Increasing **μ** from 0 to 1 (relaxing the matching constraint) means that instances where no SPaRQL is generated are decreased thereby increasing recall. However the generated SPaRQL gives many undesired results (many false positives). On the other hand, only a slight increase in true positives ('tp') is noted. By definition precision was given in table 4.1 as, *tp/(tp+fp)*, meaning the precision decreases with increase in μ. Accuracy was also defined in table 4.1 as *tp+tn/(tp+fp+tn+fn)*. As observed, increasing μ from 0 to 1 decreases instances where no SPaRQL is generated but increases 'fp' with only a slight change in 'tp'. This in effect means that only a slight increase in accuracy is expected, as was the case with the observations made from the experiments.

A suitable parameter for gauging the overall suitability is the F-score, the harmonic mean of precision and recall, which records a slight increase from 0.774 to 0.784 on tightening μ from one to zero. This indicates that any NLQ system that relies on string matching for extracting explicit and implicit concepts from a database should have an edit distance of zero in the matching function if precision and F-score are the main considerations.

4.6 Experimental Determination of Domain Independence

4.6.1 Experiments Setup

An experiment was set up to determine the degree of language independence. Four different domains were selected for testing the degree of independence. The domains were based on the fields of trading, student-management, job-search and finding-restaurants. The setup is as shown in figure 4.4.

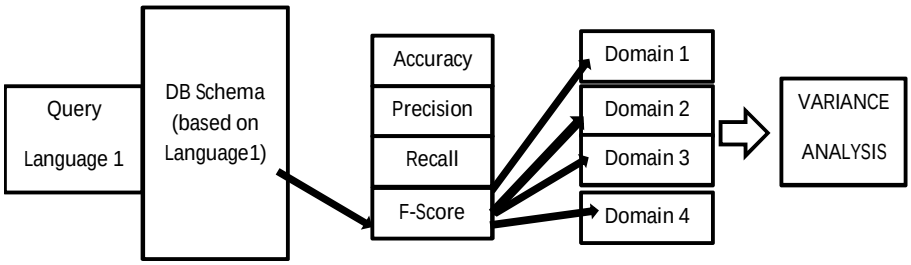


Fig. 4.4 Experimental Procedure for Domain Variance Experiments

4.6.2 Analysis Overview

By holding the querying language constant, accuracy, recall and precision were calculated for each domain and tabulated. The difference between the mean of each respective parameter (F-score, accuracy, precision or recall) and computed performance values was determined and expressed as a percentage. The results are shown in table 4.15.

Further the standard deviation, σ for each parameter (F-score, Accuracy, Precision or Recall) was calculated and tabulated as shown in table 4.16. A variance analysis was performed on the four obtained values by way of determining standard deviation for the four domains. For every given domain the departure from the standard deviation was computed and noted if it was within the standard deviation and if not by what ratio.

The standard deviation, σ was computed as follows,

- The mean of a parameter say F-Score is worked out by calculating the arithmetic mean across the four domains.
- For each domain, the mean parameter value is subtracted from the respective parameter value for that domain and the difference squared.
- The average of those squared differences is then worked out. The average of the squared differences from the mean is the variance.
- The standard deviation is obtained by obtaining the square root of the variance.

- To test whether the parameter value is outside the standard deviation, the standard deviation is subtracted from that value and if the difference is zero or less, then it is interpreted that the value is within the standard deviation.

The standard deviation is an important indicative parameter because it shows what values are within 'normal range' and which ones are not. This procedure was done for accuracy, precision, recall and F-score.

4.6.3 Results for Domain Independence Experiments

Since standard deviation is a measure of dispersion or volatility of data from its mean, then a parameter with high volatility indicates that it is affected by changes in data treatment, meaning domain change. The mean of the performance values for each domain and their aggregate mean were computed. The variance and standard deviation values were also determined and tabulated in table 4.15.

Table 4.15 Evaluating Domain Independence of the OCM Method (Std Deviation Analysis)

	Levensht. Dist. (μ)	DOMAIN (Mean Value)				Mean	Variance (σ^2)	Standard deviation (σ)
		Trade	Stud	Jobs	Rest			
F-score	1	0.71	0.71	0.76	0.74	0.730	0.00045	0.02121
	0	0.77	0.7	0.69	0.73	0.722	0.00097	0.03112
Accuracy	1	0.5	0.51	0.58	0.54	0.532	0.00097	0.03112
	0	0.61	0.51	0.5	0.54	0.540	0.00185	0.04301
Recall	1	0.68	0.66	0.74	0.76	0.710	0.00170	0.04123
	0	0.65	0.59	0.57	0.7	0.627	0.00262	0.05117
Precision	1	0.73	0.78	0.78	0.71	0.750	0.00095	0.03082
	0	0.93	0.87	0.88	0.77	0.862	0.00337	0.05804

Table 4.16 shows a deviation analysis that was done for each domain by finding out whether the performance value was outside the standard deviation or not.

Table 4.16 Evaluating Domain Independence of the OCM Method (Outlier Points Analysis)

				Deviation of Parameter (eg Accuracy) From Mean					No. of Times Means Deviates from std. deviation [(x-mean)/σ]			
	μ	Mean	σ	TRADE	STUD	JOBS	REST		TRADE	STUD	JOBS	REST
F-Score	1	0.730	**0.021**	0.020	0.020	-0.030	-0.010		0.940	0.942	1.414	0.471
	0	0.722	**0.031**	-0.048	0.022	0.032	-0.007		1.520	0.722	1.044	0.240
Accuracy	1	0.532	**0.031**	0.033	0.022	-0.047	-0.007		1.040	0.722	1.526	0.240
	0	0.540	**0.050**	-0.070	0.030	0.040	0.000		1.627	0.697	0.929	0
Recall	1	0.710	**0.041**	0.030	0.050	-0.030	-0.005		0.727	1.212	0.727	1.212
	0	0.627	**0.051**	-0.022	0.037	0.057	-0.072		0.439	0.732	1.123	1.416
Precision	1	0.750	**0.031**	0.020	0.030	-0.030	0.040		0.648	0.973	0.973	1.297
	0	0.862	**0.060**	-0.067	0.007	0.017	0.092		1.162	0.129	0.301	1.593

When expressed as standard deviation, most values were found to be within the normal standard deviation. In a few cases (those having a value greater than one in the last four columns of table 4.16) the deviation was found to be above the standard deviation. The Peirce Criterion (Ross, 2003) for determining outlier data was used to classify if the data was an outlier or not.

In order to determine whether any of the values was an outlier, the following steps illustrated through an example were followed,

- The standard deviation and the mean of the complete set (across the four domains) were determined (earlier determined and recorded in table 4.16).
- The parameter, R was obtained from the Peirce's table (Ross, 2003) for a four data point-one outlier condition,
    - From Peirce's Table, R was found to be 1.383.
- The product of σ and R is therefore 1.383 x σ, which we refer to as S.
- Assuming that the F-score for the Restaurant query set is under investigation (whether outlier or not), σ is read from table 4.16 above as 0.021, making the value of S to be 0.02904
- The maximum allowable deviation, R was calculated as follows,
    - $R_{max} = (|x_i - x_m|_{max})/\sigma$

    Where,

    - R is the ratio of the maximum allowable deviation of the measured value from the mean of the data to the standard deviation,

- x_i is the measured value (suspect data),
- x_m is the mean of the dataset and,
- σ is the population's standard deviation.

- For the suspect data, F-score of jobs query set with a μ of 1, the quantity R was calculated as follows,
  - $R_{max} = (|x_i - x_m|_{max})/\sigma|$
  - x_i, x_m and σ are read from table 4.15 as 0.74, 0.73 and 0.02121 respectively, hence R_{max} is 0.471475
  - If $S > R_{max}$, then the data is classified as an outlier, else if $S < R_{max}$, then it is classified as normal data
- S has a value of 0.02904 while R_{max} has a value of 0.471475, hence $S < R_{max}$ therefore the suspect data, F-score of jobs query set with a μ of 1 was classified as normal data.

The above procedure was repeated for all data in table 4.16 and that was outside the respective normal deviation values and no data was found to be an outlier. This led to the conclusion that the model is not sensitive to domain change and is therefore to some extent domain independent. The case study involved four different domains (a number limited by practical reasons for the scope of this work), which may not be adequate to come to a conclusion that domain independence under all possible conditions has been achieved. However, this work contributes in a significant way by pointing to an approach that can be more rigously tested.

4.7 Experimental Determination of Language Independence

4.7.1 Experiments Setup

In order to determine the degree of language independence, the performance values say the F-score, were determined in the set up illustrated in figure 4.5. The language of querying was successively changed in different set ups and the performance values calculated and tabulated. The mean, variance and standard deviation of each parameter was determined for query sets written in Kiswahili and English.

For example, if Language 1 is English, the database schema was also based on English language abbreviations and concatenations. The performance values were determined and tabulated.

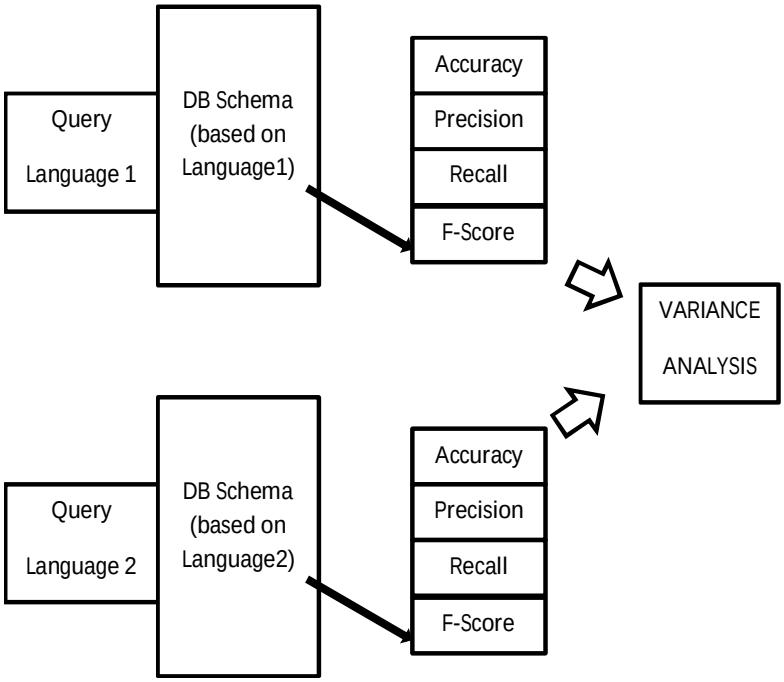


Fig. 4.5 Experimental Procedure for Language Independence Experiments

4.7.2 Analysis Overview

The aim was to evaluate the degree of language independence as opposed to evaluating handling of many languages concurrently. A similar variance analysis as described in section 4.6.2 was performed. The Peirce Criterion (Ross, 2003) for determining outlier data as described in section 4.73 was again used to classify if data was an outlier or not.

4.7.3 Results of Language Independence Experiments

Table 4.17 shows the mean, variance and standard deviation of each parameter that was determined from the experiments described in 4.7.1.

Table 4.17 Evaluating Language Independence of OCM (Performance Variance Analysis)

	Levensht. Dist. (μ)	Language (Mean performance)		Mean	Variance (σ^2)	Standard deviation (σ)
		Swahili	English			
F-score	1	0.700	0.730	0.715	0.000225	0.015
	0	0.690	0.722	0.706	0.000256	0.016
Accuracy	1	0.500	0.532	0.516	0.000256	0.016
	0	0.490	0.540	0.515	0.000625	0.025
Recall	1	0.670	0.710	0.690	0.000400	0.02
	0	0.580	0.627	0.604	0.000552	0.0235
Precision	1	0.740	0.750	0.745	0.000025	0.005
	0	0.850	0.862	0.856	0.000036	0.006

This model was tested using Kiswahili and English queries. The average F-score on Kiswahili queries was found to be 0.700 against 0.730 for English queries. The mean precision values for Kiswahili were 0.740 and 0.850 against 0.750 and 0.862 (with Levenshtein distance of 1 and 0 respectively) for English queries. The variance for F-score ranged between 0.015 and 0.016 (1.5% and 1.6% in percentage form). The variance observed for the other performance values ranged between 0.05% (precision with μ of 1) and 2.5% (accuracy with μ of 0).

The variances were subjected to Peirce Criterion and the change of language was found not to have an influence on the performance. This in turn means that the methods being used to convert the NLQ to SPaRQL are language independent. The case study involved two different languages, a number limited by practical reasons for the scope of this work. This set may not be adequate to come to a conclusion that language independence under all possible conditions has been achieved. However, this work contributes in a significant way by pointing to an approach that can be more rigously tested

4.8 Experimental Determination of Cross-Lingual Querying Ability

4.8.1 Experiments Setup

As set out in the problem statement, the issue of cross-lingual querying was central in this research. To determine the extent to which the model supports cross-lingual querying, an experimental set up shown in figure 4.6 was used.

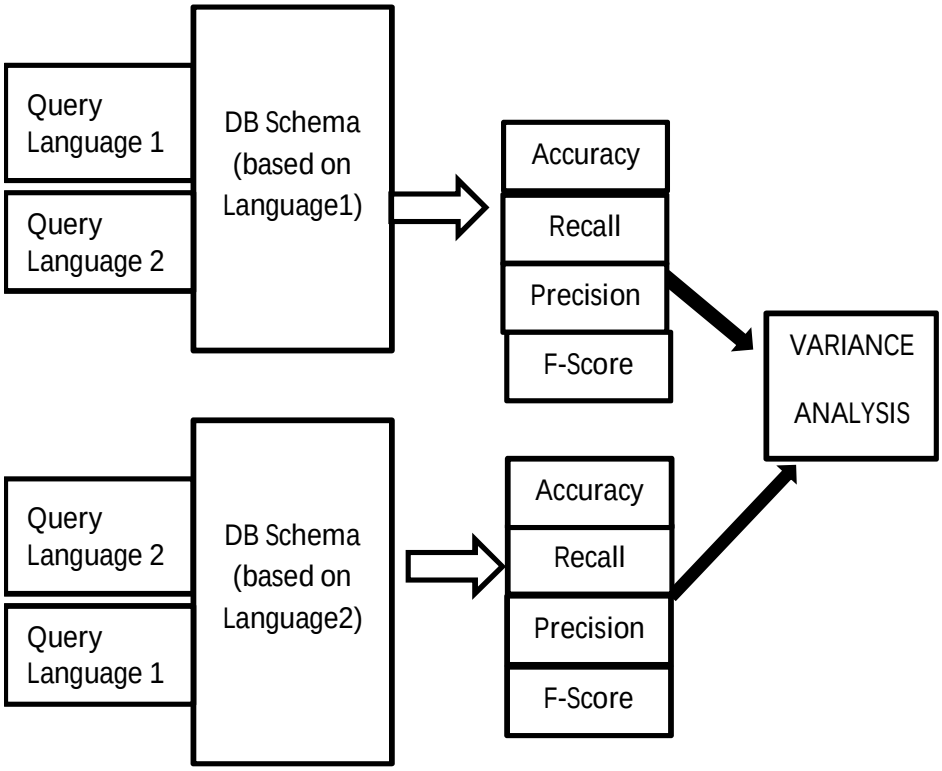


Fig. 4.6 Experimental Determination of Cross-Lingual Support

A total of four experiments were done. In the first experiment Kiswahili language was selected for querying. The underlying database schema authorship language was made similar to the querying language that is, Kiswahili. The performance parameters were determined and recorded. These included the F-score, accuracy, recall and precision.

In the second experiment the querying language was changed to English but the underlying database schema authorship language remained Kiswahili. The values for the performance parameters were

determined and recorded. Observations to check whether there were significant differences between the calculated values, that is while cross-lingual querying is present and when absent were made. In the third experiment the database was changed from farmers-db to UoN MSc coordinator's database. The underlying database schema authorship language for this database was English. Querying was done with English queries meaning there was no cross-lingual querying. The querying language was changed to Kiswahili in the fourth experiment meaning there was cross-lingual querying. The performance parameter values were calculated and observations made to check whether there were significant differences between the two calculated values (i.e. with cross-lingual and without cross lingual arrangement). A variance analysis was then carried out. The results obtained are tabulated in table 4.18.

4.8.2 Analysis Overview

A variance analysis similar to that described in 4.6.2 was applied to the performance values obtained from section 4.8.1 above. The Peirce Criterion (Ross, 2003) was used to determine if querying in an Interlingua manner had a significant effect or not. A significant deviation would indicate poor support for cross-lingual querying.

4.8.3 Results from Cross-lingual Support Experiments

Table 4.18 summarizes the results obtained from the cross-lingual experimentation study. The mean, variance and standard deviation were determined for each experiment and the results recorded in the table.

Table 4.18 Cross-Lingual Mean Variances and Standard Deviation Analysis

	Levensht Dist. (μ)	Mean performance)				Mean	Variance (σ^2)	Standard deviation (σ)
		Swahili (Farmers_db)		English (Coordinator_db)				
		Swahili Queries	English Queries	English Queries	Swahili Queries			
F-score	1	0.703	0.6900	0.730	0.715	0.7095	0.000218	0.014773
	0	0.689	0.675	0.722	0.721	0.7018	0.000415	0.0203639
Accuracy	1	0.500	0.485	0.532	0.545	0.5155	0.000578	0.0240468
	0	0.490	0.465	0.540	0.550	0.5113	0.001223	0.035067
Recall	1	0.670	0.680	0.710	0.700	0.6900	0.000250	0.015811
	0	0.580	0.595	0.627	0.620	0.6055	0.000358	0.018928
Precision	1	0.740	0.700	0.750	0.730	0.7300	0.000350	0.018708
	0	0.850	0.780	0.862	0.860	0.8380	0.001142	0.033794

In order to determine if the OCM model was significantly affected by cross lingual usage, the Peirce Criterion described in 4.6.3 was applied on data that was beyond the standard deviation. The affected data is indicated on table 4.19 as that with a value of greater that one (or negative one) in the last four columns of table 4.19, for example F-score of experiment two with a Levenshtein distance of one.

Table 4.19 Cross-Lingual Outlier Performance Analysis

				Difference of Performance & Mean					No. of Times Performance Deviates from std. deviation [(x-mean)/σ]			
				Swahili (Farmers_db)		English (Coordinator_db)						
	μ	Mean	σ	Swahili Queries	English Queries	English Queries	Swahili Queries		Expt 1	Exp 2	Exp 3	Exp 4
F-Score	1	0.710	0.015	0.006	0.020	-0.021	-0.006		0.440	1.320	-1.388	-0.372
	0	0.702	0.020	0.013	0.027	-0.020	-0.019		0.626	1.314	-0.994	-0.945
Accuracy	1	0.516	0.024	0.016	0.031	-0.017	-0.030		0.645	1.268	-0.686	-1.227
	0	0.511	0.035	0.021	0.046	-0.029	-0.039		0.606	1.319	-0.820	-1.105
Recall	1	0.690	0.016	0.020	0.010	-0.020	-0.010		1.265	0.632	-1.265	-0.632
	0	0.606	0.019	0.026	0.011	-0.022	-0.015		1.347	0.555	-1.136	-0.766
Precision	1	0.730	0.019	-0.010	0.030	-0.020	0.000		-0.535	1.604	-1.069	0.000
	0	0.838	0.034	-0.012	0.058	-0.024	-0.022		-0.355	1.716	-0.710	-0.651

A significant deviation as indicated by the Peirce criteria would indicate poor support for cross-lingual querying. Table

None of the performances showed outlier behaviour implying that the model was not affected by the cross-lingual querying, therefore it was concluded that the model is independent of cross lingual querying.

Figure 4.7 shows a graphical analysis of the performance values. From this graph it is observed that cross-lingual querying has significant effect on the performance.

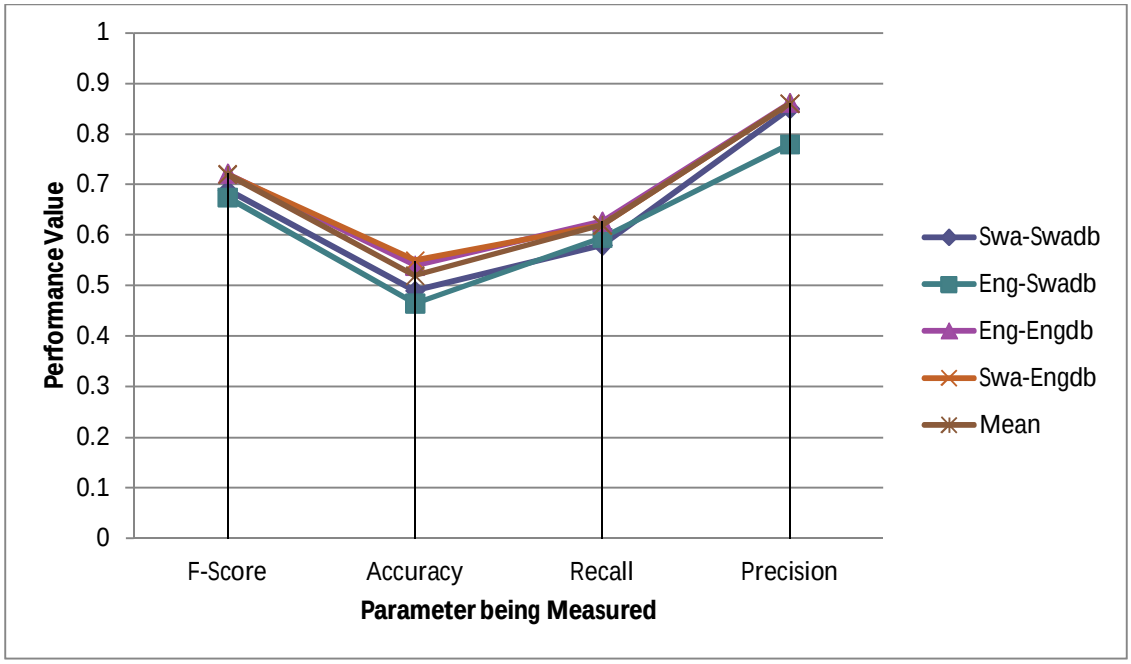


Fig. 4.7 Graphical Representation of Variances (μ = 0)

4.9 Effect of Concepts Complexity

Performance of most models depreciates with an increase in the complexity of the concepts (Dittenbach & Berger, 2003). The issue of complexity was handled in a similar manner as handled by Tablan et al. (2008) where the complexity of a query was assumed to increase with the number of concepts present in a query.

As noted in section 4.4, a stratified random sampling approach was used to select queries from each query set where each population was divided into eight strata. Each stratum contained queries of varying complexity. Complexity of a question was assumed to increase with the number of concepts present in a query. The effect of complexity of the query to F-score was studied by determining the performance value per stratum. Figure 4.8 presents a graph showing the mean F-score value across the datasets.

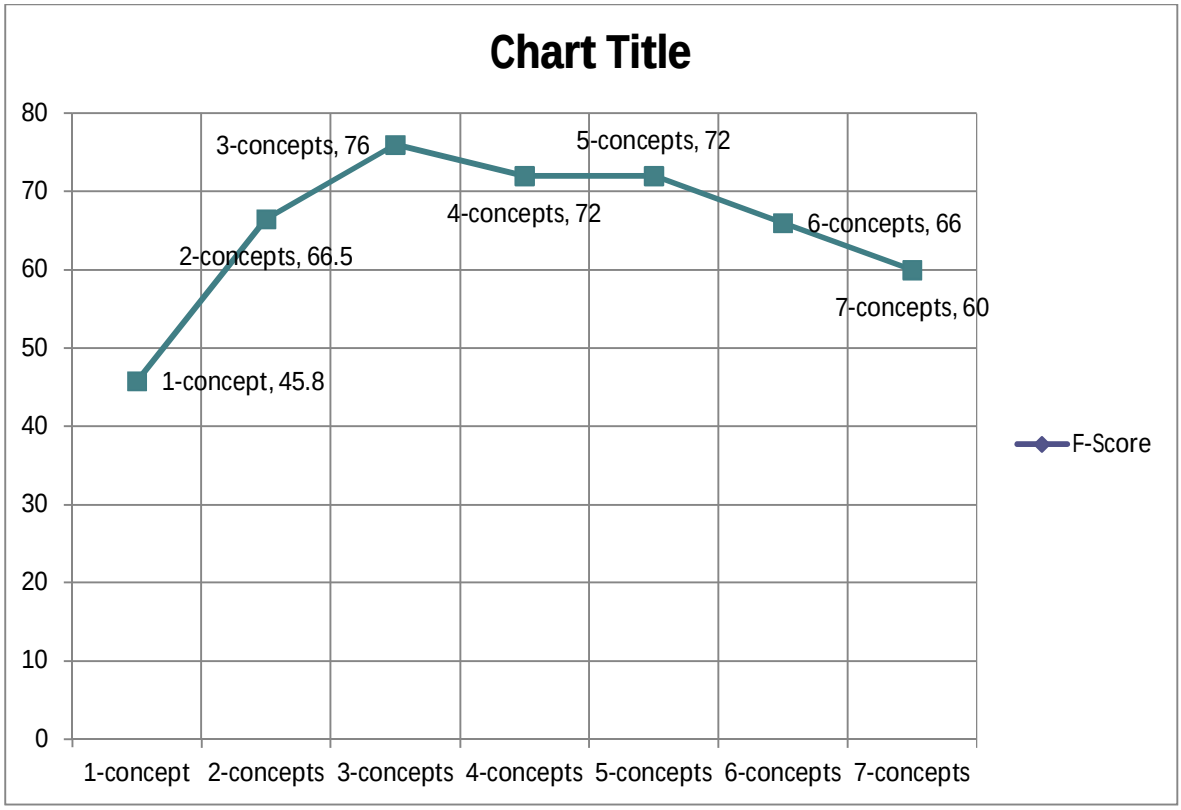


Fig 4.8 Relative Performance (F-score) versus Complexity of Query

4.10 Comparative Analysis with other Models

In order to make a direct comparison with other models both experimental and literature analysis methods were used.

In literature analysis, comparative study with several models was done. Benchmarking was achieved through comparison of performance with various published works as shown in table 4.20. Specifically where the test sets (queries and databases) and results were available a direct results comparison was done. As discussed in section 2.5 of the literature review and summarized in figure 2.11, the models were grouped into various categories. These included,

- Machine learning approaches such as semantic parsing where statistical methods were used, for example as in WASP by Ge and Mooney (2005) or grammar-based machine learning methods (e.g. machine learning with synchronous context free grammar with λ-expressions (λ-SCFG) by Minock, Olofsson and Naslund (2008))

- Logic based approaches where token or phrase-based methods were used. Examples included token-based graph-matching approach by Popescu, Etzioni and Kautz (2003) also known as PRECISE for English queries, token-based template matching approach for Kiswahili also known as TTM by Muchemi (2008) and *tiscover's* English NL interface (Dittenbach & Berger, 2003) which is a phrase-based approach among others.
- Ontology-based approach for related tasks such as access to application specific ontologies such as the GATE ontology (Tablan, Damljanovic, & Bontchev, 2008) or Health-e-Child database (Munir, Odeh, & McClatchey, 2008)

For the experimental investigation, the test bed illustrated in figure 4.3 was used. In particular the token based template mapping model (TTM) was run with the same test sets as OCM (queries and databases) and the results tabulated and compared. Results from other published works where the query sets and performance results were available were also used to conduct comparative analysis. Specifically OCM was evaluated on the same query set and database (Microsoft's Northwind-db) as was used to evaluate commercially available software namely the English Wizard (EasyAsk), English Query (Microsoft) and ELF (Elf Software Co) by Bootra (2004) and therefore its performance was directly comparable to these systems. PRECISE (Popescu, Etzioni & Kautz, 2003) was tested on three databases and query sets namely Restaurants-search, Computer-jobs-search and US Geography all from Tang and Mooney (2001). Out of these three, two of the sets namely Restaurants-search and Computer-jobs-search were used to evaluate OCM. The published results for PRECISE were therefore directly comparable to OCM's experimentally obtained results.

4.10.1 Summary of Performance Comparisons

Minock et al. (2008) has provided an elaborate review of performance of the most competitive models in logic-based mapping and semantic parsing approaches. A brief summary of the performance of the reviewed models together with the experimentally obtained results are provided in Table 4.20 below. Most of the models discussed in this section were developed and evaluated under different environments, therefore a direct comparison of results is not foolproof in itself. However the comparison has been provided to indicate a general performance trend as opposed to an absolute value.

Table 4.20 Comparison of Performance Values

Method	Model	Precision	Recall	Accuracy	F-Score	Main Principle
Machine Learning	WASP	0.800 – 0.915	0.600-0.940	0.500 – 0.866	0.690-0.930	Semantic Parsing using SCFG (Statistical)*(Ge & Mooney, 2005)*
	Minock et al. Model	0.600-0.850	0.500-0.800	0.800	0.550-0.820	SCFG with Lambda *(Miock, Olofsso & Naslund, 2008)*
Logic Mapping	PRECISE	0.800-0.100	0.550-0.775	0.450-0.775	0.650-0.870	Graph Matching *(Popescu, Etzion & Kautz, 2003)*
	TTM	0.704	0.670	0.588	0.686	Semantically Tagged phrase trees *(Muchemi, 2008)*
Ontology based Matching	CBM	0.575	Not provided	Not provided	Not provided	Constraint-based Method. From NL to OWL *(Gao, Liu, Zhong & Chen, 2007)*
	QUERIX	0.777	0.786	Not provided	0.781	NLI to Ontologies *(Kauffman, Bernstein & Zustein, 2006)*
	QuestIO	0.667	0.680	0.545	0.735	NLI to GATE *(Tablan, Damljanovic, Botcheva, 2008)*
	OCM	0.836-0.902	0.726-0.694	0.686-0.694	0.774-0.784	Ontology Concept Mapping *(This Thesis)*

The models highlighted in table 4.20 are representatives of the three main categories identified from literature. The models are among the highly performing and widely quoted within each category. The mean performance of the OCM model is shown in the last row of the table. A high precision is important because it indicates the quality of the parsed queries while recall indicates the extent to which a model generates the correct SPaRQL queries. It is also true that all models decline to answer some questions hence the need for recall. Accuracy on the other hand indicates the extent to which a user expects the correct answer from a given set of questions.

4.10.2 Comparisons with Logic-Mapping based Methods

It was noted that OCM had a precision ranging between of 0.836 and 0.902 which is close to the other models selected for comparison purposes. PRECISE seems to hit a high of 100% apparently because the question sets (also used in this work) seem to have been cleaned first, by eliminating

non-semantically tractable queries, thereby giving a 100% performance. The similarity in evaluating PRECISE and OCM converges at the selection of common databases namely Restaurants-search and Computer-jobs-search databases. These databases are published along with the query sets and therefore the queries used were from the same pool. However differences may arise from sampling techniques. The sampling method used in PRECISE was not indicated and therefore it was difficult to compare the actual questions used for evaluating. On average OCM performs marginally better on the lower limit of all performance indicators compared to PRECISE which means that OCM has a better guaranteed minimum performance. This may be attributed to the fact that OCM uses both tokens and phrase chunks as opposed to PRECISE which uses tokens only. The performance of TTM is also lower than OCM's on average and is close to PRECISE's because they use the same linguistic processing and the difference comes in the mapping of tokens to SQL fragments. The results obtained from tests carried out by Bootra (2004) on commercial systems showed that English Wizard (EasyAsk) had a precision of 0.484 and a recall of 0.31 while English Query (Microsoft) had a precision of 0.461 and a recall of 0.39. The query sets and results published by Bootra (2004) are presented in appendix 9 for ease of reference.

4.10.3 Comparisons with Machine Learning Methods

Research in machine learning approach has been active as noted in literature. The two most prominent methods are statistical parsing based and grammar-based machine learning approaches. Minock et al. (2008) has presented a synchronous CFG-grammar approach which produces meaning representations which are then mapped by a secondary process to the database elements as reviewed in the literature. The lower and upper bound values reported in Minock's et al. (2008) model seem slightly lower than those of OCM. The Minock et al. (2008) utilizing λ-SCFG had an F-score of between 0.550 and 0.820. The big range between the high and lower scores appears to be caused by distribution drift caused by requirement for prior training because it relies on semantic parsing which is based on machine learning. Another machine learning model is WASP which seemingly has higher performance values. Though WASP has seemingly higher values it is important to note that this technique requires to be interfaced with an SQL classifier thereby lowering these values by a factor equivalent to the precision of the SQL classifier.

4.10.4 Comparisons with Ontology based Methods

Table 4.20 shows some published results for models that rely on NLP to access general ontologies. OCM's functioning differs from these in that it's a general purpose model that accesses ontologies built on relational databases, while the others access ontologies which in most cases are built for specific purposes (e.g. the GATE ontology). The major difference in the ontology-based approaches indicated in table 4.20 is mainly in the manner in which each method handles the natural language query before producing the structured query. While OCM converts NLQ into kernel form and builds triples from the kernel and other linguistic components such as modifiers, the CBM converts an NLQ into an optimization mathematical formula, while Querix and QuestIO convert the NLQ into a bag-of-words and makes direct mappings to the ontology. A direct comparison of the results to these published performances may not be very suitable because of the different testing environments. For example Querix was tested on 250 queries of the US Geography (Tang & Mooney, 2001) while QuestIO was tested on 22 questions only from the GATE ontology project. CBM on the other hand was tested on 35 simulated and 40 real questions. While the number may not affect the results significantly, sampling methods need to be stated where data is provided. However the results published are indicative figures and may be used for comparative purposes in trying to establish explanations for various performances. CBM has the least precision at 0.575 followed by QuestIO at 0.667. Querix has a performance in the same range as OCM at 0.77. Although Querix and QuestIO use bag-of-words, Querix has a user feedback component which assists in guiding the questions posed by the user. This intervention may possibly explain the better performance of Querix compared to QuestIO. Apart from this performance enhancement feature, the main point of departure with the OCM is in the linguistic processing where OCM employs the kernelization process explained in 3.3.9.2 while Querix uses bag-of-words. This therefore indicates that the use of concepts, arising from a kernelization process, as opposed to bag-of-words leads to a better performance as is the case with OCM.

4.10.5 Summary of Comparisons with other Methods

A suitable parameter for gauging the overall suitability is the F-score, the harmonic mean of precision and recall. The model developed has been evaluated on five databases three of which are publicly available and hence results were easily comparable with the other state-of-the-art models.

OCM has marginally better results on recall for the lower bound value compared to all the other models. This can be explained by the fact that OCM exploits the idea of 'concept' as opposed to for example logic or machine learning models which utilize 'tokens' and 'phrase trees' respectively. A concept is comprised of tokens (mainly nouns), noun phrases, and multi-word terms including collocations. It means OCM is better able to comb through a query looking for more items to match against the database elements as opposed to the other approaches.

The model however suffers from low maximum recall and accuracy levels compared to other approaches as evident from results because it requires someone to enter information that sometimes is regarded as obvious or superfluous. For example the query *'give me customers who come from Nairobi'* might require one to add the word *'name'* within the query so that the system realizes we require 'customers' names'. While it is not surprising that all models decline to answer some questions, a good model should answer as many queries as possible. Recall indicates the extent to which the model generates SPaRQL queries. The better lower bound value compared to the others indicates more SPaRQL queries will be generated. On average the results show between 60 and 70% rate of conversion which is not a significant departure from the state of the art.

Accuracy shows the extent to which a user expects the correct answer from a given set of questions. From the results obtained one notices that OCM has a better minimum bound value at 0.686 compared to all the others except the model by Minock et al. (2008). The lower bound value for this model was not published and therefore cannot be assessed. Users can expect more accurate answers with OCM as compared to other approaches. Compared to PRECISE which has an upper bound value of 0.775, OCM has a lower value. However looking at the way experiments were conducted, PRECISE had to eliminate some questions which were found to be semantically not tractable and this obviously means better results. OCM on the other hand used the raw query set as collected and therefore has no bias.

4.11 Summary

This chapter was divided into ten subsections that highlighted various aspects of the evaluation methodologies, results, their analysis and a comprehensive discussion on performance evaluation.

An elaborate evaluation framework has been presented. The 8-point framework was one of the contributions made in this chapter. The chapter has presented the data sets and the query sampling techniques that were used.

The chapter has provided the procedures of obtaining the four quantitative parameters (precision, accuracy, recall and F-score) and the four qualitative parameters (domain independence, language independence, support for cross-lingual querying and effect of query complexity on the performance) that are central to NL access to relational database evaluation method. The chapter showed how through variance analysis the degree of independence was evaluated. These evaluation procedures were an achievement that may be replicated for this kind of analysis. Finally a comprehensive discussion on the comparative analysis study was made where it emerged that OCM has opened new frontiers in terms of new guaranteed minimum of performance and high F-score values.

Chapter 5: CONCLUSION

5.0 Preamble

This chapter provides an overview of the research carried out. In particular it revisits the focus of the problem, main objectives, approaches followed and the main results. The chapter focuses on the contributions, achievements and proposed recommendations for furtherance of this work.

5.1 Overview of Research

The unresolved issue of natural language processing for relational database access was the main problem addressed in this research. The challenge of developing a generalizable methodology that maps any given natural language to a suitable structured query language is the main issue that was tackled. The problem had three sub-components which are listed here below;

- lack of a generic language and domain independent methodology for understanding un-restrained natural language text and converting it to structured query language (SPaRQL) in the context of Kiswahili-English cross-lingual database,
- lack of language and domain independent parsers that convert free text into concepts. It is taken that relational database metadata which is organized as tuples is readily mapped to these concepts,
- Lack of domain independent ontology-parsers that convert meta-data from databases into suitable concepts that are mapped to natural language queries.

The main objectives were based on these three areas namely design and development of a generic language and domain independent approach, development of NL parsers and development of domain independent ontology-metadata parsers. The research therefore embarked on a rigorous literature review with a view of understanding the main schools of thought within this problem area, assessing the development trends of each and thereafter proposing the most suitable approach. This led the focus of the research to ontology mapping methods where a novel approach known as 'Ontology Concept Modeling (OCM)' was developed and used as the basis for the research.

To address the issue of the development of language and domain independent NL parsers, the research explored various linguistic theories with a view of proposing a universal language theory

that address the requirements of NL parsing regardless of the language. The Generative-Transformational grammar was used in the studies. It formed the theoretical bases of NL parsing. Field surveys were also conducted to collect data regarding NL usage specifically as an input to a relational database access method. Studies were done for Kiswahili and English and characterization done. This led to the development of a novel query modeling framework based on 'kernelization' where a query is converted to its deep structure form before concepts and their relationships can be isolated. The challenge of development of domain independent ontology-metadata parsers was addressed through field data collection and analysis. This led to the development of a novel algorithm for parsing the meta-data which was earlier presented. Algorithms and heuristics were developed for concepts mapping, discovery of implicit concepts as well as handling of foreign keys.

A large portion of the research was dedicated to the evaluation of the OCM model. A prototype was developed for the purpose of evaluation. In addition an elaborate 8-point evaluation framework was presented. The results were presented and bench marked against some of the most successful models in this area.

The section that follows provides details of the specific contributions and achievements that were realized in this research. The contributions and achievements which add to the body of knowledge are sub-divided into two main areas namely, theoretical and technical contributions. Theoretical contributions are further sub-divided into two specific areas which include methodological and non-methodological contributions.

5.2 Theoretical Contributions

In their book *The Unwritten Rules of PhD Research (Open up Study Skills)*, Petre & Rugg (2010) have detailed what constitutes a theoretical contribution in a PhD thesis. They argue that characterizing a theoretical contribution as significant or not amounts to articulating the answers to the following four questions,

- How important is the issue. Is the research question important and why is it worth asking?
- How significant are the findings or the contributions? Why should anyone care? Why do they matter?
- What are the implications to theory and to the body of knowledge in general
- What are the limitations to generalization?

This section discusses those original contributions and achievements made in light of these four tenets.

5.2.1 Modeling of Trends in the Approaches to NL access to Databases

No published source was available that provides a comprehensive analysis of the trends of the methods used in NL access to databases. This research analyzed these trends and documented them. The trends were presented in a concise graphical manner that was presented in section 2.5. Figure 5.1 below re-illustrates this analysis.

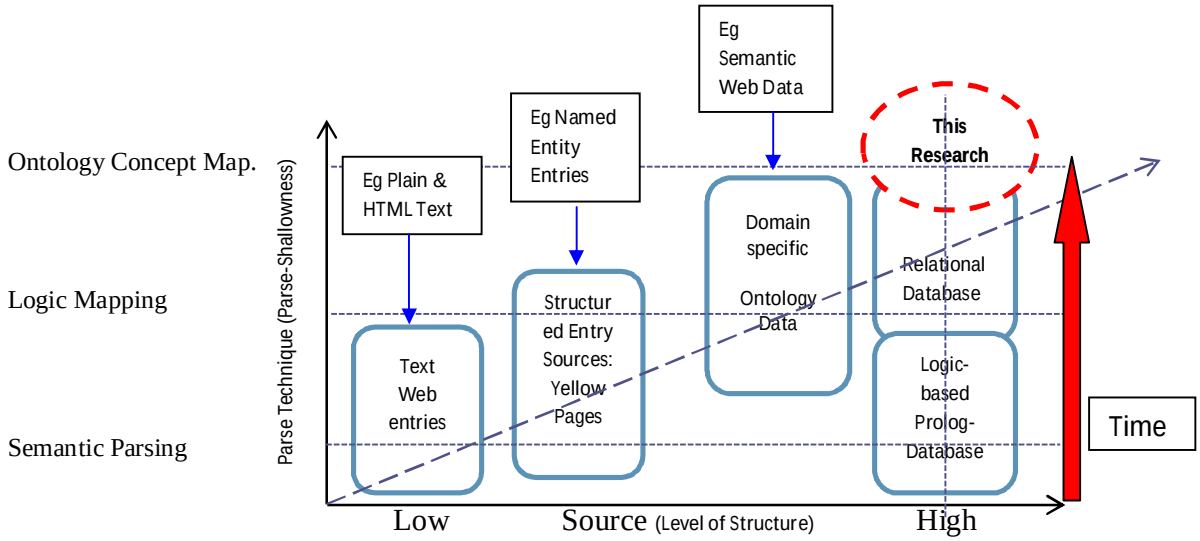


Fig 5.1 Concise Graphical Presentation of Methods and Trends in NL Access to Databases

In analyzing the significance of this contribution to the body of knowledge, the four tenets described in 5.2.1 were applied. It is true that researchers in this area have been grappling with the problem of literature that is widely dispersed and therefore organizing it into distinct schools of thought namely semantic parsing (which includes statistical and machine learning methods), logic mapping (which includes token and phrase based mapping) and Ontology Mapping (which involves the use of ontologies as an intermediate layer) is import theoretical contribution.

5.2.2 Query Semantics Transfer Modeling

Arising from linguistic characterization of NL inputs to a database access model, several important conclusions were made. These included the idea that the kernelization process originally proposed in the Generative-Transformational grammar for sentences, is a viable method of NLQ parsing. Further, it was observed that there exists a regular process in which the general semantics of a query are transferred from the surface structure to the base meaning-bearing components. This process was modeled and presented as the Query Semantics Transfer Model (QuSeT) Model in section 3.3.9.3. The model is re-illustrated in Figure 5.2.

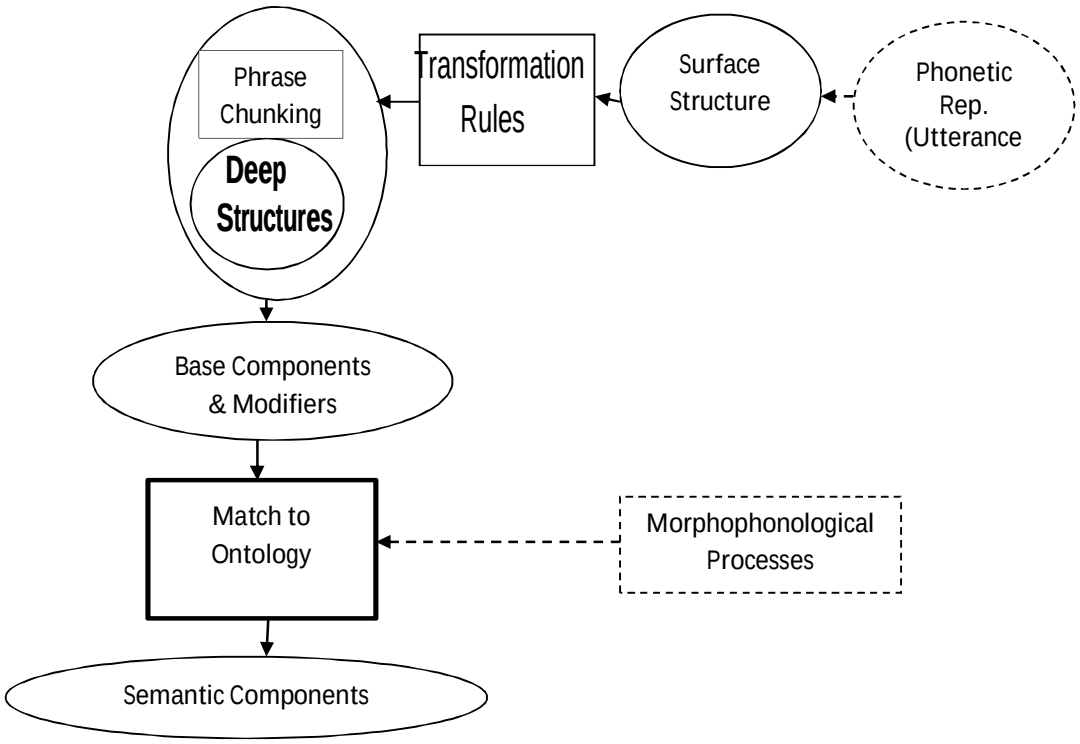


Fig 5.2 Query Semantics Transfer (QuSeT) Model based on Generative-Transformational Grammar

The QuSeT model was qualitatively and quantitatively validated as explained in section 3.3.9.3 and 3.3.9.4 respectively. In the qualitative analysis, the model was validated using primary data collected from one of the surveys where the twelve most prevalent NLQ types were applied and tested through generative-transformation modeling specifically the kernelization process.

This model, whose mean accuracy was determined as 94%, is an important contribution in that NLQs may be converted to the equivalent semantic bearing components which are in turn organized in form of triples that form the backbone of the SPaRQL query. Only the language's generative-transformational, as explained in publications such as Encyclopedia Britanica Inc, (2014), Massamba, Kihore, & Hokororo (1999), Zellig (1951) among others and phrase chunking rules which are expressed as regular expressions are required for this model to be complete for any given language. Once rules have been extracted from a particular language manually, they can then be deployed in a language independent manner as explained in QuSeT.

5.2.3 Ontology Words Recreation Algorithm (OWoRA)

One of the objectives in this work was the creation of a generic parsing method for ontologies created from relational databases. In particular this problem is challenging because no previous studies have been done with a view of characterizing database schema naming methods. Some of the research questions addressed in the database schema authorship survey were as follows,

- *Is there a finite set of patterns that database schema authors' use in representing database schema object names?*
- *'How can we decipher the meaning of 'intended concept' from the schema names?*
- *How can a general 'Concepts Re-construction Algorithm' be built from an ontology created from relational database source?*

The conclusions obtained from the database nomenclature surveys were used in the creation of an Ontology Words Reconstruction Algorithm (OWoRA). This was presented in section 3.4.7. This algorithm, which had an average performance on accuracy of 92.5%, is important contribution to the body knowledge because any RDF-based ontology, such as an OWL ontology created from a relational database, may be parsed by this algorithm to reconstruct the full words that would be used in an equivalent manner if short-forms such as abbreviations and concatenations were not used.

In about 7.5% of the observations automatic methods could not guarantee deduction of meaning from short-hand forms due to over abbreviation or use of acronyms that do not have direct mapping to meaning.

5.2.4 Ontology Concept Model (OCM)

As articulated in chapter one, this research revolved around the provision of a generic language and domain independent methodology for understanding un-restrained natural language text and converting it to structured query language (SPaRQL). Further the research aimed at tackling the challenges encountered in cross-lingual querying especially in the context of Kiswahili-English cross-lingual databases.

Section 2.6.1 presented the conceptual framework that provided the road map to the development of the architectural model presented in section 3.6. The overall algorithm for this model was presented in section 3.76. The following section discusses these key theoretical contributions.

5.2.4.1 Conceptual Framework

The conceptual framework articulated the main processes required for a solution to be obtained. This framework is presented in figure 2.12 and its components discussed in sections 2.6.1 through section 2.64. Given this framework any researcher may design an ontology based method without necessarily going the way of OCM. It is therefore an important theoretical contribution emanating from the literature analysis. From the Petre and Rugg (2010) criteria of assessing significance this is an important contribution with great significance to potential ontology-based researchers.

5.2.4.2 Architectural Model of the OCM

One major contribution of this work is in the design of an architectural model for the OCM approach. Figure 5.3 re-illustrates this architecture named in section 3.6 as *'Architecture for Ontology-based NL Access to DBs (ONLAD)'*. It is an important contribution to the body of knowledge because it has provided a methodology based on the ontology approach that can be used as a template upon which a system is developed. In this work, the architectural model led to the development of a prototype whose performance compared favorably to the cutting-edge-tools.

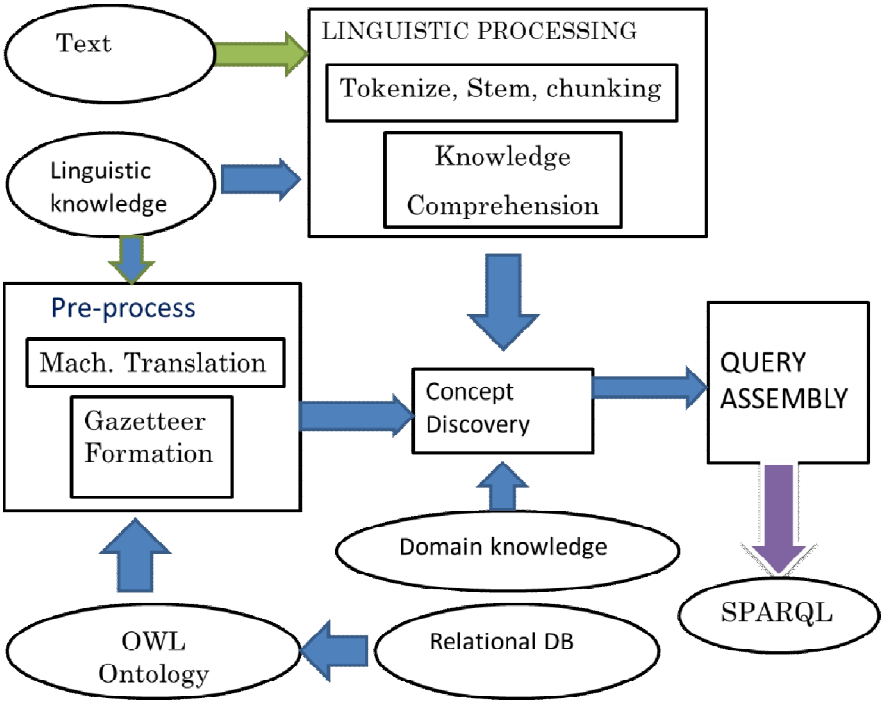


Fig 5.3 Architecture for an Ontology-based NL Access to DBs (ONLAD)

The ONLAD architecture was published as a book chapter in Springer Lecture Notes in Computer Science (LNCS 2013) (Muchemi & Popowich, 2013).

5.2.4.3 Implementation Algorithms

The OCM algorithms were presented in section 3.7 while their implementation is reported in appendix 8. The algorithms and the architectural model described in 3.6 formed the basis for the actual implementation of the OCM-based prototype. Other important components were designed and discussed in detail in various sections of this report.

Section 3.5.1 highlighted the process of designing a suitable feature space model, section 3.5.2 provided details for the design of a gazetteer while section 3.7.1 provided details for the design of a Semantically-Augmented Concept Matching (SACoMA) function.

This research has also brought forth a new approach in the handling of foreign keys that are present in tables and constraining the results formed by SPaRQL triples using the transferred keys. This was developed as a heuristic and was presented in section 3.7.4. Other heuristics included discovery of

implicit concepts and the query generation of actual structured queries in form of SPaRQL. Without these algorithms, designs and heuristics, the actual implementation would have been difficult.

In summary, these designs, algorithms and heuristics form important methodological contributions to the body of knowledge.

The model's significance to the community of researchers is great because it forms the basis for replication or bench-marking other models. The community of developers benefit in that they may use the model to develop systems that achieve high performance as demonstrated by the prototype. Its limitations, such as suffering from low maximum recall and accuracy levels compared to other approaches, were discussed in section 4.10.5 and these may be addressed by future research.

5.2.5 Evaluation Framework

An evaluation framework describes the environment, procedures and parameters used in determining the performance of a model. As established in literature, there does not currently exist a standard or *de facto* evaluation framework for a NL database access model. This work has researched and presented an elaborate evaluation framework informed by a comprehensive review of literature. This research developed an 8-parameter evaluation framework that sought to fill this gap. The framework has been fully described and used to evaluate the OCM model.

In this new framework, traditional practices of determining quantitative parameters such as accuracy, precision and recall were augmented with a further quantitative parameter, the F-score which was borrowed from related NLP tasks such as information extraction. The framework was enhanced by incorporation of four other qualitative parameters which in the context of the objectives of this research, are vital in the conclusive evaluation of performance.

The framework also describes the experimental procedures in detail. The standard deviation analysis, coupled with Peirce's criterion (Ross, 2003), were determined as the most suitable independence analysis measures and were presented in sections 4.1. The detailed experimental and analysis procedures are explained in sections 4.6 through to section 4.8.

As a contribution to the body of knowledge, this research has brought forth an evaluation framework that can be a point of reference in evaluating other database access models.

5.3 Technical Contributions

Petre and Rugg (2010) describe the implementation of theoretical principles as an important contribution to the body of knowledge. In this work several theoretical principles were pooled together and formed important practical contributions which were demonstrated through the prototype illustrated in figure 3.50. Examples of these theoretical principles included the use of Chomsky (1970), Transformational Grammar theory and formation of Kiswahili terms and collocations described in Sewangi (2001). These together with other resources described in 3.8 were constructed into one coherent practical implementation. Some of the important functions which were implemented in python are shown in appendix 8.

This work made further technical contributions in the area of creation of two datasets (queries and databases) that may be used by the research community for development of models and their evaluation. These are the Kiswahili dataset developed for the farming domain and an English dataset developed for the students' queries management. The Kiswahili dataset, being the first in the study of DB access using Kiswahili language may act as a 'gold standard' in the area of testing and evaluation of NL access models.

5.4 Achievements on Performance Advancement

5.4.1 Advancement of F-Score Performance

One significant achievement of this research was in the advancement of the F-Score performance. OCM has an F-score of between 0.774 and 0.784. The token-based graph-matching approach by Popescu et al. (2003) had a minimum F-score of 0.65 while the token and phrase based Kiswahili template mapping (TTM) by Muchemi (2008) has a score of 0.686. On the other hand the most successful machine learning models such as the λ-SCFG grammar based learners by Minock et al. (2008) and the statistical based learners such as WASP by Ge and Mooney (2005) had an F-score of between 0.550 and 0.820 and 0.69 and 0.930 respectively. Machine learning approach usually requires a secondary classifier that converts the meaning representations from a semantic parser such as WASP to a structured query such as SQL. This cascading of machine learning approaches reduces the overall F-score. For example Superimposing an SQL classifier such as that by Giordani & Moschiti, (2010) with an F-Score of 0.759 to say WASP with an F-score of 0.81, the overall DB-Access F-Score would be (0.81x0.759) which is 0.615. This is much lower than OCM's average of 0.78.

The model was tested using Kiswahili and English queries. The average F-score on Kiswahili queries was found to be 0.76 against 0.79 for English queries while precision for Kiswahili ranged between 0.74 and 0.85 against 0.75 and 0.86 for English queries.

The results from the OCM model were bench marked against the state-of-the-art models as presented in section 4.10.The model and the results were published in Muchemi & Popowich (2013b).

5.4.2 Attainment of Domain-Independence

The model was evaluated across four different domains where it was shown through domain-independence experiments and analysis to be domain independent. It was demonstrated in section 4.6 that the OCM model is insensitive to domain change thereby leading to the conclusion that the OCM model is domain independent. This is a significant contribution in that the developers can apply this model to many different domains without deterioration of the performance levels.

5.4.3 Attainment of Language Independence

Experimental results found in section 4.7 showed that OCM is applicable to different languages without deterioration in performance. This universality was achieved through application of universal language theory, specifically 'Transformational Grammar' theory by Chomsky (1970), to augment natural language processing. This is a significant contribution to both research and developers communities.

5.4.4 Achievement of Cross-lingual Querying

Most databases have to grapple with the challenge of cross-lingual interaction. This is the inability of a model to use a certain natural language to query a specified database given that the language used to author the ontology or the database schema objects is different from the one used in querying. Experiments reported in 4.8 showed that the OCM model handles this challenge effectively. This was effected through introduction of a gazetteer which is the manipulated accordingly. The attainment of cross-lingual querying is an important technical contribution.

5.5 Limitations

The main limitation of the developed solution is the lack of machine learning capacity where users' previous inputs, results and feedback can be utilized to improve performance. This means that the

model cannot learn and so will have a static performance level. While this may not necessarily be a bad thing, models which are self-improving are better than static ones.

As earlier stated, the model also suffers from low maximum recall level as evident from results because it requires full information that sometimes may be regarded as obvious or superfluous. The low recall levels would lead to no answers being generated which is better than having SPaRQL generate wrong or unexpected answers. For example the query *'give me customers who come from Nairobi'* might require you to add the word 'name' within the query so that the system realizes we require 'customers' names'. Though OCM tried to avoid this by giving all possible answers, it would be desirable to learn users' behaviour and make intelligent guesses, rather than giving unnecessary information.

Another drawback which is on implementation rather than methodological approach is that the OCM implementation relies on the integration of several modules and resources. If the individual modules are not well implemented, say training of phrase chunkers, the overall result would be lower than the one reported here.

5.6 Recommendations for Further Work

This work has made several contributions to the body of knowledge, however a few areas have the potential to be advanced further. This section highlights these areas

5.6.1 Scalability Study

The model developed in this study supports concurrent access to multiple tables but not to multiple databases. The ability to support multiple databases is important because it gives an indication of the potential for scalability. A study in this area would provide a useful extension to the OCM model because querying can be done concurrently to several databases over a network or even the internet. Research into the incorporation of multiple web crawling agents that carry OCM-based models can possibly solve this problem. Scalability study is an area that is potential for further work.

5.6.2 Discourse Processing Study

As explained earlier, deletion of agents transformation (DAT) occurs in close to 20% of the occurrences. For example, nouns are replaced with pronouns. This is common especially when users ask consecutive questions and they expect the system to 'remember' the subject being discussed. For

example the interrogator may be inquiring on the amount of food a layers chicken requires. In a consecutive question, the interrogator may want to know about the quantity of water required by the same subject. The interrogator makes an assumption that the system has a memory of the subject being discussed and therefore implicitly refers to the layers chicken by the use of a pronoun. This implicit way of referencing concepts leads to lower rates of recall because even though the pronoun is recognized as a subject by the QuSeT model, a direct mapping of this subject to the core-referenced subject is not possible without a context processor. Incorporation of content processing to decipher meaning of pronouns and other deleted agents for DAT transformed sentences would be a useful extension to this work because it would guarantee relatively higher recall rates.

5.6.3 Application of OCM to Object-Oriented Databases

This is another potential area of study in which researchers can explore the possibility of applying OCM to object oriented databases. These databases are gaining traction in the applications world especially in multi-media and game applications and it would be insightful to study, extend and evaluate performance of OCM in this data representation paradigm.

In conclusion this research ventured into the area of natural language access to relational databases and has brought forth new contributions in this area. A high performance architectural model that provides for both natural language query parsing and RDF ontology schema parsing as well as data structures and processing algorithms have been developed and evaluated. Looking back at the set objectives in chapter one, all have been successfully met.

5.7 Relevant Publications and Associated Conferences

BOOK CHAPTERS

Muchemi, L & Popowich, F.(2013). *An Ontology-Based Architecture for Natural Language Access to Relational Databases.* Springer Lecture Notes in Computer Science. HCI (6) 2013: 490-499 Vol. 8009 2013. Las Vegas, USA. ISBN 978-3-642-39188-0

JOURNAL PUBLICATION

Muchemi, L. (2008). Towards Full Comprehension of Swahili NL for Database Querying. *Strengthening the Role of ICT in Development* (pp. 50-58). Kampala-Uganda: Fountain Publishers.

CONFERENCE PROCEEDINGS

Muchemi, L & Popowich, .(2013). *NL Access to Relational Databases: The OCM Approach.* Proceedings of *7th International Conference*, UAHCI 2013, Las Vegas, NV, USA, July 21-26, 2013, Proceedings Part I.

Muchemi, L (2008). *Swahili NL Access to RDbs (TTM Approach).* Proceedings of *4th ICCR Conference*. *Makerere* University, Kampala, Uganda, August 2008

Muchemi, L, Getao K. 2007. Enhancing Citizen-Government Communication Through Natural Language Querying. *Proceedings of 1st International Conference in Computer Science and Informatics (COSCIT 2007*). :150-154., Nairobi, Kenya

Muchemi, L., & Narin'yani, A. (2007). Semantic-base NL Front-End for DB Access: Framework Adoption for Swahili Language. *1st International Conference in Computer Science and Informatics (COSCIT 2007).* Nairobi: UoN-ISBN 9966-7284-0-6 - University of Nairobi.

Bibliography

Akerkar, R., & Joshi, M. (2009). Natural Language Interface Using Shallow Parsing. *International Journal of Computer Science and Applications,*, 70 - 90.

Androutsopoulos , I., Ritchie, G., & Thanisch. (1995). Natural Language Interface to Database: An Introduction. *Journal of Natural Language Engineering, 1*(1), 29-81.

Androutsopoulos. (1993). *A principled Framework for Constructing Natural Language Interfaces to Temporal Databases.* Edinburgh: University of Edinburgh.

Androutsopoulos. (1993). *Interfacing a Natural Language Front-End to a Relational Database.* Department of AI. Edinburgh: University of Edinburgh.

Androutsopoulos, Ritchie, & Thanisch. (1993). MASQUE/SQL- An Efficient and Portable Natural Language Query Interface for Relational Databases. *Sixth International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems.* Edinburgh: Gordon and Breach Publishers Inc., PA, USA.

Banko, M. (2009). *Open Information Extraction for the Web.* Washington DC: University of Washington.

Bernstein, A., Kaufmann, E., & Kiefer, C. (2006). Ginseng: A guided Input Natural language Search Engine for Querying Ontologies. *Jena User Conference.* Bristol, UK.

Bird, C., Loper, E., & Klein, E. (2009). *Natural Language Toolkit with Python.* O'Reilly Media Inc.

Bloom, Englehart, Furst, Hill, & Krathwohl. (1956). *Taxonomy of Educational Objectives: The Classification of Educational Goals- Handbook I: Cognitive Domain.* New York: Longmans.

Bond, T. (2011, June 7). *The Reasoning Process*. Retrieved 2011, from ICT New Zealand: http://ictnz.com/Thinking Pages/reasoning.htm

Bootra, R. (2004). *Natural Language Interfaces Interfaces: Comparing English Language Front End and English Query.* Virginia, USA: Master's Thesi, Virginia Common Wealth University, 2004.

Brill, E. D. (2002). An Analyssis of AskMSR Question-Answering System. *Conference on Empirical Methods in Natural Language Processing.*

Buitelaar, P., & Ciamiano, P. (2006). Ontology Learning from Text -Tutorial at EACL. *11th Conference of European Chapter of Association of Computational Linguistics.* Trento, Italy: AIFBO.

Cavar, D. (2011, June). *Practical use of n-gram models and simple statistics*. Retrieved Nov 2012, from LID - Language Identification in Python : http://www.cavar.me/damir/LID/

Choge, S. (2009). Understanding Kiswahili Vowels. *The Journal of Pan-African Studies, 2*(8).

Chomsky, N. (1957). *Syntactic Structures* (2nd ed.). Berlin, Germany: Walter de Gruyter GmbH and Co.

Chomsky, N. (1970). *English Transformational Grammar.* (R. Jacobs, & P. Rosenbaum, Eds.) Waltham: Ginn.

Creutz, M., Lagus, K., Linden, & Virpioja. (2005). Morfessor and Hutmegs: Unsupervised Morpheme Segmentation for Highly-Inflecting and Compounding Languages. In Langemets, & Penjam (Ed.), *Proceedings of the Second Baltic Conference on Human Language Technologies* (pp. 107-112). Tallin: Tallinn University of Technology.

Csongor, N., Martin, O., & Samson, T. (2009). *DataMaster – a Plug-in for Importing Schemas and Data from Relational Databases into Protégé.* California-USA: Stanford University.

Damljanovic, D., Agatonovic, M., & Cunningham, H. (2010). *Natural Language Interfaces to Ontologies: Combining Syntactic Analysis and Ontology-based Lookup through the User Interaction.* The Semantic Web: Research and Applications.

Damljanovic, D., Tablan, V., & Bontcheva, K. (2008). A Text-based Query Interface Interface to OWL Ontologies. In N. Calzolari (Ed.), *Sixth International Conference on Language Resources and Evaluation (LREC08).* Marrakech, Morocco: European Language Resources Association (ELRA).

De Pauw, G., & Schryver, G.-M. d. (2008). Improving the Computational Morphological Analysis of a Swahili Corpus for Lexicographic Purposes. *Thirteenth International Conference of the African Association for Lexicography, organized by the Bureau of the Woordeboek van die Afrikaanse Taal. 18*, pp. 303 - 318. Stellenbosch: Lexikos (AFRILEX-reeks series).

De Pauw, G., Schryver, M., & Wagacha, P. (2006). Data-Driven Part-of-Speech Tagging of Kiswahili. *Springer-Verlag*, 197-204.

Dittenbach, M., & Berger. (2003). A Natural Query Language Interface for Tourism Information. *ENTER*, 152-162.

Django Project. (2011). *Tagging, Chunking & Named Entity Recognition with NLTK*. Retrieved October 2012, from Python NLTK POS Tagging, IOB Chunking and Named Entity Recognition: http://text-processing.com

Domingos, H., & Poon, P. (2009). Unsupervised Semantic Parsing. *EMNLP*, (pp. 1-10). Singapore.

EasyAsk. (2010). *Comparing Quiri, Siri and Watson*. Retrieved July 22, 2012, from EASYask: http://www.easyask.com/wp-content/uploads/2012/03/Quiri-Siri-Watson-WP.pdf

Encyclopedia Britanica Inc. (2014). Linguistic Articles: Transformational-Generative Grammar. *Harris & Chomsky Grammar*.

Esther, K., Abraham, B., & Renato, Z. (2006). Querix: A Natural Language Interface to Query Ontologies based on Clarification Dialogues. *International Semantic Web Conference (ISW 2006).* Athens, Georgia-USA: ISW2006.

FHI. (2012). *Research Methods Overview.* Retrieved December 2012, from Qualitative Research Methods: A Data Collector's Field Guide: http://www.fhi360.org/nr/rdonlyres/etl7vogszehu5s4stpzb3tyqlpp7rojv4waq37elpbyei3tgmc4ty6dunbccfzxtaj2rvbaubzmz4f/overview1.pdf

Funk, Tablan, Bontcheva, Cunningham, Davis, & Handschuh. (2007). CLOnE: Controlled Language for Ontology Editing. *6th International Semantic Web Conference (ISWC 2007).* Busan, Korea.

Gao, M., Liu, J., Zhong, N., & Chen, F. (2007). A Constraint-based Method for Semanic Mapping from Natural Language Questions to OWL. *Computational Intelligence and Data Mining (2007).* IEEE .

Garcia, K., Lumain, A., Wong, J., Yap, J., & Cheng, C. (2008). Natural Language Database Interface for the Community Based Monitoring System. *22nd Pacific Asia Conference on Language, Information and Computation* (pp. 384-390). Cebu City, Philippines: PACLIC 22.

Ge, R., & Mooney, R. (2005). A statistical Semantic Parser tat Integrates Syntax and Semantics. *CoNLL0-5*, (pp. 9-16). Ann Arbor - Miami.

Gene Ontology Consortium. (2001). *Creating the Gene Ontology Resource: Design and Implementation.* Retrieved March 12, 2012, from Genome Research: http://genome.cshlp.org/content/11/8/1425.full.pdf+html

Gennari, J., Nguyen, M., & Silberfein, A. (2007, May 23). DataGenie:Plug-in for Protege for Reading Databases. University of Washington & Stanford University, USA.

Giordani, A., & Moschitti, A. (2010). Corpora for Automatically Learning to Map Natural Language Questions into SQL Queries. In N. Calzolari (Ed.), *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)* (pp. 2336-2339). Valletta, Malta: European Language Resources Association (ELRA).

Harris, R. (1995). *The Linguistic Wars.* USA: Oxford University Press.

Hart, G., Johnson, M., & Dolbear, C. (2008). Rabbit: Developing a Controlled Natural Language for Authoring Ontologies. *5th European Semantic Web Conference (ESWC08).* Tenerife, Spain.

HCS. (2004). Helsinki Corpus of Swahili. *HCS2004*. Institute for Asian and African Studies (University of Helsinki) and CSC - IT Center for Science.

Hockenmaier, J., & Steedman, M. (2002). Generative Models for Statistical Parsing with Combinatory Categorial Grammar. *40th Annual Meeting of the Association for Computational Linguistics (ACL)*, (pp. 335-342). Philadelphia.

Hockenmaier, J., & Steedman, M. (2007). CCGbank: A Corpus of CCG Derivations and Dependency Structures Extracted from the Penn Treebank., (pp. 1-42).

Hu, W., & Qu, Y. (2008). Discovering Simple Mappings Between Relational Database Schemas and Ontologies. *The 6th International Semantic Web Conference and the 2nd Asian Semantic Web Conference.* Busan- S.Korea.

Hurskainen, A. (1992). A Two Level Computer Formalism for the Analysis of Bantu Morphology: An application to Swahili. *Nordic Journal of African Studies, 1*(1), 87-119.

Hurskainen, A. (1999). SALAMA:Swahili Language Manager. *Nordic Journa lof African Studies, 8*(2), 139-157.

Hurskainen, A. (2004). HCS-2004. *Helsinki Corpus of Swahili*. Helsinki, Finland: University of Helsinki and CSC.

Iribe, M. (2008). *A Synchronic Segmental Morphophonological of Standard Kiswahili.* University of Nairobi, CHSS. Nairobi: University of Nairobi erepository.

Jeff, W. (2011, October 5). *How Siri Works*. Retrieved July 20, 2012, from http://www.jeffwofford.com/?p=817

Jurafsky, D., & Gildea, D. (2002). Automatic Labelling of Semantic Roles. *Association for Computational Linguistics, 28*.

Kamusi Project. (2013). Kamusi Gold. *Global Online Living Dictionary*(Multilingual Beta). USA.

Kate, & Mooney. (2010). *Geoquery*. Retrieved March 24, 2012, from University of Texas at Aiustin - AI Lab: http://www.cs.utexas.edu/~ml/nldata/geoquery.html

Kate, J., & Wong, Y. (2010). *Semantic Parsing: The Task, the State of the Art and the Future.* Uppsala-Sweden: ACL-2010.

Kate, R. M. (2007). Semi-Supervised Learning forSemantic Parsing using Support Vector Machines. *Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics, Short Papers (NAACL/HLT-2007)* (pp. 81-84). Rochester, NY: NAACL/HLT.

Kate, R., & Mooney, R. (2006). Using string-kernels for learning semantic parsers. *COLING/ACL 2006*, (pp. 913-920). Sydney, Australia.

Kate, R., & Mooney, R. (2007). Semi-Supervised Learning forSemantic Parsing using Support Vector Machines. *Human Language Technology Conference of the North American Chapter for the Computational Linguistics (NAACL/HLT)*, (pp. 81-84). Rochester - NY, USA.

Kaufmann, E., & Bernstein, A. (2007). How Useful are Natural Language Interfaces to the Semantic web for Casual End-Users? *Fourth European Semantic Web Conference(ESWC).* Innsbruck-Austria.

Kaufmann, E., Berstein, A., & Fischer, L. (2007). NLP-Reduce: A "naive" but Domain Independent Natural Language Interface for Querying Ontologies. *4th European Semantic Web Conference (ESW2007).* Innsbruck-Austria.

Kellog, & Reed. (1877). *Higher Lessons in English.* Chicago: Unversity of Chicago Press.

Keshavarz, M., & Lee, Y.-H. (2012). Ontology matching by using ConceptNet. In Kachitvichyanukul, Luong, & Pitakaso (Ed.), *Asia Pacific Industrial Engineering & Management Systems Conference 2012*, (pp. 1917-1925).

Korokithakis, S. (2008, January 17). *Finding the Levenshtein Distance in Python*. Retrieved May 10, 2010, from Stavro's Stuff: http://www.korokithakis.net/posts/finding-the-levenshtein-distance-in-python/

Krishnamurthy, J., & Mitchell, T. (2011). Which Noun Phrases Denote Which Concepts? *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies.* Association for Computational Linguistics.

KU. (2011, 06 09). *Interrogatives.* Retrieved August 2013, from KU: www2.ku.edu/~kiswahili/pdfs/lesson_32.pdf

Kwok, Etzioni, & Weld. (2001). Scaling Question Answering to the Web. *ACM Transactions on Information Systems. 19*, pp. 242-262. ACM.

Levett-Jones. (2009). *Clinical Reasoning.* Retrieved 2011, from Instructor Resources.

Liddy, E. D. (2005). Automatic Document Retrieval. *Encyclopedia of Language & Linguistics, 2nd Edition.*

Lopez, V., Motta, E., Uren, V., & Sabou, M. (2007). *State of the Art on Semantic Question Answering - A Literature Review.* The Open University. 2007: Knowledge Media Institute - UK.

Lopez, V., Pasin, M., & Motta, E. (2004). AquaLog: An Ontology-Portable Question Answering System for the Semantic Web.

Lund Research. (2012). *Sampling Strategy*. Retrieved December 15, 2012, from Laerd Dissertation: http://dissertation.laerd.com/sampling-strategy.php

Massamba, D., Kihore, Y., & Hokororo, J. (1999). *Sarufi Miundo ya Kiswahili Sanifu (SAKISA).* Dar es Salam, Tanzania: Taasisi ya Uchunguzi wa Kiswahili, University of Dar es Salam.

Mateas, M., & Stern, A. (2011). *Natural Language Understanding In Facade: Surface Text Processing.*

Microsoft. (2004). *Northwind and Pubs Sample Databases for Microsoft SQL Server 2000.* Retrieved 2011, from http://www.microsoft.com/downloads/details.aspx?FamilyID=06616212-0356-46a0-8da2-eebc53a68034&displaylang=en

Miller, G. (1995). Wordnet: A Lexical Database for English Communications of the ACM. *ACM, 38*(11), 39-41.

Miller, S., Stallard, R., Bobrow, R., & Swartrtz, R. (1996). A fully statistical approach to natural language interfaces. *ACL*, (pp. 55-61). Santa Cruz, CA.

Mingxia, G., Jiming, L., Ning, Z., & Furong, C. (2007). A Constraint-based Method for Semantic Mapping from Natural Language Questions to OWL. *2007 IEEE Symposium on Computational Intelligence and Data Mining (CIDM 2007).* IEEE Xplore.

Minker, J. (1997). Information storage and retrieval - a survey and functional description. 1-108.

Minock, M., Olofsson, P., & Naslund. (2008). Towards Building Robust Natural Language Interfaces to Databases. In E. Kapetanios, V. Sugumaran, & Spiliopourou (Ed.), *Natural Language and Information Systems: 13th International Conference on Applications of Natural Language to Information Systems, NLDB- London, UK. 5039*, pp. 187–198. Hiedelberg: Springer.

Mooney, R. (2007). Learning for Semantic Parsing. *Eith International Conference on Computational Linguistics and Intelligent Text Processing* (pp. 311-324). Mexico City, Mexico: Springer.

Muchemi, & Popowich. (2013b). An Ontology-Based Architecture for Natural Language Access to Relational Databases. *Human Computer Interaction International-2013* (pp. 490-499). Las Vegas, Nevada-USA: UAHCI/HCII 2013.

Muchemi, L. (2008). Towards Full Comprehension of Swahili NL for Database Querying. *Strengthening the Role of ICT in Development* (pp. 50-58). Kampala-Uganda: Fountain Publishers.

Muchemi, L., & Narin'yani, A. (2007). Semantic-base NL Front-End for DB Access: Framework Adoption for Swahili Language. *1st International Conference in Computer Science and Informatics (COSCIT 2007).* Nairobi: UoN-ISBN 9966-7284-0-6 - University of Nairobi.

Muchemi, L., & Popowich, F. (2013). An Ontology-based Architecture for Natural Language Access to Relational Databases. In C. Stephanidis, & M. Antona, *Springer Lecture Notes in Computer Science* (Vol. 8009, pp. 490-499). Berlin Heidelberg: Springer-Verlag.

Mugenda, A., & Mugenda, O. (2003). *Research Methods: Quantitative and Qualitative Approaches.* Nairobi, Kenya: African Center for Technology Studies.

Munir, Odeh, & McClatchey. (2008). Ontology Assisted Query Reformulation using the Semantic and Assertion Capabilities of OWL-DL Ontologies. *ACM International Conference*, *299*, pp. 600-623.

Nakorn, T. N. (2009). *Combinatory Categorial Grammar Parser in Natural Language Toolkit.* Scotland: University of Edniburgh.

Ohly, R. (1982). Lexicographic Research at the Friendship Textile Mill. *Journal of the Institute of Kiswahili Research*, 73-86.

Ontology, t. G. (2012). *The Scope of GO*. Retrieved March 17, 2012, from An Introduction to the Gene Ontology: http://www.geneontology.org/GO.doc.shtml

Oracle. (2008). Sample Schema Scripts and Object Descriptions. In Leyderman, Austin, Bauwens, Dinesh, Drake, Greenberg, . . . Pataballa, *Oracle Databases - Sample Schemas 11g Release 1 (11.1).* California-94065: Oracle USA Inc.

Paice, C. (1990). Another Stemmer. *ACM SIGIR*, 56-61.

Petre, M., & Rugg, G. (2010). *The Unwritten Rules of PhD Research (Open Up Study Skills).* UK: Amazon.

Ponte J, C. B. (1998). A Language Modeling Approach to Information Retrieval. *SIGIR'98*, 275-281.

Popescu, A., Etzioni, O., & Kautz, H. (2003). Towards a Theory of Natural Language Interfaces to Databases. *2003 International Conference on Intelligent User Interfaces.*, (pp. 149-157).

Popescu, Armanasu, Etzioni, Ko, & Yates. (2004). ModernNaturalLanguageInterfacestoDatabases: ComposingStatisticalParsingwithSemanticTractability. *The 20th International Conference on Computational Linguistics.* Geneva-Switzerland: COLING04.

Port, R. (1982). Morphophonemics of Swahili Verb Suffixes. *Studies in African Linguistics*, 249-271.

Porter, M. (1980). An Algorithm for Suffix Stripping. *Journal Program*, 313-316.

Porter, M., Robertson, S., & Rijsbergen, C. (1980). *New models in probabilistic information retrieval.* London: British Library. British Library Research and Development.

Punyakanok, V., Roth, D., & Yin, W. (2004). Mapping Dependencies Trees: An application to Question Answering. *8th International Symposium on Artificial Intelligence and Mathematics.* Fort Lauderdale FL,USA.

Ran, A., & Lencevicius, R. (2012). *Natural Language Query System for RDF Repositories.* Retrieved March 12, 2012, from http://alumni.cs.ucsb.edu/~raimisl/SNLP.camera.pdf

Rashid, A., Mohammad, A.-K., & Rahman, A. (2009). Efficient Transformation of a Natural Language Query to SQL for Urdu. *Conference on Language & Technology 2009.* Peshawar Pakistan.

Robertson S, S. J. (1976). Relevance Weighting of Search Terms. *Journal of the American Society for Information Sciences, 27*(3), 129-46.

Robin. (2010, November 1). *Natural Language Processing*. Retrieved February 12, 2012, from Natural Language Understanding: http://language.worldofcomputing.net/understanding/natural-language-understanding

Ross, S. (2003). Peirce's criterion for the elimination of suspect experimental data. *Journal of Engineering Technology*, 1-12.

Ruiz-Martınez, J., Castellanos-Nieves, D., Valencia-Garcıa, R., Fernandez-Breis, J., Garcıa-Sanchez, F., Vivancos-Vicente, P., . . . Martınez-Bejar, R. (2009). Accessing Touristic Knowledge Bases through a Natural Language Interface. (D. Richards, & B. Kang, Eds.) *PKAW 2008, LNAI 5465*, 147-160.

Salton G, F. E. (1983). Extended Boolean Information Retrieval. *Communications of ACM, 26*(11), 1022-36.

Salton G, W. A. (1975). A Vector Space Model for Automatic Indexing. *Communications of the ACM - Information Retrieval and Language Processing*, 613-620.

Salton, G. (1971). *The SMART Retrieval System.* Englewood Cliffs, N.J.: Prentice Hall.

Sewangi, S. (2001). *Computer Assisted Extraction of Terms in Specific Domains: The Case of Swahili.* Helsinki Netherlands: PhD Doctoral Thesis, University of Helsinki.

Shin, D.-G., & Chu, L.-Y. (1998). Establishing Logical Connectivity between Query Key Words and Database Contents. *12th Biennial Conference of the Canadian Society for Computational Studies of Intelligence on Advances in Artificial Intelligence .* UK: Springer-Verlag London.

Smart, P. (2008). *Controlled Natural Languages and the Semantic Web.* University of Southampton, School of Electronics and Computer Science. Southampton - UK: International Technology Alliance.

Swier, R., & Stevenson, S. (2004). Unsupervised Semantic Role Labeling. *Proceedings of 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP 2004)* (pp. 95-102). Barcelona- Spain: EMNLP.

Tablan, V., Damljanovic, D., & Bontchev, K. (2008). A Natural Language Query Interface Structured Information. *5th Annual European Semantic Web Conference (ESWC).* Ternerife-Spain.

Tang, L., & Mooney, R. (2001). Using multiple clause constructors in inductive logic programming for semantic parsing. *12th European Conference on Machine Learning (ECML-2001),* (pp. 466-477). Freiburg, Germany: ECML-2001.

Thomson, C., Mooney, R., & Tang, L. (1997). Learning to Parse Natural Language Database Queries into Logical Form. *Automata Induction, Grammatical Inference and Language Acquisition.* Nashville-TN: ICML-97.

TUKI. (2000). *Kamusi ya Kiswahili-Kiingereza.* Dar es Salaam, Tanzania: Taasisi Ya Uchunguzi wa Kiswahili Chuo Kikuu Cha Dar es Salaam.

Vanessa Lopez, E. M. (2007). *State of the art on Semantic Question Answering.* The Open University, Knowledge Media Institute. Milton Keynes,: Knowledge Media Institute, - U.K.

Vertica Systems. (2011, June 27). *Analytic Databases: Getting Started Guide*. Retrieved 2013, from Vertica.

Wang, C., Xiong, M., Zhou, Q., & Yu, Y. (2007). A Portable Natural Language Interface to Ontologies. *4th ESWC* (pp. 473-487). Innsbruck: Springer-VerLag.

Wong, Y. (2005). *Learning for Semantic Parsing Using Statistical Machine Techniques.* Austin - USA: University of Texas at Austin.

Wu, Z., Chen, H., Cui, M., & Yin, A. (2007, May). Semantic-based Search and Query System for the Traditional Chinese Medicine Community. *W3C Publication - Semantic Web Use-Cases and Case studies and China Academy of Chinese Medicine Sciencies(CACMS)*.

Yates, A., Etzioni, O., & Weld, D. (2003). A Reliable Natural Language Interface To Household Appliances. *IUI*.

Yin, R. (1994). *Case Study Research: Design and Methods.* Newbury Park, California: Sage Publications.

Zelle, R., & Mooney, R. (1996). Learning to Parse Database Queries Using Inductive Logic Programming. *AAAI*, (pp. 1050-1055). Portland-Oregon.

Zellig, H. (1951). *Structural Linguistics.* University of Chicago Press, Chicago: Phoenix Books.

Zettlemoyer, L., & Collins, M. (2005). Learning to Map Sentencies to Logical Form. *Twenty First Conference on Uncertainty in Artificial Intelligence* (pp. 658-666). Edinburgh, Scotland: AUAI Press.

Zorzi, I., Tessaris, S., & Dongilli, P. (2007). Improving Responsiveness of Ontology-Based Query Formulation. (G. Semeraro, E. Sciascio, & C. Stoermer, Eds.) *SWAP:CEUR-WS.org*.

Zucker, D. (2009). *How to Do Case Study Research.* Massachusetts-Amhrest, USA: School of Nursing Faculty Publication Series. Paper 2.

Appendix 1: Characterizing Linguistic Features of user Inputs

Segment of Farmers Query Set (Reprinted from original Set (Muchemi L. , 2008) **)**

1.	Ametoka nchi ipi	Which country Ametoka
2.	Ana tabia gani	What Has character
3.	inataga kwa mda gani	inataga for what time
4.	vifaranga ni bei gani	What is the price chick
5.	wakisha komaa nitauzaje	When mature, they will uzaje
6.	baada ya kutaga nitauza aje	After I sell come lay
7.	nitaagiza vifaranga kupitia nani	I suffered through the chick who
8.	nitaletewa vifaranga siku ngapi baada ya kuagiza	I brought the chick, how many days after ordering
9.	nitabebewa vifaranga na nani	Who will bebewa chick
10.	kuku wakigojeka nitamwona nani	hen I saw who they gojeka
11.	kuna vipingo ngapi vya ukuaji	How much growth there Vipingo
12.	unaweza badilisha chakula bila kuangalia watengenezaji	You can eat without looking at the developers changed
13.	nafaa kuwapa kuku maji kiasi kipi	nafaa much water to give a chicken
14.	kuna shida maji yakimwagika sakafuni	There are water problems on the floor yakimwagika
15.	vyumba vya kuku vinafaa kujegwaje	You should kujegwaje chicken rooms
16.	vinafaa kujenga vikielekea jua au la	You should know or not to build vikielekea
17.	chini kwa sakafu inafaa kukorogwa au la	under the floor or the appropriate kukorogwa
18.	ni chombo kipi kinafaa cha kuleta joto inayofaa	What are expedient tool to bring the appropriate temperature
19.	wakati gani mtu anafaa kujua joto limezidi	when does a person need to know the temperature limezidi
20.	nibaridi kiasi gani inatakikana	How nibaridi inatakikana
21.	chombo kipi kinafaa kutumika	What are expedient tool used
22.	ni vyombo vipi vinafaa kwa usafi.	How nivyombo You should clean.
23.	kuku wanafaa kuachana katika ukuaji na “gap” gani	They should stop the chicken growth and, Äúgap, or what
24.	kuku akikomaa anafaa kuwa na uzito kimo gani	If chicken is perfect height and weight should be what
25.	dawa huharibika baada ya mda upi	medicine perishes after what time
26.	kuku anayepigwa na wengine anafaa kutengwa	chickens that received by others is appropriate isolation
27.	nidalili gani zilizo za kawaida kuku akiugua	nidalili What if chickens are common illnesses
28.	nafaa kutumia dawa gani	What medications nafaa
29.	ni njia gani mwafaka ya kuzuia magonjwa	What is the best way to prevent diseases
30.	unaweza kula kuku mgonjwa	You can eat sick chickens
31.	unajua aje kuku amefikisha wakati wake wa kuuzwa	You know how chicken is delivered during his sold
32.	wanunuzi bora	best buyers
33.	bei bora	best price
34.	je tunaweza kutafutiwa soko	how can we market kutafutiwa
35.	unajua aje kuku amefikisha …	** You know how chicken is delivered, a ¶
36.	aina hii imetoka nchi gani	** This kind from what country
37.	Kukua na kuishi kukoje	** Growth and living kukoje
38.	anataga kwa mda upi	** What time he lay with

39.	vifaranga ni pesa ngapi	** Chick is how much
40.	bei yao wakisha komaa ni ngapi	** Their price when they mature is how many
41.	baada ya kutaga nitauzaje	** After I lay uzaje
42.	nitaagiga vifaranga kupitia nani	** Who will agiga chick through
43.	nitaletewa vifaranga siku ngapi baada ya kuagiza	** I brought the chick, how many days after ordering
44.	nitabebewa vifaranga na nani	** Who will bebewa chick
45.	kuku wakiugua nitamwona nani	** I saw chickens who were suffering from
46.	kuna vipingo ngapi vya ukuaji	How much growth there Vipingo
47.	je ni salama kubadilisha chakula bila kuzingatia mtengenezaji	Is it safe to change the food regardless of manufacturer
48.	nafaa kuwapa kuku maji	nafaa giving chickens water
49.	kiasi kipi cha maji	much water
50.	kuna shida yakimwagika sakafuni	There are problems on the floor yakimwagika
51.	kuku wanafaa kuachana katika ukuaji na “gap” gani	They should stop the chicken growth and, Äúgap, or what
52.	kuku akikomaa anafaa kuwa na uzito kimo gani	If chicken is ready to be mature height and weight did
53.	dawa huharibika baada ya mda upi	medicine perishes after what time
54.	kuku anayepigwa na wengine anafaa kutengwa	chickens that received by others is appropriate isolation
55.	nidalili gani zilizo za kawaida kuku akiugua	nidalili What if chickens are common illnesses
56.	nafaa kutumia dawa gani	What medications nafaa
57.	ni njia gani mwafaka ya kuzuia magonjwa	What is the best way to prevent diseases
58.	unaweza kula kuku mgonjwa	You can eat sick chickens ====
59.	kuku wakutaga mayai anapatikana aje	wakutaga chicken eggs is found to come
60.	naweza kumpata kuku wa nyama	I find chicken meat
61.	kifaranga wa kuku wa nyama anapatikana na pesa ngapi	chick chicken meat is available and how much
62.	je, unastahili kuagiza kuku wako kabla ya siku ngapi	Do you need to order your chickens before how many days
63.	je, ni chakula kipi unaweza patia kuku wa nyama na wa mayai	Is what you eat can give the chicken meat and eggs
64.	kuku wa nyama anastahili kuwa na kilo ngapi kwa siku arobainne	poultry meat should be and how many kilograms arobainne
65.	vyumba vyafaa kujengwa kwa nini	Why vyafaa apartments built
66.	vifaa vipi vya faa kutumiwa kupima joto	how materials should be used to measure temperatures of
67.	nyumba ya faa kusafishwa vipi	How the house should be cleaned
68.	vyumba vya faa kusafishwa baada ya mda upi.	rooms should be cleaned after what time.
69.	kuku afaa kuwa na kilo ngapi baada ya wiki 6	afaa a chicken, how many kg after 6 weeks
70.	anapewa chajo baada ya mda upi	He received chajo after what time
71.	anapewa chajo wapi	Where is given chajo
72.	kuhara ni ugonjwa	diarrheal disease is
73.	bona kuku hutetemeka	Why chickens are trembling
74.	kuku wanao kuwa na kukosa kutembea kwa nini.	lack of chickens who had walked for what.
75.	kuku akihara apewa dawa zipi	chicken, and diarrhea what drug apewa
76.	akitetemeka afanywe nini	What trembling afanywe
77.	naweza kuzuia kuhara vipi	How can I prevent diarrhea

78. kuku wa nyama auzwe baada ya siku ngapi	auzwe chicken meat after how many days
79. kuku wa nyama auzwe wapi.	Where poultry meat auzwe .
80. broilers?	Broilers?
81. shilingi hamsini kila kifaranga	fifty shillings each chick
82. unaenda kujichukulia	You go kujichukulia
83. mwangaza wa masaa ishirini na nne.	manifestation of twenty-four hours .
84. kuku aina ngapi za kuku za mayai	How many types of poultry chicken egg
85. kuna aina ngapi za kuku	There are how many types of poultry
86. kuna aina ngapi za chakula	How many kinds of food there
87. kuku za mayai ni bei gani	hen's eggs, what is the price
88. kuku za nyama ni bei gani	poultry meat, what is the price
89. kuagiza ni nini (how do I book)	What order is (how do I book)
90. mtu huchukua wiki ngapi	How someone can take weeks
91. kuku za paswa kupewa chakula mara ngapi	of chicken meat should be many times
92. kuku aina ngapi za chakula	How many types of poultry meat
93. kuku za nyama zinapaswa kunywa maji mara ngapi	poultry meat should drink water often
94. kuna aina ngapi za maji	There are how many types of water
95. nyumba ya kuku inapaswa kujengwa aje	The chicken house should be built to come
96. unapaswa kuwasha taa kila siku	You should light a candle every day
97. kuku za nyama huuzwa kwa bei gani	poultry meat is sold for what price
98. kuku za [mayai] huuzwa bei gani	hen the [eggs] are sold, what price
99. chakula cha kuku wa mayai ni bei gani	eat the chicken's eggs, what is the price
100.utapata wapi soko ya kuku za mayai	Where will the market find a chicken egg
101.utapata wapi soko ya kuku za nyama	Where you will find a market of poultry meat
102.kama za kienyenji	as kienyenji
103.kama za mayai	as egg
104.kama za nyama	as meat
105.ya kienyenji pesa ngapi	of how much kienyenji
106.za nyama pesa ngapi	How much money meat
107.za mayai pesa ngapi	how much egg
108.nani ataniletea	Who will me
109.nanikijichukulia	Nanikijichukulia
110.utanidai pesa ukiniletea	the money you gave me nidai
111.naweza pata aina ya kutaga	I get kind of lay
112.kuna aina ya nyama	any kind of meat
113.nay a nyama na mayai	nay a meat and eggs
114.ya kutaga pesa ngapi	how much of the lay
115.ya mayai pesa ngapi	of how much eggs
116.50 za mayai pesa ngapi	50 eggs, how much
117.kuchukua kuku uanze vipi	How to take chicken begin
118.kuagiza ningonje mpaka lini	When ordering ningonje until

119.kuku 50 pesa ngapi kuagiza	how much chicken to order 50
120.nitaletewa baada ya siku ngapi	I brought how many days
121.ninani atanichukulia kuku	Who will take chicken
122.zinachukuliwa wapi	Where zinachukuliwa
123.vifaranga wa nyama wanakaa kwa wiki ngapi	chick of the meat they spent how many weeks
124.vifaranga wa nyama wakula chakula kipi	chick eat meat what
125.broilers ni zipi na ni za nini	What are broilers and what are
126.layers ni zipi	What layers are
127.za mayai ni pesa ngapi	of eggs is how much
128.zikichinjwa pesa ngapi	how much zikichinjwa
129.mtu anaagiza wakati upi	What is allowed during one
130.mtu anangojea siku ngapi	How many days a person is waiting
131.nitachukua na nini gari ‘carton ‘ama nini	What will I take the car, Äòcoton, what Äòama

Elf’s Queries to Microsoft’s North-Wind Database Questions

Segment of Trade Query Set (Reprinted from original Set by (Bootra, 2004))

1. where are the suppliers from Germany located
2. show the names and complete address of the biscuit companies
3. at which company does Ian work
4. who handles the specialty items(Modify to: who supplies speciality items?)
5. show the domestic suppliers
6. show the New Orleans suppliers
7. show the New England suppliers
8. which company handles the specialty products
9. which companies have Product Managers
10. show the Product Managers
11. show the orders by Leverling to Hanover Sq
12. which products come in bottles
13. What are the names of our Canadian customers?
14. Give the name and location of suppliers from Germany.
15. Which are our Australian suppliers?
16. List the countries where suppliers are located, arranging the countries in alphabetical order.
17. Products with names that start with "La".
18. Suppliers who are not located in Canada
19. Find the products that have between 10 and 20 units in stock
20. Records for customers who are located in Canada and whose names begin with the letter "M"
21. Suppliers who are located in Canada and whose names begin with the letters A-N.
22. Suppliers who have a fax number
23. Show the employees hired between May 1, 1992 and June 1, 1993
24. Employees who live in the United Kingdom or employees who live in Seattle
25. Orders placed before 1-Jan-93
26. Customers whose company names start with N-Z and who are located in either the United Kingdom or Paris
27. Orders that were placed during the month of February 93
28. Find customers from Canada or the UK who have placed over 15 orders
29. Suppliers who provide seafood products and who are from Singapore or Japan.
30. Find the customers who ordered the "Chef Anton's Cajun Seasoning" product
31. information on orders that were placed after 31-Mar-92, including the employee who made the sale and the customer who placed the order

32. What's the average price of all our products
33. Give the name and id for each category.
34. List the customers
35. Count the orders that have been placed for each seafood product
36. Show the ship date and order subtotals since March of 1994
37. Display the subtotal and shipping date of all orders
38. List the suppliers in alphabetical order
39. Find the total number of Northwind suppliers
40. orders that were shipped today
41. orders that were shipped during the past ten years
42. The number of orders that were shipped within the past 3100 days
43. Find the total value of orders that have been shipped to each country
44. Which products cost between $3 and $6?
45. Give the order id, product name, product id, price, quantity, discount and extended price for each purchase
46. Show catalog information for the active products.
47. the minimum price of all products in the Products table
48. all records with the current date
49. What's the total number of orders we received this month
50. all employees who have birthdays today
51. all employees who have birthdays on July 2
52. All employee records that contain photos
53. Find the total number of customers in Canada or the United Kingdom who have placed orders, and group them by country
54. Find the total value of orders shipped to each customer within each country
55. Which employee sold the most units of tofu?
56. Subtotal and customer for orders shipped between 10/1/91 and 12/31/91, sorting on the value
57. photos of employees whose last names start with "B"
58. show photos of employees hired during 1991
59. which customers have ordered both Konbu and Filo Mix?
60. which products are more expensive than chai
61. how much does chai cost
62. customers that ordered both chai and filo
63. how many products are there in each category
64. which customers have ordered every meat/poultry product
65. which customers have never ordered seafood
66. which customers ordered Longlife tofu but not filo mix
67. which customers always use Federal Shipping
68. which product costs the most
69. which customers have placed more orders than average
70. show the seafood products in reverse price order
71. customers that have ordered from both Ma Maison and Tokyo Traders
72. show company names of the suppliers that have more than 3 products
73. which orders were neither shipped to Canada nor sent via Speedy Express
74. which orders were not both shipped to Canada and sent via Speedy Express
75. how many customers have ordered every meat/poultry product
76. what percentage of customers have ordered every meat/poultry product
77. which customers bought products from every category
78. which customers ordered the fewest items
79. show the names and complete address of the pear companies
80. which of the clients that purchased tofu have also purchased chai?
81. Show the ship date and subtotals for all orders since March of 1991
82. how many customers in each country have ordered tofu?
83. which customers exclusively use Federal Shipping
84. which customers use Federal Shipping exclusively
85. customers that work at 12 Orchestra Terrace
86. customers in the t2f area

87. count the orders for tofu versus those for chai
88. graph the number of tofu or chai orders
89. graph the number of Seattle employees against London
90. graph the sum of subtotals for seafood against beverages
91. graph the average subtotal for each category
92. graph the sum of subtotals for tofu, chai and konbu
93. show the average number of products sold by each employee sales representative
94. compare the average unit price showing employee and product
95. which products were shipped by Federal in the last 5 years
96. list employees with home phones = (206) 555-8122, (206) 555-8122
97. Find the total number of different customers in Canada or UK who have placed orders
98. find the total number of DISTINCT customers in Canada or the United Kingdom who have placed orders, and group them by country
99. which suppliers have order dates that are newer than 600 months old
100. show the difference between discount and unit price

COMPUTER JOBS QUERIES

Segment of Jobs Search Query Set (Recreated from Original set by Tang & Mooney, 2001)

1. 'All of it?'
2. 'All the jobs please?'
3. 'All?'
4. 'Any jobs available using database?'
5. 'Are there ada jobs outside austin?'
6. 'Are there any autocad jobs open?'
7. 'Are there any computer jobs for the playstation?'
8. 'Are there any computer jobs in the field of statistics?'
9. 'Are there any jobs at applied materials that require a bscs?'
10. 'Are there any jobs at dell that require no experience and pay 50000?'
11. 'Are there any jobs for a client server specialist?'
12. 'Are there any jobs for a data warehousing specialist?'
13. 'Are there any jobs for a graphics specialist?'
14. 'Are there any jobs for a odbc specialist?'
15. 'Are there any jobs for a programmer?'
16. 'Are there any jobs for people in austin that want to program in lisp but do not have a degree?'
17. 'Are there any jobs in austin developing games in x86 using assembly?'
18. 'Are there any jobs in austin paying over 100000 per year?'
19. 'Are there any jobs in austin requiring at least a bscs and knowing latex?'
20. 'Are there any jobs in austin?'
21. 'Are there any jobs in 'c++' that the salary is 50000?'
22. 'Are there any jobs in houston?'
23. 'Are there any jobs in lan?'
24. 'Are there any jobs in san antonio?'
25. 'Are there any jobs in tcp ip?'
26. 'Are there any jobs in the us with the title verification engineer?'
27. 'Are there any jobs in usa?'
28. 'Are there any jobs on ibm?'
29. 'Are there any jobs on novell?'
30. 'Are there any jobs on pc?'
31. 'Are there any jobs on sun?'
32. 'Are there any jobs on vax?'
33. 'Are there any jobs on windows?'
34. 'Are there any jobs requiring a bscs for boeing in seattle?'
35. 'Are there any jobs requiring ba?'
36. 'Are there any jobs requiring bs?'

37. 'Are there any jobs requiring bscs?'
38. 'Are there any jobs requiring bsee?'
39. 'Are there any jobs specializing in ai with jpl?'
40. 'Are there any jobs that do not require 5 years of experience?'
41. 'Are there any jobs that require a knowledge of linux in san antonio?'
42. 'Are there any jobs that require the knowledge of linux platform?'
43. 'Are there any jobs using assembly in usa?'
44. 'Are there any jobs using 'c++' with dell?'
45. 'Are there any jobs using cobol?'
46. 'Are there any jobs using java that are not with ibm?'
47. 'Are there any jobs using java that are not with tivoli?'
48. 'Are there any jobs using java that dont require a bscs?'
49. 'Are there any jobs using powerbuilder?'
50. 'Are there any jobs using sql?'
51. 'Are there any jobs using 'vc++'?'
52. 'Are there any jobs with a salary of 100000?'
53. 'Are there any jobs with microsoft involving sql?'
54. 'Are there any jobs with microsoft?'
55. 'Are there any mac jobs open?'
56. 'Are there any mac programmer jobs open in austin?'
57. 'Are there any mac programmer jobs?'
58. 'Are there any programmer jobs open?'
59. 'Are there any project manager positions open?'
60. 'Are there any software developer jobs requiring bs?'
61. 'Are there any systems administrator jobs in austin?'
62. 'Are there any unix jobs?'
63. 'Are there jobs that do not require a degree in houston?'
64. 'Are there jobs using vb in seattle with sql server and on windows nt?'
65. 'Can i find a job making more than 40000 a year without a degree?'
66. 'Can you offer anything with at least 60000 on a sun?'
67. 'Can you show me all the jobs?'
68. 'Can you show me vb jobs with 50000 salary with databases and excel?'
69. 'Could a senior consulting engineer find work in boston?'
70. 'Could i have some jobs using sql with oracle?'
71. 'Could you list all the jobs?'
72. 'Do any jobs exist programming for apple on pdp11s?'
73. 'Do you have any jobs involving 'c++' on aix?'
74. 'Does anyone still use mvs?'
75. 'Does apple have any software engineer positions?'
76. 'Does national instruments have any positions that dont require experience?'
77. 'Everything?'
78. 'Find all 'c++' jobs in austin?'
79. 'Find all network administration jobs in austin?'
80. 'Give me a list of all the jobs?'
81. 'Give me 'c++' jobs on windows nt?'
82. 'Give me jobs for a data warehousing specialist?'
83. 'Give me jobs for a games specialist?'
84. 'Give me jobs in cobol ii?'
85. 'Give me jobs in dallas?'
86. 'Give me jobs in san antonio using cobol?'
87. 'Give me jobs in san antonio?'
88. 'Give me jobs in usa?'
89. 'Give me jobs on the mac using perl?'
90. 'Give me jobs on vms using sql?'
91. 'Give me jobs requiring bs?'
92. 'Give me jobs that require ethernet experience but no html?'

93. 'Give me jobs using visual 'c++'?'
94. 'Give me the jobs for a database specialist in usa?'
95. 'Give me the jobs for a games specialist?'
96. 'Give me the jobs for a ole specialist?'
97. 'Give me the jobs in 'c++'?'
98. 'Give me the jobs in dallas?'
99. 'Give me the jobs in houston?'
100. 'Give me the jobs in visual 'c++'?'
101. 'Give me the jobs on novell?'
102. 'Give me the jobs on unix?'
103. 'Give me the jobs on vms in assembly?'
104. 'Give me the jobs on windows nt?'
105. 'Give me the jobs requiring bscs?'
106. 'Give me the jobs requiring bsee?'
107. 'Give me the jobs using c?'
108. 'Give me the jobs using 'c++' that dont require windows?'
109. 'Give me the jobs using cobol?'
110. 'Give me the jobs using sql?'
111. 'Give some jobs in dallas on a sun system?'
112. 'Greed for 80000 and java plagues developer wanting to live in san jose at apple?'
113. 'How much experience is wanted for a job at microsoft?'
114. 'I hold a degree in bscs in austin are there any jobs for me?'
115. 'I sure do wish there were java assembly jobs out there '." can you help?'
116. 'I sure would like a perl job at microsoft involving databases?'
117. 'I want a job that doesnt use windows?'
118. 'I want a job that use 'c++'?'
119. 'I wish there were some perl jobs in boston?'
120. 'I wonder what jpl does on unix with prolog and vax?'
121. 'I would like to find a job using java?'
122. 'I would like to see all the jobs?'
123. 'Id like to see everything?'
124. 'Id like to see the jobs in houston for a prolog programmer making at least 50000 a year involving databases?'
125. 'If i moved to california and learned sql on oracle could i find anything for 30000 on unix?'
126. 'Is anyone offering 40000 for ai work?'
127. 'Is fortran required for any jobs?'
128. 'Is there anything for an old cobol programmer on mvs?'
129. 'List all jobs using 'c++' and java in california?'
130. 'List jobs in austin?'
131. 'List jobs in client server?'
132. 'List jobs in cobol ii?'
133. 'List jobs in usa?'
134. 'List jobs in wan?'
135. 'List jobs on sun?'
136. 'List jobs on vms?'
137. 'List jobs on windows?'
138. 'List jobs requiring ba?'
139. 'List jobs requiring bscs using java?'
140. 'List jobs requiring bsee?'
141. 'List jobs using assembly?'
142. 'List jobs using cobol ii?'
143. 'List jobs using java?'
144. 'List jobs using sql?'
145. 'List the companies that desire 'c++' experience?'
146. 'List the jobs for a client server specialist?'
147. 'List the jobs for a database specialist?'

148. 'List the jobs in 'c++'?'
149. 'List the jobs in database?'
150. 'List the jobs in san antonio?'
151. 'List the jobs in visual 'c++'?'
152. 'List the jobs on unix?'
153. 'List the jobs requiring bs?'
154. 'List the jobs requiring java a bscs 2 years experience?'
155. 'List the jobs using assembly?'
156. 'List the jobs using cics?'
157. 'List the jobs using cobol ii?'
158. 'List the jobs using html for a games specialist?'
159. 'List the positions that require a knowledge of microsoft excel?'
160. 'List the required experience for a job using lisp?'
161. 'Moving to canada need a job with unix java and ibm?'
162. 'Mvs cobol and databases are the key to tivoli?'
163. 'Only microsoft vb windows nt and excel and 70000 dollars can satiate me?'
164. 'Prolog ai and lisp and graphics?'
165. 'Show a list of jobs requiring experience in 'c++' or java?'
166. 'Show all intern positions in texas with network and java?'
167. 'Show jobs for a com specialist?'
168. 'Show jobs for a data warehousing specialist?'
169. 'Show jobs for a shell programmer familiar with the unix environmen?'
170. 'Show jobs in austin that require a bscs?'
171. 'Show jobs in html?'
172. 'Show jobs in usa?'
173. 'Show jobs on windows?'
174. 'Show jobs requiring ba?'
175. 'Show jobs requiring bsee?'
176. 'Show jobs that are not in austin pay less than 10000 require knowledge of 'c++' pascal and java and desire a phd?'
177. 'Show jobs that do not require a degree for visual basic programmers?'
178. 'Show jobs using cics?'
179. 'Show jobs using powerbuilder?'
180. 'Show jobs using visual basic?'
181. 'Show me a dell job in austin requiring a bscs?'
182. 'Show me a job not requirng java and not in austin?'
183. 'Show me a job that requires 'c++' and java and is in austin?'
184. 'Show me all job that are available?'
185. 'Show me all of the software engineer jobs in austin?'
186. 'Show me all of the software qa jobs in austin?'
187. 'Show me austin jobs desiring a bscs?'
188. 'Show me austin jobs requiring a bscs degree with a salary greater than 50000 per year?'
189. 'Show me austin jobs requiring a bscs?'
190. 'Show me austin jobs with a salary of 50000?'
191. 'Show me 'c++' jobs requiring a bscs in austin?'
192. 'Show me dallas jobs requiring a bscs?'
193. 'Show me developer jobs requiring experience with mac?'
194. 'Show me everything?'
195. 'Show me graphics jobs which phil smith is recruiting for?'
196. 'Show me houston jobs using c in the specialty area of oil pipeline modeling?'
197. 'Show me houston jobs using 'c++' on pc?'
198. 'Show me jobs are dell requiring experience on unix?'
199. 'Show me jobs at dell earning 60000?'
200. 'Show me jobs at dell requiring a bscs degree?'
201. 'Show me jobs at dell requiring no experience and a bscs?'
202. 'Show me jobs desiring a ma in austin with microsoft?'

203. 'Show me jobs desiring a mscs in austin with microsoft?'
204. 'Show me jobs for dell requiring experience on unix?'
205. 'Show me jobs in austin that use java on unix for a developer paying at least 50000?'
206. 'Show me jobs in austin using solaris that do not require a bscs?'
207. 'Show me jobs in computer graphics requiring a ba in art and knowledge of 'speedy3dgraphics'?'
208. 'Show me jobs in dallas requiring knowledge of linux and pays more than 50000 a year?'
209. 'Show me jobs in texas?'
210. 'Show me jobs in tulsa using fortran on vax requiring a bscs?'
211. 'Show me jobs located in austin for 'c++' programmers?'
212. 'Show me jobs not involving 'c++'?'
213. 'Show me jobs paying greater than 50000 per year?'
214. 'Show me jobs requiring a bscs on sun?'
215. 'Show me jobs requiring a bscs on suns?'
216. 'Show me jobs requiring no experience?'
217. 'Show me jobs that require 3 years work experience in 'c++'?'
218. 'Show me jobs using lisp that require a bscs and desire a msee?'
219. 'Show me jobs with a salary greater than 50000 dollars a year?'
220. 'Show me jobs with the playstation in the specialty area of animation?'
221. 'Show me management jobs in boston requiring an mba and the knowledge of visual basic?'
222. 'Show me new york jobs requiring a bscs?'
223. 'Show me positions in web programming?'
224. 'Show me programmer jobs in tulsa?'
225. 'Show me programmer jobs requiring no experience on unix?'
226. 'Show me something that requires oracle?'
227. 'Show me systems analyst jobs at ibm?'
228. 'Show me systems analyst jobs at tivoli?'
229. 'Show me the 'c++' jobs in nashville that desire 2 years experience?'
230. 'Show me the hardware platforms associated with a netware administrator with ibm?'
231. 'Show me the job application for ic design engineer?'
232. 'Show me the jobs at companies in austin that want a degree in bscs?'
233. 'Show me the jobs concerning game developer on a playstation?'
234. 'Show me the jobs concerning games development on a playstation?'
235. 'Show me the jobs in austin that desire 3 years of experience and use 'c++'?'
236. 'Show me the jobs in texas using ai on unix?'
237. 'Show me the jobs requiring 3 years of experience at ibm?'
238. 'Show me the jobs requiring 3 years of experience at tivoli?'
239. 'Show me the jobs that are not in haskell?'
240. 'Show me the jobs that operate on sun?'
241. 'Show me the jobs that require 1 year of experience but desire 2 years of experiences?'
242. 'Show me the jobs that require 2 years experience?'
243. 'Show me the jobs using 'c++' that require a bscs but desire a mscs?'
244. 'Show me the jobs using java with salaries greater than 50000 per year?'
245. 'Show me the jobs using lisp requiring a bscs?'
246. 'Show me the jobs using perl with lockheed martin aeronautics in colorado?'
247. 'Show me the jobs which use excel?'
248. 'Show me the jobs with 30000 salary?'
249. 'Show me the jobs with a salary of 50000?'
250. 'Show me the networking jobs in houston with a salary of 50000?'
251. 'Show me the research assistant job in austin?'
252. 'Show me the senior development engineer jobs which require a master?'
253. 'Show me the senior software developer jobs which require a master?'
254. 'Show me the titles of the available jobs using prolog in houston?'
255. 'Show me web developer job opennings at trilogy?'
256. 'Show me what jobs there are?'
257. 'Show me what needs experience?'
258. 'Show me whats out there for perl developers on windows?'

259. 'Show the jobs for a odbc specialist?'
260. 'Show the jobs for bscs in austin?'
261. 'Show the jobs in austin?'
262. 'Show the jobs in client server?'
263. 'Show the jobs in dallas?'
264. 'Show the jobs in san antonio?'
265. 'Show the jobs in visual 'c++'?'
266. 'Show the jobs offering 40000 working with c on windows nt?'
267. 'Show the jobs on mvs?'
268. 'Show the jobs on pc requiring bscs?'
269. 'Show the jobs on pc?'
270. 'Show the jobs on sun?'
271. 'Show the jobs on vms?'
272. 'Show the jobs requiring ba?'
273. 'Show the jobs requiring bs in usa?'
274. 'Show the jobs requiring bsee?'
275. 'Show the jobs using html?'
276. 'Show the jobs using lisp not requiring a degree in cs?'
277. 'Show the jobs using powerbuilder?'
278. 'Show the jobs with the title systems analyst requiring 2 years of experience?'
279. 'Tell me jobs for a device driver specialist?'
280. 'Tell me jobs for a mfc specialist?'
281. 'Tell me jobs in austin?'
282. 'Tell me jobs in networking?'
283. 'Tell me jobs on sun?'
284. 'Tell me jobs on windows 95 in mfc?'
285. 'Tell me jobs requiring ba?'
286. 'Tell me jobs requiring bscs?'
287. 'Tell me jobs using cobol ii?'
288. 'Tell me jobs using html?'
289. 'Tell me jobs using visual basic?'
290. 'Tell me the jobs in '3d' graphics?'
291. 'Tell me the jobs in lan?'
292. 'Tell me the jobs in usa?'
293. 'Tell me the jobs on mvs?'
294. 'Tell me the jobs on windows nt?'
295. 'Tell me the jobs requiring ba using cobol?'
296. 'Tell me the jobs requiring bs in usa?'
297. 'Tell me the jobs requiring bs?'
298. 'Tell me the jobs using powerbuilder?'
299. 'Tell me the jobs using 'vc++'?'
300. 'Tell me the jobs using visual basic?'
301. 'Tell me what jobs there are?'
302. 'Test engineer in need of 40000 in seattle on windows nt?'
303. 'There must be some jobs out there for a 'c++' programmer that thinks in unix databases?'
304. 'Vanity wants 5000 a month with buzwords like java apple internet and california?'
305. 'What ai jobs are there in texas that pay 65000?'
306. 'What ai positions require only a bscs?'
307. 'What are all the jobs?'
308. 'What are the 'c++' jobs in austin requiring a bscs?'
309. 'What are the degree requirements for a software engineer?'
310. 'What are the jobs for a 'c++' programmer in austin?'
311. 'What are the jobs for programmer in austin that has salary 50000 that uses 'c++' and not related with ai?'
312. 'What are the jobs in austin requiring knowledge of oracle?'
313. 'What are the jobs in washington that require at least 5 years of experience?'

314. 'What are the jobs that pay 50000 per year?'
315. 'What are the jobs that require experience with aix but not windows nt?'
316. 'What are the jobs that require experience with microsoft word?'
317. 'What are the jobs using 'c++' with salaries of 50000?'
318. 'What are the positions within dell that requires bscs?'
319. 'What are the positions within hp that pay 40000 per year?'
320. 'What are the software engineer jobs available using ada?'
321. 'What are the software engineering jobs available using ada?'
322. 'What austin area web jobs require java and 'c++'?'
323. 'What austin jobs that use cobol do not require any experience?'
324. 'What 'c++' jobs are in austin?'
325. 'What can i find using java on unix?'
326. 'What database jobs are there?'
327. 'What developer jobs in austin require a bscs and 'c++'?'
328. 'What do you have paying over 40000 on the vax?'
329. 'What engineer positions in telecommunications companies in dallas do not require 'c++'?'
330. 'What ibm jobs require using java on commodores?'
331. 'What is out there?'
332. 'What java jobs are there with ibm in austin?'
333. 'What job is there for a bscs with 5 years of experience?'
334. 'What job is there for 'c++' but not visual 'c++'?'
335. 'What jobs are available for a solaris systems administrator?'
336. 'What jobs are available for someone who knows oracle on solaris?'
337. 'What jobs are available that require java but not internet experience or ai experience?'
338. 'What jobs are available using apache with a specialty area of networking?'
339. 'What jobs are available?'
340. 'What jobs are in seattle that are not at microsoft?'
341. 'What jobs are longhorn employment hiring for?'
342. 'What jobs are there doing computer graphics on silicon graphics machines?'
343. 'What jobs are there for a '3d' graphics specialist?'
344. 'What jobs are there for a com specialist?'
345. 'What jobs are there for a graphics specialist?'
346. 'What jobs are there for a gui specialist?'
347. 'What jobs are there for a networking specialist?'
348. 'What jobs are there for a test engineer using java?'
349. 'What jobs are there for a visual basic developer?'
350. 'What jobs are there for assembly programer that require a bscs?'
351. 'What jobs are there for austin mac programmer using 'c++'?'
352. 'What jobs are there for 'c++' programmers which pay more than 60000 per year?'
353. 'What jobs are there for 'c++' unix developer?'
354. 'What jobs are there for pascal programers who dont know 'c++'?'
355. 'What jobs are there for programmers that know assembly?'
356. 'What jobs are there for programmers who know java?'
357. 'What jobs are there for web developer who know 'c++'?'
358. 'What jobs are there for windows nt developers that know oracle?'
359. 'What jobs are there in austin for people with knowledge of the application oracle?'
360. 'What jobs are there in austin for project manager area games on mac using pascal?'
361. 'What jobs are there in austin requiring a phd?'
362. 'What jobs are there in austin that require 5 years experience?'
363. 'What jobs are there in austin that require a bscs degree?'
364. 'What jobs are there in austin that requires experience with unix?'
365. 'What jobs are there in austin with a salary of at least 100000 per year?'
366. 'What jobs are there in dallas that requires a mscs?'
367. 'What jobs are there in data warehousing?'
368. 'What jobs are there in games?'
369. 'What jobs are there in houston?'

370. 'What jobs are there in odbc?'
371. 'What jobs are there in sql?'
372. 'What jobs are there in texas that use java and require no experience?'
373. 'What jobs are there in usa on ibm?'
374. 'What jobs are there in usa?'
375. 'What jobs are there on aix?'
376. 'What jobs are there on ibm?'
377. 'What jobs are there on novell involving the internet?'
378. 'What jobs are there on vms?'
379. 'What jobs are there on windows 95?'
380. 'What jobs are there on windows nt?'
381. 'What jobs are there on x86?'
382. 'What jobs are there on x86?'
383. 'What jobs are there outside austin which pay less than 60000 per year?'
384. 'What jobs are there requiring ba?'
385. 'What jobs are there requiring bs?'
386. 'What jobs are there requiring bscs?'
387. 'What jobs are there requiring bsee?'
388. 'What jobs are there that dont require a degree but use perl?'
389. 'What jobs are there using cics?'
390. 'What jobs are there using cobol ii?'
391. 'What jobs are there using cobol?'
392. 'What jobs are there using rpg?'
393. 'What jobs are there using sql?'
394. 'What jobs are there using 'tcl/tk'?'
395. 'What jobs are there using 'vc++'?'
396. 'What jobs are there using visual basic?'
397. 'What jobs are there which require java on windows and unix?'
398. 'What jobs are there with a salary of 40000?'
399. 'What jobs are there with a salary of more than 50000 dollars per year?'
400. 'What jobs are there working for microsoft programming lisp for autocad?'
401. 'What jobs are there?'
402. 'What jobs as a senior software developer are available in houston but not san antonio?'
403. 'What jobs as an sql engineer pay 100000?'
404. 'What jobs as manufacturing manager pay 100000?'
406. 'What jobs at dell require a bscs?'
407. 'What jobs can a delphi developer find in san antonio on windows?'
408. 'What jobs can i find with tivoli?'
409. 'What jobs desire 2 years of experience with powerbuilder on windows nt?'
410. 'What jobs desire a degree but dont use 'c++'?'
411. 'What jobs do not require a degree but pay more than 60000?'
412. 'What jobs do you have?'
413. 'What jobs does lcs recruit for?'
414. 'What jobs does microsoft recruit for?'
415. 'What jobs give me 40000 to work in houston on internet and web with perl?'
416. 'What jobs have a recruiter named phil smith?'
417. 'What jobs have a salary greater than 20 and hour?'
418. 'What jobs in austin are for a lisp programmer that involve unix and the internet?'
419. 'What jobs in austin are there that pay at least 100000 per year?'
420. 'What jobs in austin desiring a bscs are there for a 'c++' programmer?'
421. 'What jobs in austin have a salary of 60000?'
422. 'What jobs in austin need knowledge in unix?'
423. 'What jobs in austin only require a bscs and no experience?'
424. 'What jobs in austin or dallas desire a degree?'
425. 'What jobs in austin require 10 years of experience?'
426. 'What jobs in austin require 5 years of experience but desire 10 years of experience?'

427. 'What jobs in austin require a bscs degree and deal with 'tcp/ip'?'
428. 'What jobs in austin require 'c++' and unix?'
429. 'What jobs in austin require knowledge of the platform unix?'
430. 'What jobs in austin require no experience?'
431. 'What jobs in austin use 'c++' and java?'
432. 'What jobs in boston have openings for a 'c++' programmer?'
433. 'What jobs in california pay 60000 for sql development?'
434. 'What jobs in california require java and internet experience?'
435. 'What jobs in dallas require a bscs and 'c++' but not java?'
436. 'What jobs in dallas require experience with unix?'
437. 'What jobs in houston are there that requires a bscs with 1 year of experience?'
438. 'What jobs in houston require a bacs?'
439. 'What jobs in ibm in austin do not need a degree?'
440. 'What jobs in san antonio require the use of cobol?'
441. 'What jobs in san jose offer a java programmer for 40000 a year?'
442. 'What jobs need at least 2 years of experience?'
443. 'What jobs need knowledge of 'c++' or java?'
444. 'What jobs on pc are for programming assembly and desire 5 years experience?'
445. 'What jobs pay 40000 per year that require a bscs?'
446. 'What jobs pay 40000?'
447. 'What jobs pay 60000 are located in austin and require a bscs?'
448. 'What jobs pay 60000 are located in austin and require a degree?'
450. 'What jobs pay at least 80000 dollars per year?'
451. 'What jobs require 10 years of experience require a phd are in cobol and are located in texas?'
452. 'What jobs require 10 years of experience require a phd in cs are in cobol and are located in texas?'
453. 'What jobs require a bscs 4 years of experience pay 50000 and are in san jose?'
454. 'What jobs require a bscs and experience with oracle?'
455. 'What jobs require a bscs and no experience?'
456. 'What jobs require a bscs degree and desire an mscs degree?'
457. 'What jobs require a bscs?'
458. 'What jobs require a degree for pascal programmers who do not know 'c++'?'
459. 'What jobs require a msee and pays more than 100000 per year?'
460. 'What jobs require at least 1 year of experience in 'c++'?'
461. 'What jobs require 'c++' and pays a salary greater than 90000 per year?'
462. 'What jobs require experience in 'c++' and java but not perl?'
463. 'What jobs require knowledge of 'c++' but not perl?'
464. 'What jobs use 'c++' on mac and pay 70000?'
465. 'What jobs use 'c++' with the web on macs?'
466. 'What jobs use cobol on ibm machines and pay 70000?'
467. 'What jobs use html but do not require a degree?'
468. 'What jobs using fortran are there in houston?'
469. 'What jobs using fortran are there in los alamos?'
470. 'What jobs using java and perl are available in dallas and pay 50000 a year?'
471. 'What kind of jobs could i find for an old cobol programmer?'
472. 'What kinds of jobs are available for visual basic consultants in boston?'
473. 'What level of experience does ibm desire?'
474. 'What locations offer jobs using java on sun?'
476. 'What microsoft jobs do not require a bscs?'
477. 'What oracle jobs are there with compaq in houston using pc?'
478. 'What oracle jobs are there with compaq in houston using pcs?'
479. 'What position in microsoft do i need a phd to work?'
480. 'What positions are there in networking?'
481. 'What positions are there that use 'c++' and java?'
482. 'What programmer positions in austin require no experience?'
483. 'What programming jobs are there in austin that uses java?'
484. 'What programming languages are desired for a job as a programmer at ibm?'

485. 'What project manager jobs are there that require experience?'
486. 'What software engineer jobs are there that use 'c++'?'
487. 'What system administrator jobs are available from dell?'
488. 'What systems analyst jobs are there in austin?'
489. 'What tivoli jobs are there that require a bscs degree?'
490. 'What web developer jobs are there in austin?'
491. 'What web jobs are available that need mac experience and no degree?'
492. 'What web related jobs require a bscs but no experience?'
493. 'What work do you have available?'
494. 'Whats all there?'
495. 'Whats available on vax and near austin?'
496. 'Whats in dallas that pays over 60000 on linux with graphics and java?'
497. 'Where can i work with a bscs and no experience?'
498. 'Which jobs are for bsee majors with at least 5 years experience in windows nt?'
499. 'Which jobs at trilogy deal with 'c++'?'
500. 'Which jobs in austin offer for students fresh out of college in networking?'
501. 'Which jobs in houston offer over 50000 in graphics?'
502. 'Which jobs offer me 40000 to work on internet and web with perl?'
503. 'Which jobs pay 60000 that do not require a phd?'
504. 'Which jobs require c and 'c++' but not java?'
505. 'Which jobs require knowledge of lisp but dont specialize in ai?'
506. 'Which jobs use visual 'j++' as their development tool?'
507. 'Which system administrator jobs in dallas require 2 years' experience and pay 50000?'
508. 'Who gives 50000 for fortran?'
509. 'Who might offer me 50000 for web development?'

RESTAURANT QUERIES SET

Segment of Restaurants Search Query Set (Recreated from Original set by Tang & Mooney, 2001)

1. 'Give me a good american restaurant on fairgrounds dr in sunnyvale?'
2. 'Give me a good bakery in aptos?'
3. 'Give me a good bakery in berkeley?'
4. 'Give me a good bakery in bethel island?'
5. 'Give me a good bakery on appleton dr in aptos?'
6. 'Give me a good bakery on bethel island rd in bethel island?'
7. 'Give me a good bakery on shattuck ave in berkeley?'
8. 'Give me a good chinese restaurant in the bay area?'
9. 'Give me a good chinese restaurant on buchanan in san francisco?'
10. 'Give me a good french restaurant in alameda?'
11. 'Give me a good italian restaurant in the yosemite and mono lake area?'
12. 'Give me a good place in san francisco for french food?'
13. 'Give me a good place in the bay area for french food?'
14. 'Give me a good place on buchanan in san francisco for arabic food?'
15. 'Give me a good restaurant in alameda?'
16. 'Give me a good restaurant in san francisco for french food?'
17. 'Give me a good restaurant in the bay area for french food?'
18. 'Give me a good restaurant in the bay area?'
19. 'Give me a good restaurant on el camino in palo alto?'
20. 'Give me a good restaurant on soquel dr in aptos for french food?'
21. 'Give me a restaurant in alameda?'
22. 'Give me a restaurant in aptos that serves good french food?'
23. 'Give me a restaurant in san francisco that serves good chinese food?'
24. 'Give me a restaurant in sunnyvale that serves good american food?'
25. 'Give me a restaurant in the bay area?'
26. 'Give me a restaurant on buchanan in san francisco that serves good arabic food?'
27. 'Give me a restaurant on el camino in palo alto?'
28. 'Give me a restaurant on fairgrounds dr in sunnyvale that serves good american food?'

29. 'Give me a restaurant on soquel dr in aptos that serves good french food?'
30. 'Give me some good arabic restaurants in mountain view?'
31. 'Give me some good cafes in alameda?'
32. 'Give me some good cafes on webster st in alameda?'
33. 'Give me some good places for ice cream in alameda?'
34. 'Give me some good places for ice cream in alameda?'
35. 'Give me some good places for ice cream on blanding ave in alameda?'
36. 'Give me some good places for ice cream on blanding ave in alameda?'
37. 'Give me some good places for pizza in alameda?'
38. 'Give me some good places for pizza in alameda?'
39. 'Give me some good places for pizza on el camino in palo alto?'
40. 'Give me some good places for pizza on el camino in palo alto?'
41. 'Give me some good places for pizza on webster st in alameda?'
42. 'Give me some good places for pizza on webster st in alameda?'
43. 'Give me some good places on fairgrounds dr in sunnyvale for american food?'
44. 'Give me some good places on soquel dr in aptos for french food?'
45. 'Give me some good restaurants in alameda?'
46. 'Give me some good restaurants in mountain view?'
47. 'Give me some good restaurants in the bay area?'
48. 'Give me some good restaurants on bethel island rd in bethel island?'
49. 'Give me some good restaurants on blanding ave in alameda?'
50. 'Give me some good restaurants on buchanan in san francisco for chinese food?'
51. 'Give me some good restaurants on el camino in palo alto?'
52. 'Give me some good restaurants on fairgrounds dr in sunnyvale for american food?'
53. 'Give me some restaurants good for arabic food in mountain view?'
54. 'Give me some restaurants good for arabic food in the bay area?'
55. 'Give me some restaurants good for arabic food on buchanan in san francisco?'
56. 'Give me some restaurants good for arabic food?'
57. 'Give me some restaurants good for french food in the yosemite and mono lake area?'
58. 'Give me some restaurants good for french food on fairgrounds dr in sunnyvale?'
59. 'Give me some restaurants good for french food?'
60. 'Give me some restaurants good for italian food in alameda?'
61. 'Give me some restaurants in alameda?'
62. 'Give me some restaurants in mountain view?'
63. 'Give me some restaurants in the bay area?'
64. 'Give me some restaurants on bethel island rd in bethel island?'
65. 'Give me some restaurants on blanding ave in alameda?'
66. 'Give me some restaurants on el camino in palo alto?'
67. 'Give me the best bakery in fremont?'
68. 'Give me the best bakery in palo alto?'
69. 'Give me the best bakery in the bay area?'
70. 'Give me the best bakery in the bay area?'
71. 'Give me the best french restaurant in san francisco?'
72. 'Give me the best french restaurant in sunnyvale?'
73. 'Give me the best french restaurant in the bay area?'
74. 'Give me the best french restaurant in the bay area?'
75. 'Give me the best place in alameda for french food?'
76. 'Give me the best restaurant in fremont for american food?'
77. 'Give me the best restaurant in fremont for chinese food?'
78. 'Give me the best restaurant in monterey for french food?'
79. 'Give me the best restaurant in monterey for french food?'
80. 'Give me the best restaurant in palo alto for chinese food?'
81. 'Give me the best restaurant in palo alto for italian food?'
82. 'Give me the best restaurant in san jose for american food?'
83. 'Give me the best restaurant in san jose for french food?'
84. 'Give me the best restaurant in sunnyvale for french food?'

86. 'Give me the best restaurant in the bay area for american food?'
87. 'Give me the best restaurant in the bay area for chinese food?'
88. 'Give me the best restaurant in the bay area for chinese food?'
89. 'How many bakery are there in the bay area?'
90. 'How many buttercup kitchen are there in san francisco?'
91. 'How many buttercup kitchen are there in walnut creek?'
92. 'How many chinese restaurant are there in palo alto?'
93. 'How many chinese restaurant are there in san jose?'
94. 'How many chinese restaurant are there in the bay area?'
95. 'How many chinese restaurants are there in palo alto?'
96. 'How many chinese restaurants are there in the bay area?'
97. 'How many 'denny''s' are there in fremont?'
98. 'How many 'denny''s' are there in monterey county?'
99. 'How many 'denny''s' are there in palo alto?'
100. 'How many 'denny''s' are there in san francisco?'
101. 'How many 'denny''s' are there in san mateo county?'
102. 'How many 'denny''s' are there in sunnyvale?'
103. 'How many 'denny''s' are there in the bay area?'
104. 'How many french restaurant are there in palo alto?'
105. 'How many french restaurant are there in the bay area?'
106. 'How many french restaurants are in the santa clara county?'
107. 'How many french restaurants are in the yolo county?'
108. 'How many french restaurants are there in fremont?'
109. 'How many french restaurants are there in the bay area?'
110. 'How many italian restaurant are there in san jose?'
111. 'How many italian restaurant are there in the bay area?'
112. 'How many italian restaurants are in the santa clara county?'
113. 'How many italian restaurants are in the yolo county?'
114. 'How many italian restaurants are there in san francisco?'
115. 'How many italian restaurants are there in the bay area?'
116. 'How many jamerican cuisine are there in san francisco?'
117. 'How many jamerican cuisine are there in santa cruz county?'
118. 'How many jamerican cuisine are there in sunnyvale?'
119. 'How many places for chinese food are there in the bay area?'
120. 'How many places for chinese food are there in the bay area?'
121. 'How many places for french food are there in palo alto?'
122. 'How many places for french food are there in the bay area?'
124. 'How many places for ice cream are there in fremont?'
125. 'How many places for ice cream are there in the bay area?'
126. 'How many places for italian food are there in the bay area?'
127. 'How many wendys are there in the bay area?'
128. 'Show me a good italian restaurant in palo alto?'
129. 'What are some good places for ice cream in alameda?'
130. 'What are some good places for ice cream on blanding ave in alameda?'
131. 'What are some good places for pizza in alameda?'
133. 'What are some good places for pizza on el camino in palo alto?'
134. 'What are some good places for pizza on webster st in alameda?'
135. 'What are some good places in mountain view for chinese food?'
136. 'What are some good places in the bay area for chinese food?'
137. 'What are some good restaurants in alameda?'
138. 'What are some good restaurants in mountain view for arabic food?'
139. 'What are some good restaurants in mountain view?'
140. 'What are some good restaurants in the bay area for chinese food?'
141. 'What are some good restaurants in the bay area?'
142. 'What are some good restaurants on bethel island rd in bethel island?'
143. 'What are some good restaurants on blanding ave in alameda?'

144. 'What are some good restaurants on el camino in palo alto?'
145. 'What is a good restaurant in alameda?'
146. 'What is a good restaurant in the bay area?'
147. 'What is a good restaurant on el camino in palo alto?'
148. 'What is the best bakery in fremont?'
149. 'What is the best bakery in the bay area?'
150. 'What is the best french restaurant in san francisco?'
151. 'What is the best french restaurant in the bay area?'
152. 'What is the best place in alameda for french food?'
153. 'What is the best restaurant in fremont for american food?'
154. 'What is the best restaurant in monterey for french food?'
155. 'What is the best restaurant in palo alto for chinese food?'
156. 'What is the best restaurant in palo alto for italian food?'
157. 'What is the best restaurant in san jose for french food?'
158. 'What is the best restaurant in the bay area for american food?'
159. 'What is the best restaurant in the bay area for chinese food?'
160. 'Where are some good cafes in alameda?'
161. 'Where are some good cafes on webster st in alameda?'
162. 'Where are some good chinese restaurants in mountain view?'
163. 'Where are some good places for ice cream in alameda?'
164. 'Where are some good places for ice cream on blanding ave in alameda?'
165. 'Where are some good places for pizza in alameda?'
166. 'Where are some good places for pizza on webster st in alameda?'
167. 'Where are some restaurants good for arabic food in mountain view?'
168. 'Where are some restaurants good for arabic food in the bay area?'
169. 'Where are some restaurants good for arabic food on buchanan in san francisco?'
170. 'Where are some restaurants good for arabic food?'
171. 'Where are some restaurants good for french food in alameda?'
172. 'Where are some restaurants good for french food in the yosemite and mono lake area?'
173. 'Where are some restaurants good for french food?'
174. 'Where are some restaurants good for italian food on fairgrounds dr in sunnyvale?'
175. 'Where can i eat american food on fairgrounds dr in sunnyvale?'
176. 'Where can i eat arabic food in alameda?'
177. 'Where can i eat arabic food on buchanan in san francisco?'
178. 'Where can i eat chinese food in the bay area?'
179. 'Where can i eat french food in mountain view?'
180. 'Where can i eat french food in the bay area?'
181. 'Where can i eat french food on buchanan in san francisco?'
182. 'Where can i eat italian food in san francisco?'
183. 'Where can i eat italian food in the bay area?'
184. 'Where can i eat some good american food on fairgrounds dr in sunnyvale?'
185. 'Where can i eat some good arabic food in alameda?'
186. 'Where can i eat some good arabic food in the bay area?'
187. 'Where can i eat some good chinese food on buchanan in san francisco?'
188. 'Where can i eat some good french food in mountain view?'
189. 'Where can i eat some good french food in the bay area?'
190. 'Where can i eat some good french food on fairgrounds dr in sunnyvale?'
191. 'Where can i eat some good italian food in san francisco?'
192. 'Where can i eat some good italian food in the bay area?'
193. 'Where can i find a 'denny''s' in san francisco?'
194. 'Where can i find a jamerican cuisine in san francisco?'
195. 'Where can i find a restaurant in the bay area?'
196. 'Where can we find a restaurant in alameda?'
197. 'Where can we find a restaurant on el camino in palo alto?'
198. 'Where can we find some restaurants in alameda?'
199. 'Where can we find some restaurants in mountain view?'

200. 'Where can we find some restaurants in the bay area?'
201. 'Where can we find some restaurants on bethel island rd in bethel island?'
202. 'Where can we find some restaurants on blanding ave in alameda?'
203. 'Where can we find some restaurants on el camino in palo alto?'
204. 'Where is a 'denny''s' in san francisco?'
205. 'Where is a french restaurant on bethel island rd in bethel island?'
206. 'Where is a good american restaurant on fairgrounds dr in sunnyvale?'
207. 'Where is a good arabic restaurant in the bay area?'
208. 'Where is a good arabic restaurant on buchanan in san francisco?'
209. 'Where is a good bakery in aptos?'
210. 'Where is a good bakery in berkeley?'
211. 'Where is a good bakery in bethel island?'
212. 'Where is a good bakery on appleton dr in aptos?'
213. 'Where is a good bakery on bethel island rd in bethel island?'
214. 'Where is a good bakery on shattuck ave in berkeley?'
215. 'Where is a good french restaurant in alameda?'
216. 'Where is a good place in alameda for arabic food?'
217. 'Where is a good place in the bay area for chinese food?'
218. 'Where is a good place on buchanan in san francisco for chinese food?'
219. 'Where is a good place on fairgrounds dr in sunnyvale for american food?'
220. 'Where is a good place on soquel dr in aptos for french food?'
221. 'Where is a good restaurant in alameda for chinese food?'
222. 'Where is a good restaurant in the bay area for arabic food?'
223. 'Where is a good restaurant on buchanan in san francisco for arabic food?'
224. 'Where is a good restaurant on fairgrounds dr in sunnyvale for american food?'
225. 'Where is a good restaurant on soquel dr in aptos for french food?'
226. 'Where is a italian restaurant on el camino in palo alto?'
227. 'Where is a jamerican cuisine in san francisco?'
228. 'Where is a restaurant in alameda?'
229. 'Where is a restaurant in aptos that serves good french food?'
230. 'Where is a restaurant in san francisco that serves good chinese food?'
231. 'Where is a restaurant in sunnyvale that serves good american food?'
232. 'Where is a restaurant on buchanan in san francisco that serves good chinese food?'
233. 'Where is a restaurant on fairgrounds dr in sunnyvale that serves good american food?'
234. 'Where is a restaurant on soquel dr in aptos that serves good french food?'
235. 'Where is buttercup kitchen?'
236. 'Where is 'denny''s' in san francisco?'
237. 'Where is 'denny''s' in the bay area?'
238. 'Where is 'denny''s'?'
239. 'Where is jamerican cuisine in san francisco?'
240. 'Where is jamerican cuisine in the bay area?'
241. 'Where is jamerican cuisine?'
242. 'Where is the best bakery in palo alto?'
243. 'Where is the best bakery in the bay area?'
244. 'Where is the best french restaurant in sunnyvale?'
245. 'Where is the best french restaurant in the bay area?'
246. 'Where is the best restaurant in fremont for chinese food?'
247. 'Where is the best restaurant in monterey for french food?'
248. 'Where is the best restaurant in san jose for american food?'
249. 'Where is the best restaurant in sunnyvale for french food?'
250. 'Where is the best restaurant in the bay area for american food?'
251. 'Where is the best restaurant in the bay area for arabic food?'

Appendix 2: Illustrative Examples Of Kernelization Process

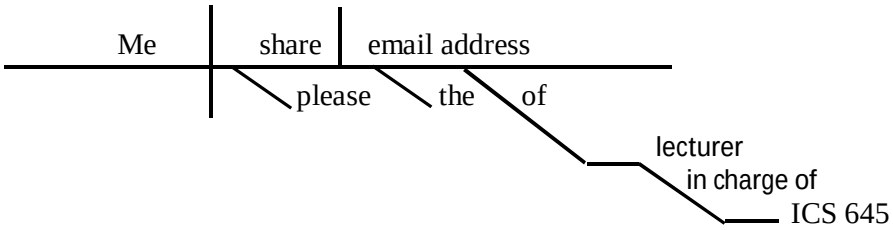


Fig. App1: Please share with me email address of the lecturer in charge of ICS 645

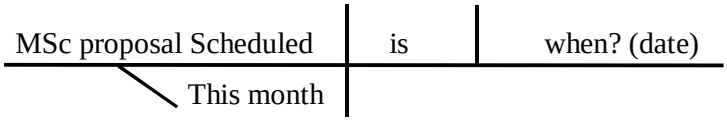


Fig. App2: When is this month's MSc proposal presentation scheduled?

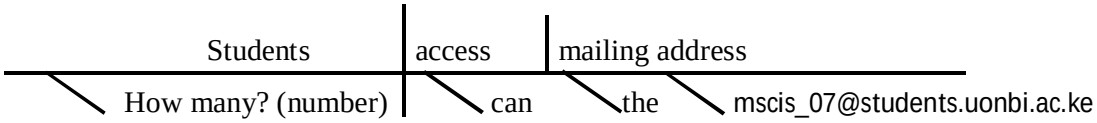


Fig. App3: How many students can access the mailing address mscis_07@students.uonbi.ac.ke

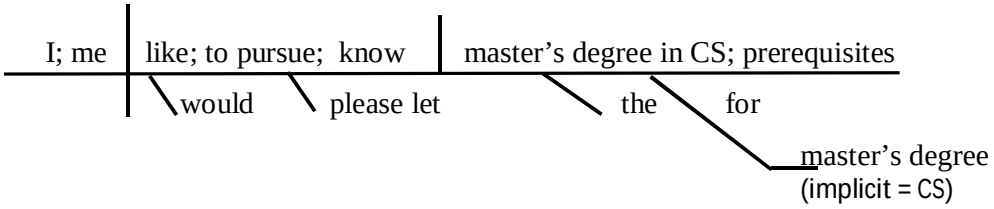


Fig. App4: I would like to pursue a master's degree in CS, please let me know the prerequisites for course

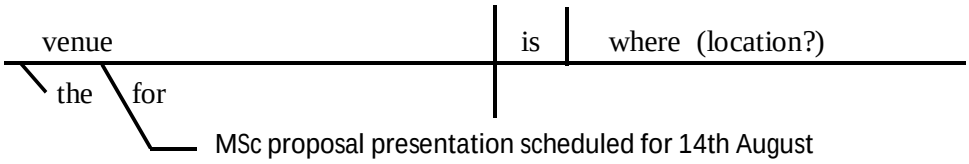


Fig. App5: Where is the venue for MSc proposal presentation scheduled for 14th August

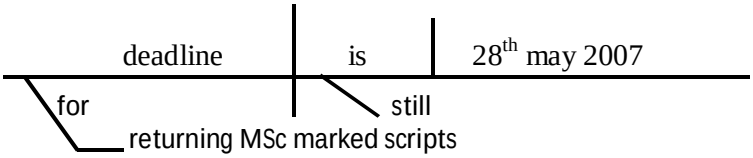


Fig. App6: Is deadline for returning MSc marked scripts still 28th may 2007?

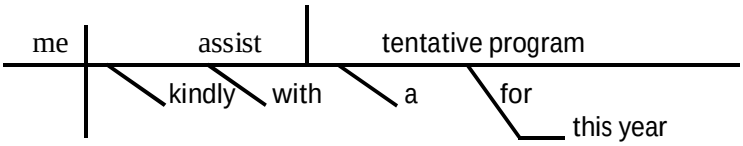


Fig. App7: Kindly assist me with a tentative program for this year

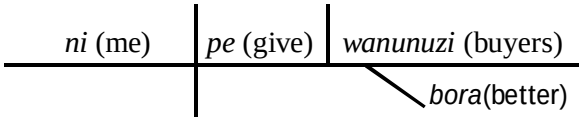


Fig. App8: [Nipe] wanunuzi bora ([Give me] better buyers)

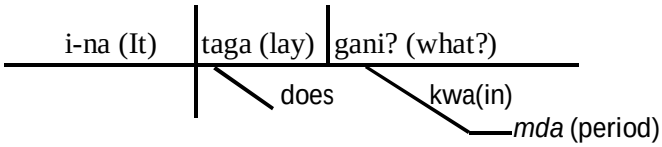


Fig. App9: i-na-taga (In what period it lays)

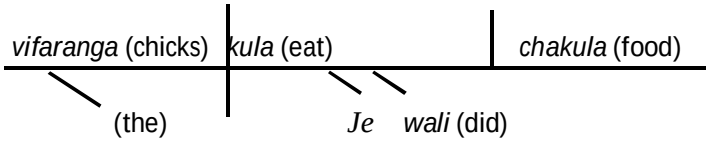


Fig. App10: *'Je, vifaranga walikula chakula?* (Did the chicks eat food)

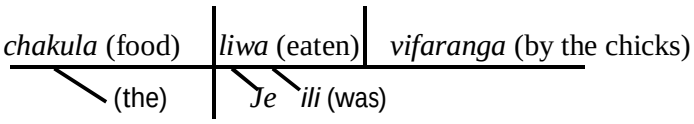


Fig. App11:*'Je, chakula ililiwa na vifaranga?'* (Was the food eaten by the chicks?)

Appendix 3: Survey on Database Schema Authorship

QUESTIONNAIRE ON NOMENCLATURE OF DB SCHEMA ATTRIBUTES

This questionnaire is administered by Mr. Lawrence Muchemi a PhD student at University of Nairobi. The purpose of this survey is to identify if there exists any common practice naming procedures or policies for database-names, tables-names' and column-names'. The survey will lead to a better understanding of database schema authorship characteristics.

1. In what position do you serve in your organization? …….

2. What have you been actively involved in? (tick all applicable)
   a. Database design …….. Yes ☐ No ☐
   b. Database Development …….. Yes ☐ No ☐
   c. Database administration …….. Yes ☐ No ☐
   d. Web designer/developer/ admin…….. Yes ☐ No ☐
   e. Other …….

3. List the **names of some databases** you have worked with
   a. …… c. …… e. ……
   b. …… d. …… f. ……

4. List the **names of some tables** you have worked with in any of the databases named 3 above
   a. …… e. …… i. ……
   b. …… f. …… j. ……
   c. …… g. …… k. ……
   d. …… h. …… l. ……

5. List the **some names of columns** you have worked with in any of the tables named in 4 above
   …..... …… …… ……..
   …..... …… …… ……..
   …..... …… …… ……..

6. Does your organization have a policy on naming procedure for databases, table names or column names (It can be formal/non-formal or documented/ undocumented).

Yes ☐ No ☐ If yes provide some detail ……

7. Do you personally have a common practice that you use in naming tables and columns

Yes ☐ No. ☐ If yes provide some details ……..……

8.

Please fill in this table the extent of usage of each of the indicated schema objects' naming styles (ie table names, column names, database names etc). Use a scale of 1 to 5 where 5 indicates the mostly used style while 1 indicates the least. Indicate a 0 where you have not used that style.

	Pattern	Extent of usage (scale of 1-5; 5=most used, 1= least used; 0= never used)	Comments
1	Under_score		
2	camelCase		
3	Da-sh		
4	Abbreviations emp for employe		
5	Pascal Casing		
6	Finger_Breaking_Underscore		
7	SCREAMING_UNDERSCORE		
8	Acronyms eg ID, UI, IO		
9	Dot eg hr.hire_date		
10	"string like this"		

8. Do you know **somebody in a different organization** who can help in giving similar information as required above? Please recommend someone and give contact information.
……

Appendix 4: Training the Chunk Parser and Evaluating its Performance

```
>>> from nltk.corpus import conll2000 // load the CoNll corpus

>>> test_sents = conll2000.chunked_sents('test.txt', chunk_types=['NP'])// define testing set and types of phrases

>>> train_sents = conll2000.chunked_sents('train.txt', chunk_types=['NP'])// define training set and types of phrases

>>> class ChunkParser(nltk.ChunkParserI): // Defining chunk parser class

    def __init__(self, train_sents):

     train_data = [[(t,c) for w,t,c in nltk.chunk.tree2conlltags(sent)]

          for sent in train_sents]

     self.tagger = nltk.TrigramTagger(train_data)

    def parse(self, sentence):

     pos_tags = [pos for (word,pos) in sentence]

     tagged_pos_tags = self.tagger.tag(pos_tags)

     chunktags = [chunktag for (pos, chunktag) in tagged_pos_tags]

     conlltags = [(word, pos, chunktag) for ((word,pos),chunktag)

          in zip(sentence, chunktags)]

     return nltk.chunk.conlltags2tree(conlltags)

>>> NPChunker = ChunkParser(train_sents)// Training the chunker

>>> print NPChunker.evaluate(test_sents)// Evaluating performance

ChunkParse score:

   IOB Accuracy:  93.3%

   Precision:     82.5%

   Recall:        86.8%

   F-Measure:     84.6%

>>>
```

Appendix 5: Concept Templates used

NB: Concept Patterns = Nouns + Noun phrases + Verb phrases+ Term Collocations)

A: Regular Patterns of Noun-phrases (reported by Ohly, (1982) and recast by Sewangi, (2001))

- nominalized verb phrase (VN N) for example 'kukaza uzi' (stretching thread),
- deverbative head with a noun complement (DV N) for example 'kiweka damu',
- two nouns (N N) for example 'haidrogeni peroksaidi',
- combination of noun and adjective (N Adj),
- noun construction with a connecter -a (N -a N) for example 'rangi za moto'
- and constructions with connecter -a followed by a verb noun qualifier (N -a VN) for example sindano ya kutungia' (boring needle).

A: Patterns Of Kiswahili Common Multi-Word Terms' (Term Collocations, Noun-Phrases and Verb Phrases) Sewangi, (2001)

PATTERNS OF KISWAHILI TERM (PHRASES) COLLOCATIONS

1. DOMAIN-N + A-INFL "2"
2. DOMAIN-N + GEN-CON + ADV "2"
3. DOMAIN-N + GEN-CON + DOMAIN-N "2"
4. DOMAIN-N + GEN-CON + DOMAIN-V "2"
5. DOMAIN-N + GEN-CON + N "2"
6. DOMAIN-N + DOMAIN-N "2"
7. DOMAIN-N + INF "2"
8. DOMAIN-N + N "2"
9. DOMAIN-N + POSS + N "2"
10. DOMAIN-N + PREP + DOMAIN-N "2"
11. DOMAIN-V + ADV "2"
12. DOMAIN-V + GEN-CON + DOMAIN-N "2"
13. DOMAIN-V + GEN-CON + N "2"
14. DOMAIN-V + DOMAIN-N "2"
15. DOMAIN-V + N "2"
16. INF + GEN-CON + DOMAIN-N "2"
17. INF + DOMAIN-N "2"
18. N + GEN-CON + DOMAIN-N "2"
19. N + GEN-CON + DOMAIN-V "2"
20. N + DOMAIN-N "2"
21. DOMAIN-N + A-INFL + GEN-CON + N "3"
22. DOMAIN-N + A-UNINFL + GEN-CON + N "3"
23. DOMAIN-N + GEN-CON + DOMAIN-N + GEN-CON + ADV "3"
24. DOMAIN-N + GEN-CON + DOMAIN-N + GEN-CON + N "3"
25. DOMAIN-N + GEN-CON + DOMAIN-N + PREP + DOMAIN-N "3"
26. DOMAIN-N + GEN-CON + DOMAIN-V + GEN-CON + N "3"
27. DOMAIN-N + GEN-CON + DOMAIN-V + DOMAIN-N "3"
28. DOMAIN-N + GEN-CON + INF + GEN-CON + DOMAIN-N "3"
29. DOMAIN-N + GEN-CON + INF + DOMAIN-N "3"
30. DOMAIN-N + GEN-CON + INF + N "3"
31. DOMAIN-N + GEN-CON + N + A-INFL "3"
32. DOMAIN-N + GEN-CON + N + CC + N "3"
33. DOMAIN-N + GEN-CON + N + GEN-CON + DOMAIN-N "3"
34. DOMAIN-N + GEN-CON + N + DOMAIN-V "3"

35. DOMAIN-N + GEN-CON + N + PREP + N "3"
36. DOMAIN-N + GEN-CON + PREP + N "3"
37. DOMAIN-N + GEN-CON- DOMAIN-N + GEN-CON + DOMAIN-N "3"
38. DOMAIN-N + N + GEN-CON + DOMAIN-N "3"
39. DOMAIN-N + POSS + DOMAIN-N + CARD "3"
40. DOMAIN-N + PREP + DOMAIN-N + GEN-CON + DOMAIN-N "3"
41. DOMAIN-N + PREP + N + GEN-CON + DOMAIN-N "3"
42. DOMAIN-V + GEN-CON + DOMAIN-N + A-INFL "3"
43. DOMAIN-V + GEN-CON + N + GEN-CON + ADV "3"
44. DOMAIN-V + GEN-CON + N + GEN-CON + DOMAIN-N "3"
45. DOMAIN-V + DOMAIN-N + PREP + DOMAIN-V "3"
46. DOMAIN-V + N + GEN-CON + DOMAIN-N "3"
47. DOMAIN-V + N + GEN-CON + N "3"
48. INF + DOMAIN-N + GEN-CON + INF "3"
49. INF + DOMAIN-N + GEN-CON + N "3"
50. INF + DOMAIN-N + LOC "3"
51. INF + N + DOMAIN-N "3"
52. N + A-INFL + GEN-CON + DOMAIN-N "3"
53. N + GEN-CON + ADV + GEN-CON + DOMAIN-N "3"
54. N + GEN-CON + DOMAIN-N + A-INFL "3"
55. N + GEN-CON + DOMAIN-N + GEN-CON + N "3"
56. N + GEN-CON + N + GEN-CON + DOMAIN-N "3"
57. N + GEN-CON + ORD + GEN-CON + DOMAIN-N "3"
58. N + POSS + DOMAIN-N + GEN-CON + N "3"
59. N + PREP + N + GEN-CON + DOMAIN-N "3"
60. DOMAIN-N + A-UNINFL + GEN-CON + N + GEN-CON + DOMAIN-N "4"
61. DOMAIN-N + A-UNINFL + GEN-CON + N + GEN-CON + DOMAIN-V "4"
62. DOMAIN-N + GEN-CON + DOMAIN-N + GEN-CON + N + GEN-CON + DOMAIN-N "4"
63. DOMAIN-N + GEN-CON + INF + GEN-CON + DOMAIN-N + GEN-CON + ADV "4"
64. DOMAIN-N + GEN-CON + N + GEN-CON + N + GEN-CON + DOMAIN-N "4"
65. DOMAIN-N + GEN-CON + N + N + GEN-CON + DOMAIN-N "4"
66. DOMAIN-N + INF + PREP + N + GEN-CON + DOMAIN-N "4"
67. DOMAIN-N + GEN-CON + DOMAIN-N + PREP + N + GEN-CON + DOMAIN-N "4"
68. DOMAIN-N + GEN-CON + DOMAIN-V + DOMAIN-N + DOMAIN-V "4"
69. DOMAIN-V + ADV + GEN-CON + DOMAIN-N + GEN-CON + DOMAIN-N "4"
70. DOMAIN-V + GEN-CON + N + GEN-CON + N + GEN-CON + DOMAIN-N "4"
71. DOMAIN-V + DOMAIN-N + PREP + N + GEN-CON + DOMAIN-N "4"
72. DOMAIN-V + N + GEN-CON + ADV + ADV "4"
73. DOMAIN-V + N + DOMAIN-N + GEN-CON + N "4"
74. DOMAIN-V + ADV + N + GEN-CON + ADV "4"
75. INF + ADV + PREP + DOMAIN-N + GEN-CON + DOMAIN-N "4"
76. INF + GEN-CON + N + N + GEN-CON + DOMAIN-V "4"
77. INF + DOMAIN-N + N + GEN-CON + INF "4"
78. INF + DOMAIN-N + POSS + DOMAIN-N + LOC "4"
79. INF + DOMAIN-N + PREP + DOMAIN-N + GEN-CON + DOMAIN-V "4"
80. N + GEN-CON + ADV + GEN-CON + DOMAIN-V + DOMAIN-N "4"
81. N + GEN-CON + DOMAIN-N + INF + CC + DOMAIN-N "4"
82. N + GEN-CON + N + GEN-CON + DOMAIN-N + GEN-CON + N "4"
83. N + GEN-CON + N + GEN-CON + DOMAIN-N + N "4"
84. N + GEN-CON + N + GEN-CON + N + GEN-CON + DOMAIN-N "4"
85. DOMAIN-N + GEN-CON + DOMAIN-N + GEN-CON + DOMAIN-N + A-INFL + GEN-CON + DOMAIN-N "5"
86. DOMAIN-V + CC + DOMAIN-V + GEN-CON + DOMAIN-N + GEN-CON + N + GEN-CON + DOMAIN-N "5"
87. DOMAINV + DOMAIN-N + ADV + DOMAIN-N + A-UNINFL "5"

Appendix 6: Regular Expressions to the NLTK RegExp Chunker for Kiswahili Texts

```
#Define tag patterns to find NP-chunks; PP-Chunks (prepositional phrases chunks) ; terms/collocations etc
patterns1 = """
        NP: {<DT|PP\$>?<JJ>*<NN>}
        {<NNP>+}
        {<NN>+}
        {<DT>?<JJ>*<NN>}
"""“
patterns2 = """
        PP: {<DT|PP\$>?<JJ>*<NN>}
        {<NNP>+}
        {<NN>+}
"""“
patterns3 = """
        TP: {<DT|PP\$>?<JJ>*<NN>}
        {<NNP>+}
        {<NN>+}
"""“
```

Appendix 7: List of Institutions and Companies

1. University of Nairobi
2. Iron-Speed
3. Jomo Kenyatta University of Agriculture and Technology
4. Africa Nazarene University
5. Inoorero University
6. Daystar University
7. Kabarak University
8. Kenya Methodist University
9. KCA University College
10. Zetech College
11. Institute of Advanced Technology (IAT)
12. Nairobi Institute of Business Studies (School of Computer Sciences)

Software Development Companies

1. Futures Group, Kenya office- Ngong Road
2. Sybrin Kenya Ltd, Victoria Towers, Kilimanjaro Ave, Nairobi Hill
3. Ascribe Ltd, Software Developers, Citadel Bldg, 3rd Flr, Muthithi Rd, Nairobi
4. Seven seas Technology Group, Riverside Drive, Nairobi
5. System Integration Limited, Symphony Place, Waiyaki Way, Westlands, Nairobi
6. Idea Kenya, Nairobi
7. Safemark Group Ltd, Crawford Business Park, State Hse Road
8. ICT Center, University of Nairobi
9. Wilcom Systems Kenya Limited
10. Software Dynamics, Nairobi
11. TechnoBrain (K) Ltd, Nairobi
12. Software Technologies Limited, Gigiri Shopping Center, Limuru Rd Nairobi
13. Comp-rite Kenya Limited, Crescent Business Centre, Parklands Road, Nairobi, Kenya.
14. Adelphi Africa (Software Developers) Ltd, Vision Plaza, Mombasa Rd, Nairobi
15. Digital Horizons (Software Developers) Ltd, Occidental Plaza, Muthithi Rd, Parklands, Nairobi
16. Computech Limited, Nairobi

Appendix 8: Prototype's Python Code for Concept Identification and Assembly

```
from __future__ import division
def sasa():
    """ IDENTIFIES QUERY CONCEPTS & MATCHES AGAINST ONTOLOGY ELEMENTS; PRUNES CANDIDATES & ASSEMBLES. """
    import nltk, re, pprint
    #works with object property
    kount = 0
    fr = open('C:/Program Files/Protege_3.4.4/JulyNortha.owl', 'rU')
    raw1 = "start"
    outputk = nltk.word_tokenize(raw1)
    while outputk != "[]" and outputk is not []:
        if outputk != []:
            kount += 1
            raw1 = fr.readline()
            outputk = nltk.word_tokenize(raw1)
        else:
            print " Prototype by Lawrence Muchemi - UoN (Kenya) & SFU (BC-Canada)"
            print "         PhD Supervisor - Dr. Wanjiku Nga'ng'a- Uon-Kenya"
            print "             Supervised - Prof. Fred Popowich- SFU-Canada"
            print "     Ontology: OWL Full; Size = ", kount, 'lines'
            print "             (March-Sept. 2010)"
            print "------------------------------------------------------------------"
            break
    output2,origlist,initialist, multitablist = askprune()
    head(output2)
    body(output2,origlist,multitablist)
    constraint (output2,origlist, multitablist)
    superative(output2,origlist, initialist)
    if 'maximum' not in initialist and 'minimum' not in initialist and 'largest' not in initialist and 'biggest' not in initialist and 'least' not in initialist and 'smallest' not in initialist and 'shortest' not in initialist and 'lowest' not in initialist and 'highest' not in initialist and 'longest' not in initialist and 'most' not in initialist:
```

```
        print '}'
    print "-----------------------------------------------------------------"
    print "Now copy and paste me on SPARQL Query Panel of Protege"
def askprune():
    import nltk, re, pprint
    lancaster = nltk.LancasterStemmer()
    raw = raw_input("Enter your NL query here: ")
    tokens = nltk.word_tokenize(raw)
    output2 = [w.lower() for w in tokens]
    initialist= output2
    multitablist= output2
    #multitablist = list(set(multitablist))
    #print "---Tokenized Raw Input from User", output2
    mylist = list(output2)
    #print mylist
    lex(mylist)
    mmlist = list(output2)
    syno(mylist)
    #print mylist
    lexlist = syno(mylist)
    mylist = [w.lower() for w in lexlist]
    mylist = set(mylist)
    mmlist = set(mmlist)
    mylist = sorted(mylist | mmlist)
    mylist = list(mylist)
    output2=mylist
    origlist=mylist
    #print origlist
    output2=[lancaster.stem(t) for t in output2]
    output2 = sorted (set(output2))
    output2 = list(output2)
    #print "---Expanded & stemmed with synonyms", output2, origlist,initialist, multitablist
```

```
    print '                                        '
    return output2,origlist,initialist, multitablist

def head(output2):
    import nltk, re, pprint
    lancaster = nltk.LancasterStemmer()
    kount = 0
    fr = open('C:/Program Files/Protege_3.4.4/JulyNortha.owl', 'rU')
    raw1 = "start"
    outputk = nltk.word_tokenize(raw1)
    while outputk != "'[]'" and outputk is not []:
        if outputk != []:
            kount += 1
            raw1 = fr.readline()
            outputk = nltk.word_tokenize(raw1)
            if 'xmlns'in outputk and '='in outputk and 'http'in outputk and '.owl'in outputk[5]:
                print 'PREFIX dbs: <'+outputk[3]+outputk[4]+outputk[5]+outputk[6]+'>'
        else:
            print 'SELECT',
            break
    fr = open('C:/Program Files/Protege_3.4.4/JulyNortha.owl', 'rU')
    raw = "start"
    kount = kount - 2
    count=0
    k=0
    while kount != 0:
        kount += -1
        k += 1
        raw = fr.readline()
        #print 'raw.......', raw, k
        tokens = nltk.word_tokenize(raw)
        #print 'tokens.......', tokens
```

```
    otherlist = [w.lower() for w in tokens]
    #print 'otherlist.....', otherlist
    output1 = nltk.word_tokenize(raw)
    #print ' 22222 Tokens Output', output1
    #print 'output1 list before filter', output1
    if ("'owl'" in output1) and ("'rdf'" in output1) and ("':'" in output1) and ("'='" in output1) and ('FunctionalProperty' in output1):#and ("'ID'"in output1) \remov DatatypeProperty feb5/2' funct prop
        n = output1.index('=')
        #print 'output1 list after index', output1
        #print '.....', n
        m = n+6
        q = output1[m]
        #print '--', q
        r = q[:3]
        r1 = q[:2]
        #print r #'calling head here '
        if (r != 'has') and (r1 != 'is'):
            s = q.split('.')
            #print '--', s
            t1 = s[1]
            t2 = s[0]
            #print r
            list2 = t2
            list1 = t1
            #print 't1 and t2', t1, t2
            list21= lancaster.stem(list2)
            mylist = output2
            #print 'mylist===', mylist
            for t in mylist:
                d = mylist[mylist.index(t)]
                #print 'd===', d
                if d in list21 and len(d)>=3:
```

```
                p=t1
                #print 'd in list21 d=', d
                #print 't1===property', t1
                head_prop(p,otherlist,mylist,output1,k,t2,t1)
    elif ("'>'" is output1[0]) and ("':'" in output1) and ("'<'" in output1) and ("'.'" in output1[output1.index('<')+1]) and ("'/'"in output1[output1.index('<')+1]):
        a = output1[output1.index(':')+1]
        # print "this is for split:", a
        b = a.split('.')
        t1 = b[1]
        t2 = b[0]
        t21 = lancaster.stem(t2)
        mylist = output2
        for t in mylist:
            d = mylist[mylist.index(t)]
            if d in otherlist and d!= "'" and d!= "'''" and d!= "[" and d!= "]" and  d!= "." and d!= ":" and d!= "," and d!= ")" and d!= "(" and d!= "p0" and d!= "p1" and d!= "p2" and d!= "tp3":
                p=t1
                print 'calling 2'
                head_prop(p,otherlist,mylist,output1,k,t2,t1)
            tx = t1.lower()
            if d in tx:
                p=tx
                print 'calling 3'
                head_prop(p,otherlist,mylist,output1,k,t2,t1)


    elif ("'>'" is output1[0]) and ("'<'" in output1) and ("'.'" in output1[output1.index('<')+1]) and ("'/'"in output1[output1.index('<')+1]):
        a = output1[output1.index('<')+1]
        #print "this is for split:", a
        b = a.split('.')
        t1 = b[1]
        t2 = b[0]
```

```
        t3 = t2[1:]
        mylist = output2
        for t in mylist:
            d = mylist[mylist.index(t)]
            if  d in otherlist and d!= """ and d!='the' and d!='and'and d!='all'and d!= "'""" and d!= "[" and d!= "]" and d!= "." and d!= ":" and d!= "," and d!= ")" and d!= "(" and d!= "p0" and d!= "p1" and d!= "p2" and d!= "tp3":
                inst =output1[1:output1.index('<')]
                count += 1
                p=t1
                i= inst
                #print 'calling 4'
                head_prop(p,otherlist,mylist,output1,k,t2,t1)
                head_inst(i,count,t1,mylist,t3)


    print ' '
    print 'WHERE  {',


def head_prop(p,otherlist,mylist,output1,k,t2,t1):
    import nltk
    lancaster = nltk.LancasterStemmer()
    p1=p
    p1 = lancaster.stem(p1)
    t21= lancaster.stem(t2)
    if p1 in mylist and t21 in mylist:
        tscore=1
        print  ' ?'+p,
    else:
        s=p
        kn = 0
        for t in s:
            if t.isupper():
                s.find(t)
```

```
            kn += 1
    tscore=0
    #print s, tscore
    for t in s:
        if t.isupper():
            s0=s.split(t)
            #print s0
            s1=s0[0]
            s2=t.lower()+s0[0]# return to 1
            #print '+++', s2
            #print '****',t.lower()
            s=s2
            #print 's2=.....', s2
            s1 = lancaster.stem(s1)
            if s1 in mylist:
                score=1/kn
                tscore=tscore+score
    s2 = lancaster.stem(s2)
    t21= lancaster.stem(t2)
    if s2 in mylist and t21 in mylist:
        tscore= tscore +1/kn
        #print tscore
        if tscore >=0.5:
            #print s2,t21,tscore,mylist
            print  ' ?'+p,
if 'who' in mylist:
    if p.lower() == 'firstname':
        print  ' ?'+p,
if 'who' in mylist:
    if p.lower() == 'lastname':
        print  ' ?'+p,
if 'who' in mylist:
```

```
    if 'contact' in p.lower():
        print  ' ?'+p,
if 'wher' in mylist and 'country' not in mylist:
    if p.lower() == 'country':
        if p.lower() in mylist:
            print  ' ?'+p,
if 'wher' in mylist and 'reg' not in mylist:
    if p.lower() == 'region':
        if p.lower() in mylist:
            print  ' ?'+p,
if 'wher' in mylist and 'city' not in mylist:
    if p.lower() == 'city':
        print  ' ?'+p,
if 'when' in mylist:
    if 'hir' in p.lower():
        print  ' ?'+p,
if 'when' in mylist:
    if 'bir' in p.lower():
        print  ' ?'+p,
if 'when' in mylist:
    if p.lower() == 'dat':
        print  ' ?'+p,
if 'which' in mylist:
    if 'id' in p.lower():
        if p.lower()!= 'categoryid' :
            if t21 in mylist:
                print  ' ?'+p,
if 'which' in mylist:
    if 'nam' in p.lower():
        if p.lower()!= 'firstnam'and p.lower()!= 'lastnam' and p.lower()!= 'contactnam' and p.lower()!= 'categoryna':
            if t21 in mylist:
                print  ' ?'+p,
```

```
def head_inst(i,count,t1,mylist,t3):
    import pprint, pickle, nltk
    lancaster = nltk.LancasterStemmer()
    scor=0.75
    s=t1
    kn = 0
    for t in s:
        if t.isupper():
            s.find(t)
            kn += 1
    tscore=0
    for t in s:
        if t.isupper():
            s0=s.split(t)
            s1=s0[0]
            s2=t.lower()+s0[0] # Return to s0[1]
            s=s2
            s11= lancaster.stem(s1)
            if s11 in mylist:
                score=1/kn
                tscore=tscore+score
                s21= lancaster.stem(s2)
                t31= lancaster.stem(t3)
                if s21 in mylist and t31 in mylist:
                    tscore=1
                    print ' ?'+t1,


def body(output2,origlist, multitablist):
    running = True
    import nltk, re, pprint
    lancaster = nltk.LancasterStemmer()
    kount = 0
```

```
fr = open('C:/Program Files/Protege_3.4.4/JulyNortha.owl', 'rU')
raw1 = "start"
outputk = nltk.word_tokenize(raw1)
while outputk != "'[]'" and outputk is not []:
    if outputk != []:
        kount += 1
        raw1 = fr.readline()
        outputk = nltk.word_tokenize(raw1)
    else:
        print ' '
        break
fr = open('C:/Program Files/Protege_3.4.4/JulyNortha.owl', 'rU')
raw = "start"
kount = kount - 2
count=0
k=0
classlist = []
while kount != 0:
    kount += -1
    k += 1
    raw = fr.readline()
    tokens = nltk.word_tokenize(raw)
    otherlist = [w.lower() for w in tokens]
    output1 = nltk.word_tokenize(raw)
    if ("'owl'" in output1) and ("'rdf'" in output1) and ("':'" in output1) and ("'='" in output1) and ('FunctionalProperty'
in output1):#changed
        n = output1.index('=')
        m = n+6 #changed
        q = output1[m]
        r = q[:3]
        r1 = q[:2]
        if (r != 'has') and (r1 != 'is'):
```

```
            s = q.split('.')
            t1 = s[1]
            t2 = s[0]
            list2 = t2
            list1 = t1
            list11= lancaster.stem(list1)
            list21= lancaster.stem(list2)
            mylist = output2
            for t in mylist:
               d = mylist[mylist.index(t)]
               if d in otherlist and d!= ""and len(d)>=3 and d!='the':
                  p=t1
                  #print 'calling a'
                  triples(p,otherlist,mylist,output1,t2,t1, multitablist)


               if d in list11 and len(d)>=3 and d!='the':
                  p=t1
                  #print 'calling b'
                  triples(p,otherlist,mylist,output1,t2,t1, multitablist)
               if d in list21 and len(d)>=3:
                  p=t1
                  #print 'calling c'
                  triples(p,otherlist,mylist,output1,t2,t1, multitablist)
                  d1 = [mylist[mylist.index(t)]]
                  if classlist != None and d1 != []:
                     pass
                  d2= classlist.insert(0, (d1))
      elif ('">'" is output1[0]) and ('":'" in output1) and ('"<'" in output1) and ('"."' in output1[output1.index('<')+1]) and
('"/"'in output1[output1.index('<')+1]):
         a = output1[output1.index(':')+1]
        # print "this is for split:", a
         b = a.split('.')
```

```
        t1 = b[1]
        t2 = b[0]
        t21 = lancaster.stem(t2)
        mylist = output2
        for t in mylist:
            d = mylist[mylist.index(t)]
            if d in otherlist and d!= "'" and d!= '"' and d!= "[" and d!= "]" and  d!= "." and d!= ":" and d!= "," and d!= ")" and d!= "(" and d!= "p0" and d!= "p1" and d!= "p2" and d!= "tp3":
                p=t1
                print 'calling e'
                triples(p,otherlist,mylist,output1,t2,t1, multitablist)
            tx = t1.lower()
            tx1= lancaster.stem(tx)
            if d in tx1:
                p=tx
                print 'calling f'
                triples(p,otherlist,mylist,output1,t2,t1, multitablist)



if multitablist != None :
    multitablist = list(set(multitablist))
    multitablist=[lancaster.stem(t) for t in multitablist]
    multitablist = list(set(multitablist))
    #33print 'this is ~~~', multitablist,type(multitablist)
#classlist = list(set(classlist))
#33print 'finaly', classlist
#33print 'individual', classlist[2],
common=[]
j=len(multitablist)-1
while j>=0:
    #print j, multitablist[j], str(classlist)
    if multitablist[j] in str(classlist):
```

```
            #print 'Participating classes', multitablist[j],list11
            s=list([multitablist[j]])
            common1=common.append(s)
        j-=1
    #print 'commons==', common
    #33print 'am done'
    if len(common)>1 and len(common)<=3 :
        fkey(common, mylist)

def triples(p,otherlist,mylist,output1,t2,t1, multitablist):
    import nltk, re, pprint
    lancaster = nltk.LancasterStemmer()
    p1=p
    p1 = lancaster.stem(p1)
    t21= lancaster.stem(t2)
    if p1 in mylist and t21 in mylist:
        tscore=1
        print '?'+t2,'dbs:'+t2+'.'+p, ' ?'+p+'.'# causes lots of repetations in properties list
        #type(t2)
        multitablist = multitablist.append(t2)
        #multitablist = list(set(multitablist))
    else:
        s=p
        kn = 0
        for t in s:
            if t.isupper():
                s.find(t)
                kn += 1
        tscore=0
        for t in s:
            if t.isupper():
                s0=s.split(t)
```

```
            s1=s0[0]
            s2=t.lower()+s0[0] # return to s[1]
            s=s2
            s11 = lancaster.stem(s1)
            if s11 in mylist:
               score=1/kn
               tscore=tscore+score
      s21 = lancaster.stem(s2)
      t21= lancaster.stem(t2)
      if s21 in mylist and t21 in mylist:
         tscore= tscore +1/kn
         if tscore >= 0.5:
            print '?'+t2,'dbs:'+t2+'.'+p, ' ?'+p+'.'
            multitablist = multitablist.append(t2)

if 'who' in mylist and t21 in mylist:
   if p.lower() == 'firstname':
      print '?'+t2,'dbs:'+t2+'.'+p, ' ?'+p+'.'
if 'who' in mylist and t21 in mylist:
   if p.lower() == 'lastname':
      print '?'+t2,'dbs:'+t2+'.'+p, ' ?'+p+'.'
if 'who' in mylist and t21 in mylist:
   if 'contact' in p.lower():
      print '?'+t2,'dbs:'+t2+'.'+p, ' ?'+p+'.'
if 'wher' in mylist and 'country' not in mylist:
   if p.lower() == 'country':
      if p.lower() in mylist:
         print '?'+t2,'dbs:'+t2+'.'+p, ' ?'+p+'.'
if 'wher' in mylist and 'reg' not in mylist:
   if p.lower() == 'region':
      if p.lower() in mylist:
         print '?'+t2,'dbs:'+t2+'.'+p, ' ?'+p+'.'
```

```
    if 'wher' in mylist and 'city' not in mylist:
        if p.lower() == 'city':
            print '?'+t2,'dbs:'+t2+'.'+p, ' ?'+p+'.'
    if 'when' in mylist:
        if 'hir' in p.lower():
            print '?'+t2,'dbs:'+t2+'.'+p, ' ?'+p+'.'
    if 'when' in mylist:
        if 'bir' in p.lower():
            print '?'+t2,'dbs:'+t2+'.'+p, ' ?'+p+'.'
    if 'when' in mylist:
        if p.lower() == 'date':
            print '?'+t2,'dbs:'+t2+'.'+p, ' ?'+p+'.'
    if 'which' in mylist:
        if 'id' in p.lower():
            if t21 in mylist:
                print '?'+t2,'dbs:'+t2+'.'+p, ' ?'+p+'.'
    if 'which' in mylist:
        if 'nam' in p.lower():
            if p.lower()!= 'firstnam'and p.lower()!= 'lastnam' and p.lower()!= 'contactnam' and p.lower()!= 'categoryna':
                if t21 in mylist:
                    print '?'+t2,'dbs:'+t2+'.'+p, ' ?'+p+'.'

def constraint (output2,origlist, multitablist):
    import nltk, re, pprint
    lancaster = nltk.LancasterStemmer()
    kount = 0
    fr = open('C:/Program Files/Protege_3.4.4/JulyNortha.owl', 'rU')
    raw1 = "start"
    outputk = nltk.word_tokenize(raw1)
    while outputk != "'[]'" and outputk is not []:
        if outputk != []:
            kount += 1
```

```
        raw1 = fr.readline()
        outputk = nltk.word_tokenize(raw1)
    else:
        print ' '
        break
#output2 = askprune()
fr = open('C:/Program Files/Protege_3.4.4/JulyNortha.owl', 'rU')
raw = "start"
kount = kount - 2
count=0
k=0
while kount != 0:
    kount += -1
    k += 1
    raw = fr.readline()
    tokens = nltk.word_tokenize(raw)
    otherlist = [w.lower() for w in tokens]
    output1 = nltk.word_tokenize(raw)
    #########print 'output1------', output1
    #if ("'>'" is output1[0]) and ("'<'" in output1) and ("'.'" in output1[output1.index('<')+1]) and ("'/'"in
output1[output1.index('<')+1]):
    #if ("'>'" is output1[0]) and ("'<'" in output1) and ("'.'" in output1[output1.index('<')+1]) and ("'/'"in
output1[output1.index('<')+1]):
    if ("'db'" in output1) and ("'rdf'" in output1) and ("':'" in output1) and ("'='" in output1) and ('datatype' in output1)
and ('hasPrimaryKeyFields' not in output1) and ('isBridgeTable' not in output1) and ('hasOrigColumnName' not in
output1):
        a = output1[output1.index(':')+1]
        #print "this is for split:", a
        b = a.split('.')
        t1 = b[1]
        t3 = b[0]
        t2 = t3[1:]
        t21 = lancaster.stem(t2)
```

```
        mylist = output2
        for t in origlist:
            d = origlist[origlist.index(t)]
            if  d in otherlist and d!= "+" and d!= "" and d!='the' and d!='and'and d!='all'and d!= """" and d!= "[" and d!= "]" and d!= "." and d!= ":" and d!= "," and d!= ")" and d!= "(" and d!= "p0" and d!= "p1" and d!= "p2" and d!= "tp3":
                inst =output1[1:output1.index('<')]
                count += 1
                p=t1
                i= inst
                filter1(i,count,t1,mylist,t2,origlist)
                t11 = lancaster.stem(t1)
                t21 = lancaster.stem(t2)


def filter1(i,count,t1,mylist,t2,origlist):
    import nltk, re, pprint, pickle
    ln=0
    ct=0
    lancaster = nltk.LancasterStemmer()
    listinst=[t2,t1,i]
    t11 = lancaster.stem(t1)
    t21 = lancaster.stem(t2)
    l=[w.lower() for w in i]
    r= [w.lower() for w in origlist]
    #print 'r===', r
    for w in l:
        ln += 1
    if ln >1:
        for w in l:
            if w in origlist:
                ct+=1
```

```
    print ct, ln
    conf=ct/ln ## pushed in one step
if ln > 1 and conf>0.5 and t21 in mylist and t11 in mylist :
    ir= ' '.join(i)
    print '?'+t2,'dbs:'+t2+'.'+t1, ' ?'+t1+'.'
    print 'FILTER(?'+t1,'=',"  '"+ir+"' " ,')'
    #_____________________
##>>>>>if l[len(i)-1] in r and t21 in mylist and t11 in mylist and ln==1 and 'not' not in mylist and ('mor' not in mylist and 'gre' not in mylist) and ('less' not in mylist and 'few' not in mylist):## This works fine: print 'kkkk', if l[0] in r
    #print '----', l[len(i)-1] #, t1, str(i)
    if t1.lower() not in str(i):
        if '+'in i:
            inw = l.remove('+')
            u=str(l)
            print '?'+t2,'dbs:'+t2+'.'+t1, ' ?'+t1+'.'
            print 'FILTER(?'+t1,'=',"  '"+i[len(i)-1]+"' " ,')'
        if '@' in i:
            ir= ''.join(i)
            print '?'+t2,'dbs:'+t2+'.'+t1, ' ?'+t1+'.'
            print 'FILTER(?'+t1,'=',"  '"+ir+"' " ,')'
        if ',' in i:
            for w in i:
                if ',' in i :
                    i.remove(',')
            ir= ', '.join(i)
            ir= str(ir)
            print '?'+t2,'dbs:'+t2+'.'+t1, ' ?'+t1+'.'
            print 'FILTER(?'+t1,'=',"  '"+ir+"' " ,')'
        elif '+'not in i and '@' not in i :
            ir= ' '.join(i)
            print '?'+t2,'dbs:'+t2+'.'+t1, ' ?'+t1+'.'
```

```
        print 'FILTER(?'+t1,'=',' '"+ir+'" ' ,')'
        #== less than Begins ==========#
##>>>>>if l[len(i)-1] in r and t21 in mylist and t11 in mylist and ln==1 and 'not' not in mylist and ('less' in mylist or 'few' in mylist) and ('mor' not in mylist and 'gre' not in mylist):## This works fine: print 'kkkk', if l[0] in r
    #print l[len(i)-1], t1, str(i)
    if t1.lower() not in str(i):
        if '+'in i:
            inw = l.remove('+')
            u=str(l)
            print '?'+t2,'dbs:'+t2+'.'+t1, ' ?'+t1+'.'
            print 'FILTER(?'+t1,'<',' '+"+i[len(i)-1]+'" ' ,')'
        if '@' in i:
            ir= ''.join(i)
            print '?'+t2,'dbs:'+t2+'.'+t1, ' ?'+t1+'.'
            print 'FILTER(?'+t1,'<',' '"+ir+'" ' ,')'
        if ',' in i:
            for w in i:
                if ',' in i :
                    i.remove(',')
            ir= ', '.join(i)
            ir= str(ir)
            print '?'+t2,'dbs:'+t2+'.'+t1, ' ?'+t1+'.'
            print 'FILTER(?'+t1,'<',' '"+ir+'" ' ,')'
        elif '+'not in i and '@' not in i :
            ir= ' '.join(i)
            print '?'+t2,'dbs:'+t2+'.'+t1, ' ?'+t1+'.'
            print 'FILTER(?'+t1,'<',' '+ir+' ' ,')'
            #== Greater than Begins ==========#
#>>>>>if l[len(i)-1] in r and t21 in mylist and t11 in mylist and ln==1 and 'not' not in mylist and ('mor' in mylist or 'gre' in mylist) and ('less' not in mylist and 'few' not in mylist):## This works fine: print 'kkkk', if l[0] in r
    #print l[len(i)-1], t1, str(i)
    if t1.lower() not in str(i):
```

```
        if '+'in i:
            inw = l.remove('+')
            u=str(l)
            print '?'+t2,'dbs:'+t2+'.'+t1, ' ?'+t1+'.'
            print 'FILTER(?'+t1,'>'," '+"+i[len(i)-1]+'" " ,')'
        if '@' in i:
            ir= ''.join(i)
            print '?'+t2,'dbs:'+t2+'.'+t1, ' ?'+t1+'.'
            print 'FILTER(?'+t1,'>'," '"+ir+'" " ,')'
        if ',' in i:
            for w in i:
                if ',' in i :
                    i.remove(',')
            ir= ', '.join(i)
            ir= str(ir)
            print '?'+t2,'dbs:'+t2+'.'+t1, ' ?'+t1+'.'
            print 'FILTER(?'+t1,'>'," '"+ir+'" " ,')'
        elif '+'not in i and '@' not in i :
            ir= ' '.join(i)
            print '?'+t2,'dbs:'+t2+'.'+t1, ' ?'+t1+'.'
            print 'FILTER(?'+t1,'>'," "+ir+" " ,')'
            #== Negation Begins================#
#>>>>>if l[len(i)-1] in r and t21 in mylist and t11 in mylist and ln==1 and 'not'in mylist and ('less' not in mylist and
'few' not in mylist) and ('mor' not in mylist and 'gre' not in mylist):## This works fine: print 'kkkk', if l[0] in r
    if t1.lower() not in str(i):
        if '+'in i:
            inw = l.remove('+')
            u=str(l)
            print '?'+t2,'dbs:'+t2+'.'+t1, ' ?'+t1+'.'
            print 'FILTER(?'+t1,'!='," '+"+i[len(i)-1]+'" " ,')'
        if '@' in i:
            ir= ''.join(i)
```

```
            print '?'+t2,'dbs:'+t2+'.'+t1, ' ?'+t1+'.'
            print 'FILTER(?'+t1,'!=',' '"+ir+'" ' ,')'
        if ',' in i:
            for w in i:
                if ',' in i :
                    i.remove(',')
            ir= ', '.join(i)
            ir= str(ir)
            print '?'+t2,'dbs:'+t2+'.'+t1, ' ?'+t1+'.'
            print 'FILTER(?'+t1,'!=',' '"+ir+'" ' ,')'
        elif '+'not in i and '@' not in i :
            ir= ' '.join(i)
            print '?'+t2,'dbs:'+t2+'.'+t1, ' ?'+t1+'.'
            print 'FILTER(?'+t1,'!=',' '"+ir+'" ' ,')'
            #== Negation of less than Begins===============#
#>>>>>if l[len(i)-1] in r and t21 in mylist and t11 in mylist and ln==1 and 'not'in mylist and ('less' in mylist or 'few' in mylist) and ('mor' not in mylist and 'gre' not in mylist):## This works fine: print 'kkkk', if l[0] in r
    #print l[len(i)-1], t1, str(i)
    if t1.lower() not in str(i):
        if '+'in i:
            inw = l.remove('+')
            u=str(l)
            print '?'+t2,'dbs:'+t2+'.'+t1, ' ?'+t1+'.'
            print 'FILTER(?'+t1,'>=',' '+'"+i[len(i)-1]+'" ' ,')'
        if '@' in i:
            ir= ''.join(i)
            print '?'+t2,'dbs:'+t2+'.'+t1, ' ?'+t1+'.'
            print 'FILTER(?'+t1,'>=',' '"+ir+'" ' ,')'
        if ',' in i:
            for w in i:
                if ',' in i :
                    i.remove(',')
```

```
            ir= ', '.join(i)
            ir= str(ir)
            print '?'+t2,'dbs:'+t2+'.'+t1, ' ?'+t1+'.'
            print 'FILTER(?'+t1,'>=',' "'+ir+'" " ,')'
        elif '+'not in i and '@' not in i :
            ir= ' '.join(i)
            print '?'+t2,'dbs:'+t2+'.'+t1, ' ?'+t1+'.'
            print 'FILTER(?'+t1,'>=',' "'+ir+" " ,')'
            #== Negation of Greater than Begins================#
#>>>>>if l[len(i)-1] in r and t21 in mylist and t11 in mylist and ln==1 and 'not'in mylist and ('mor' in mylist or 'gre' in mylist) and ('less' not in mylist and 'few' not in mylist):## This works fine: print 'kkkk', if l[0] in r
    #print l[len(i)-1], t1, str(i)
    if t1.lower() not in str(i):
        if '+'in i:
            inw = l.remove('+')
            u=str(l)
            print '?'+t2,'dbs:'+t2+'.'+t1, ' ?'+t1+'.'
            print 'FILTER(?'+t1,'<=',' '+"'+i[len(i)-1]+'" " ,')'
        if '@' in i:
            ir= ''.join(i)
            print '?'+t2,'dbs:'+t2+'.'+t1, ' ?'+t1+'.'
            print 'FILTER(?'+t1,'<=',' "'+ir+'" " ,')'
        if ',' in i:
            for w in i:
                if ',' in i :
                    i.remove(',')
            ir= ', '.join(i)
            ir= str(ir)
            print '?'+t2,'dbs:'+t2+'.'+t1, ' ?'+t1+'.'
            print 'FILTER(?'+t1,'<=',' "'+ir+'" " ,')'
        elif '+'not in i and '@' not in i :
            ir= ' '.join(i)
```

```
            print '?'+t2,'dbs:'+t2+'.'+t1, ' ?'+t1+'.'
            print 'FILTER(?'+t1,'<=',"  "+ir+"  " ,')'
            #== Half Named Properties Begins===============#
s=t1
kn = 0
for t in s:
    if t.isupper():
        s.find(t)
        kn += 1
tscore=0
for t in s:
    if t.isupper():
        s0=s.split(t)
        s1=s0[0]
        s2=t.lower()+s0[0] #return to s0[1]
        s=s2
        s11 = lancaster.stem(s1)
        if s11 in mylist:
            score=1/kn
            tscore=tscore+score
            s21 = lancaster.stem(s2)
            t21= lancaster.stem(t2)
            if s21 in mylist and t21 in mylist and ln==1 and 'not' not in mylist and ('less' not in mylist and 'few' not in mylist) and 'mor' not in mylist and 'gre' not in mylist:
                tscore=1
                if '@' in i:
                    ir= ''.join(i)
                    print '?'+t2,'dbs:'+t2+'.'+t1, ' ?'+t1+'.'
                    print 'FILTER(?'+t1,'=',"  '"+ir+"'  " ,')'
                else:
                    ir= ' '.join(i)
                    print '?'+t2,'dbs:'+t2+'.'+t1, ' ?'+t1+'.'
```

```
            print 'FILTER(?'+t1,'=', " '"+ir+"' " ,')'
            #== Less than Begins ==========#
```

if s21 in mylist and t21 in mylist and ln==1 and 'not' not in mylist and ('less'in mylist or 'few' in mylist) and ('mor' not in mylist or 'gre' not in mylist):

```
            #print 'seen', s1,s2
            tscore=1
            if '@' in i:
               ir= ''.join(i)
               print '?'+t2,'dbs:'+t2+'.'+t1, ' ?'+t1+'.'
               print 'FILTER(?'+t1,'<'," '"+ir+"' " ,')'
            else:
               ir= ' '.join(i)
               print '?'+t2,'dbs:'+t2+'.'+t1, ' ?'+t1+'.'
               print 'FILTER(?'+t1,'<', " "+ir+" " ,')'
               #== Greater than Begins ===========#
```

if s21 in mylist and t21 in mylist and ln==1 and 'not' not in mylist and ('mor' in mylist or 'gre' in mylist) and ('less' not in mylist or 'few' not in mylist):

```
            #print 'seen', s1,s2
            tscore=1
            if '@' in i:
               ir= ''.join(i)
               print '?'+t2,'dbs:'+t2+'.'+t1, ' ?'+t1+'.'
               print 'FILTER(?'+t1,'>'," '"+ir+"' " ,')'
            else:
               ir= ' '.join(i)
               print '?'+t2,'dbs:'+t2+'.'+t1, ' ?'+t1+'.'
               print 'FILTER(?'+t1,'>', " "+ir+" " ,')'
               #== Negation Begins ==============#
```

if s21 in mylist and t21 in mylist and ln==1 and 'not' in mylist and ('less' not in mylist and 'few' not in mylist) and ('mor' not in mylist and 'gre' not in mylist):

```
            tscore=1
            if '@' in i:
               ir= ''.join(i)
```

```
            print '?'+t2,'dbs:'+t2+'.'+t1, ' ?'+t1+'.'
            print 'FILTER(?'+t1,'!=',' "'+ir+'" " ,')'
        else:
            ir= ' '.join(i)
            print '?'+t2,'dbs:'+t2+'.'+t1, ' ?'+t1+'.'
            print 'FILTER(?'+t1,'!=', " "'+ir+'" " ,')'#, mylist
            #== Negation of less than Begins ==#
    if s21 in mylist and t21 in mylist and ln==1 and 'not' in mylist and ('less'in mylist or 'few' in mylist) and ('mor' not in mylist and 'gre' not in mylist):
        tscore=1
        if '@' in i:
            ir= ''.join(i)
            print '?'+t2,'dbs:'+t2+'.'+t1, ' ?'+t1+'.'
            print 'FILTER(?'+t1,'>=',' "'+ir+" " ,')'
        else:
            ir= ' '.join(i)
            print '?'+t2,'dbs:'+t2+'.'+t1, ' ?'+t1+'.'
            print 'FILTER(?'+t1,'>=', " "'+ir+" " ,')0000', tscore, s21
            #== Negation of Greater than Begins =#
    if s21 in mylist and t21 in mylist and ln==1 and 'not' in mylist and ('less'not in mylist or 'few' not in mylist) and ('mor' in mylist or 'gre' in mylist):
        tscore=1
        if '@' in i:
            ir= ''.join(i)
            print '?'+t2,'dbs:'+t2+'.'+t1, ' ?'+t1+'.'
            print 'FILTER(?'+t1,'<=',' "'+ir+" " ,')'
        else:
            ir= ' '.join(i)
            print '?'+t2,'dbs:'+t2+'.'+t1, ' ?'+t1+'.'
            print 'FILTER(?'+t1,'<=', " "'+ir+" " ,')'
    if ln > 1 and conf>0.5 and t21 in mylist and s21 in mylist :
        ir= ' '.join(i)
```

```
            print '?'+t2,'dbs:'+t2+'.'+t1, ' ?'+t1+'.'
            print 'FILTER(?'+t1,'=',' '"+ir+'" ' ,')'

def syno(mylist):
    import nltk
    lexlist = []
    from nltk.corpus import wordnet as wn
    for s in mylist:
        for synset in wn.synsets(s):
            L = synset.lemma_names
            lexlist.extend(L)
    lexlist = list (sorted(set(lexlist)))
    return lexlist
def lex(mylist):
    # Geographical Database Module ##
    if 'kenya' in mylist:
        mylist.append('country')
    s= 'usa'
    syn(s, mylist)
    s= 'germany'
    syn(s, mylist)
    if 'japan' in mylist:
        mylist.append('country')
    if 'mexico' in mylist and 'city' not in mylist:
        mylist.append('country')
    if 'uganda' in mylist:
        mylist.append('country')
    if 'canada' in mylist:
        mylist.append('country')
    s= 'uk'
    syn(s, mylist)
    if 'ghana' in mylist:
```

```
    mylist.append('country')
if 'nairobi' in mylist:
    mylist.append('city')
if 'rome' in mylist:
    mylist.append('city')
    mylist.append('shipcity')
if 'new_york' in mylist and 'city' not in mylist:
    mylist.append('state')
if 'new' in mylist and 'york' in mylist:
    mylist.append('city')
if 'embu' in mylist and 'city' not in mylist:
    mylist.append('city')
if 'london' in mylist and 'city' not in mylist:
    mylist.append('city')
if 'vancouver' in mylist and 'city' not in mylist:
    mylist.append('city')
if 'toronto' in mylist and 'city' not in mylist:
    mylist.append('city')
if 'seattle' in mylist and 'city' not in mylist:
    mylist.append('city')
if 'liverpool' in mylist and 'city' not in mylist:
    mylist.append('city')
if 'berlin' in mylist and 'city' not in mylist:
    mylist.append('city')
if 'kampala' in mylist and 'city' not in mylist:
    mylist.append('city')
if 'chicago' in mylist and 'city' not in mylist:
    mylist.append('city')
if 'kumasi' in mylist and 'city' not in mylist and 'region' not in mylist:
    mylist.append('city')
if 'karatina' in mylist and 'city' not in mylist:
    mylist.append('city')
```

```
if 'where' in mylist and 'city' not in mylist and 'countries' not in mylist and 'country' not in mylist and 'region' not in mylist:
    mylist.append('city')
    #== Jargon Here ====#
if 'retailers' in mylist or 'retailer' in mylist:
    mylist.append('supplier')
if 'domestic' in mylist:
    mylist.append('kenya')
    mylist.append('country')
    #== Normalization of Country Names Begins Here ==#
if 'german' in mylist:
    mylist.remove('german')
    mylist.append('germany')
    mylist.append('country')
if 'canadian' in mylist:
    mylist.remove('canadian')
    mylist.append('canada')
    mylist.append('country')
if 'ugandan' in mylist:
    mylist.remove('ugandan')
    mylist.append('uganda')
    mylist.append('country')
if 'ghanaian' in mylist:
    mylist.remove('ghanaian')
    mylist.append('ghana')
    mylist.append('country')
    #== dealing with challenges brought about by synonym set/ stem e.g. find has syn notice stemmed to not==#

if 'find' in mylist:
    mylist.remove('find')
    mylist.append('list')
if 'biggest' in mylist:
```

```
        mylist.remove('biggest')
        mylist.append('most')
    if 'largest' in mylist:
        mylist.remove('largest')
        mylist.append('most')
    if 'maximum' in mylist:
        mylist.remove('maximum')
        mylist.append('most')
    if 'minimum' in mylist:
        mylist.remove('minimum')
        mylist.append('smallest')
    #== anticipating usage of verbs in question ======#
    if 'give' in mylist:
        mylist.append('list')
    if 'day' in mylist or 'days' in mylist:
        mylist.append('date')
def syn(s,mylist):
    import nltk
    from nltk.corpus import wordnet as wn
    #mylist = lex(mylist)
    for synset in wn.synsets(s):
        sn = synset.lemma_names
        sn = [x.lower() for x in sn]
        for w in sn:
            if w in mylist:
                mylist.append('country')

def superative(output2,origlist, initialist):
    import nltk, re, pprint
    lancaster = nltk.LancasterStemmer()
    kount = 0
    fr = open('C:/Program Files/Protege_3.4.4/JulyNortha.owl', 'rU')
```

```
raw1 = "start"
outputk = nltk.word_tokenize(raw1)
while outputk != "'[]'" and outputk is not []:
    if outputk != []:
        kount += 1
        raw1 = fr.readline()
        outputk = nltk.word_tokenize(raw1)
    else:
        print ' '
        break
fr = open('C:/Program Files/Protege_3.4.4/JulyNortha.owl', 'rU')
raw = "start"
kount = kount - 2
count=0
k=0
while kount != 0:
    kount += -1
    k += 1
    raw = fr.readline()
    tokens = nltk.word_tokenize(raw)
    otherlist = [w.lower() for w in tokens]
    output1 = nltk.word_tokenize(raw)
    if ("'owl'" in output1) and ("'rdf'" in output1) and ("':'" in output1) and ("'='" in output1) and ('FunctionalProperty'
in output1):
    # This is the section that selects Classname and Property(for ALL tables)
        n = output1.index('=')
        m = n+6
        q = output1[m]
        r = q[:3]
        r1 = q[:2]
        if (r != 'has') and (r1 != 'is'):
            #print "this is for split:", q
```

```
s = q.split('.')
t1 = s[1]
t2 = s[0]
#t2 = lancaster.stem(t2)
list2 = t2
list1 = t1
t11= lancaster.stem(list1)
t21= lancaster.stem(list2)
mylist = output2
for t in mylist:
   d = mylist[mylist.index(t)]
   if (d in t21) and len(d)>=3 and d!='the' and d!= "tp3" :# ((d in otherlist) or (d in t11) or
      p=t1
      if t11 in mylist and t21 in mylist and ('less' not in mylist and 'few' not in mylist) and 'mor' not in mylist and 'gre' not in mylist:
         if 'maximum'in initialist or'highest'in initialist or 'longest'in initialist or 'most' in initialist or 'biggest' in initialist or 'largest' in initialist:
            print '} ORDER BY DESC','(?'+t1+')'
            print 'LIMIT 1'
         if  'minimum'in initialist or'lowest'in initialist or 'shortest' in initialist or 'least' in initialist or 'smallest' in initialist:
            print '} ORDER BY ','?'+t1
            print  'LIMIT 1'
      else:
         s=p
         kn = 0
         for t in s:
            if t.isupper():
               s.find(t)
               kn += 1
         tscore=0
         for t in s:
            if t.isupper():
```

```
            s0=s.split(t)
            s1=s0[0]
            s2=t.lower()+s0[0] # return to s0[1]
            s=s2
            s11 = lancaster.stem(s1)
            if s11 in mylist:
                score=1/kn
        s21 = lancaster.stem(s2)
        t21= lancaster.stem(t2)
        #print s11, kn, s2
        if s21 in mylist and t21 in mylist:
            tscore= tscore +1/kn
            #scored =0
            if tscore >= 0.3 and  'not' not in mylist and ('less' not in mylist and 'few' not in mylist) and 'mor' not in mylist and 'gre' not in mylist:
                if 'maximum'in initialist or'highest'in initialist or 'longest'in initialist or 'most' in initialist or 'biggest' in initialist or 'largest' in initialist:
                    if 'highest'in initialist:
                        t='highest'
                        superbig(t,initialist,t1, kn)
                    if 'longest'in initialist:
                        t='longest'
                        superbig(t,initialist,t1, kn)
                    if 'most'in initialist:
                        t='most'
                        superbig(t,initialist,t1, kn)
                    if 'biggest'in initialist:
                        t='biggest'
                        superbig(t,initialist,t1, kn)
                    if 'largest'in initialist:
                        t='largest'
                        superbig(t,initialist,t1, kn)
```

```
                    if 'maximum'in initialist:
                        t='maximum'
                        superbig(t,initialist,t1, kn)


                if  'minimum'in initialist or'lowest'in initialist or 'shortest' in initialist or 'least' in initialist or
'smallest' in initialist:
                    if 'lowest'in initialist:
                        t='lowest'
                        supersmall(t,initialist,t1, kn)
                    if 'shortest'in initialist:
                        t='shortest'
                        supersmall(t,initialist,t1, kn)
                    if 'least'in initialist:
                        t='least'
                        supersmall(t,initialist,t1, kn)
                    if 'smallest'in initialist:
                        t='smallest'
                        supersmall(t,initialist,t1, kn)
                    if 'minimum'in initialist:
                        t='minimum'
                        supersmall(t,initialist,t1, kn)


def supersmall(t,initialist,t1,kn):
    scored = 0
    u = initialist[(initialist.index(t)+1):]
    if u != []:
        u = initialist[(initialist.index(t)+1):]
        for w in u:
            if w in t1.lower():
                scored += 1
        scoredf= scored*(1/kn)
        if scoredf > 0.5:
```

```
            print '} ORDER BY ','?'+t1
            print  'LIMIT 1'
        else:
            u = initialist[-6:(initialist.index(t)+1)]
            for w in u:
                if w in t1.lower():
                    scored += 1
            scoredf= scored*(1/kn)
            if scoredf > 0.5:
                print '} ORDER BY ','?'+t1
                print  'LIMIT 1'
    def superbig(t,initialist,t1,kn):
        scored = 0
        u = initialist[(initialist.index(t)+1):]
        if u != []:
            u = initialist[(initialist.index(t)+1):]
            for w in u:
                if w in t1.lower():
                    scored += 1
            scoredf= scored*(1/kn)
            if scoredf > 0.5:
                print '} ORDER BY DESC','(?'+t1+')'
                print  'LIMIT 1'
        else:
            u = initialist[-6:(initialist.index(t)+1)]
            for w in u:
                if w in t1.lower():
                    scored += 1
            scoredf= scored*(1/kn)
            if scoredf > 0.5:
                print '} ORDER BY DESC','(?'+t1+')'
                print  'LIMIT 1'
```

```
def fkey (common, mylist):
    running = True
    import nltk, re, pprint
    lancaster = nltk.LancasterStemmer()
    kount = 0
    fr = open('C:/Program Files/Protege_3.4.4/JulyNortha.owl', 'rU')
    raw1 = "start"
    outputk = nltk.word_tokenize(raw1)
    while outputk != "[]" and outputk is not []:
        if outputk != []:
            kount += 1
            raw1 = fr.readline()
            outputk = nltk.word_tokenize(raw1)
        else:
            print ' '
            break
    fr = open('C:/Program Files/Protege_3.4.4/JulyNortha.owl', 'rU')
    raw = "start"
    kount = kount - 2
    count=0
    k=0
    proplist=[]
    proplist1 = []
    proplist2 = []
    proplist3 = []
    #common = [['produc'], ['supply']]
    #print 'common', common
    while kount != 0:
        kount += -1
        k += 1
        raw = fr.readline()
```

```
    output1 = nltk.word_tokenize(raw)
    if ("'owl'" in output1) and ("'rdf'" in output1) and ("':'" in output1) and ("'='" in output1) and ('FunctionalProperty' in output1):
        n = output1.index('=')
        m = n+6
        q = output1[m]
        r = q[:3]
        r1 = q[:2]
        if (r != 'has') and (r1 != 'is'):
            #print "this is for split:", q
            s = q.split('.')
            t1 = s[1]
            t2 = s[0]
            t21= lancaster.stem(t2)
            t11= lancaster.stem(t1)
            proplist.append(t1)
            for t in mylist:
                if t in t2 and t21 in mylist and t in common[0]:
                    t2=s[0]
                    proplist1.append(t1)
                    t0=t2
                    #print 'this is my t2',t2
            ##if t21 in common[0] and t21 in mylist:
                ##proplist1.append(t1)
                #print t21
            if t21 in common[1]and t21 in mylist:
                proplist2.append(t1)
                t3=t2
            if len(common)==3 and t21 in common[2]and t21 in mylist:
                proplist3.append(t1)
                t4=t2
#print 'proplist1', proplist1
```

```
    #print '--------'
    #print 'proplist2',proplist2
    proplist1 = set(proplist1)
    proplist2 = set(proplist2)
    proplist3 = set(proplist3)
    prop_comm = sorted(proplist1 & proplist2)
    prop_comm1 = sorted(proplist1 & proplist2)
    prop_comm2 = sorted(proplist1 & proplist3)
    prop_comm3 = sorted(proplist2 & proplist3)
    #print mylist, t2
    #print prop_comm1
    #print prop_comm2
    #print prop_comm3
    #print proplist1
    if prop_comm != [] and len(common)==2:
       print '?'+str(t0),'dbs:'+str(t0)+'.'+str(prop_comm[0]), ' ?'+str(prop_comm[0])+'.'
       print '?'+str(t3),'dbs:'+str(t3)+'.'+str(prop_comm[0]), ' ?'+str(prop_comm[0])+'.'

    if prop_comm1 != [] and len(common)==3:
       print '?'+str(t0),'dbs:'+str(t0)+'.'+str(prop_comm1[0]), ' ?'+str(prop_comm1[0])+'.'
       print '?'+str(t3),'dbs:'+str(t3)+'.'+str(prop_comm1[0]), ' ?'+str(prop_comm1[0])+'.'
    if prop_comm2 != [] and len(common)==3:
       print '?'+str(t0),'dbs:'+str(t0)+'.'+str(prop_comm2[0]), ' ?'+str(prop_comm2[0])+'.'
       print '?'+str(t4),'dbs:'+str(t4)+'.'+str(prop_comm2[0]), ' ?'+str(prop_comm2[0])+'.'
    if prop_comm3 != [] and len(common)==3:
       print '?'+str(t3),'dbs:'+str(t3)+'.'+str(prop_comm3[0]), ' ?'+str(prop_comm3[0])+'.'
       print '?'+str(t4),'dbs:'+str(t4)+'.'+str(prop_comm3[0]), ' ?'+str(prop_comm3[0])+'.'
def quset (common, mylist):
        # Python code for BASE COMPONENTS IDENTIFICATION (Includes segmentation of sentences,
        tokenization, POS tagging, phrase-chunking)
        import nltk
        rawtext = open(plain_text_file).read()# Plain text file contains entire query set
```

```
sentences = nltk.sent_tokenize(rawtext) # NLTK default sentence segmenter
sentences = [nltk.word_tokenize(sent) for sent in sentences] # NLTK word tokenizer
sentences = [nltk.pos_tag(sent) for sent in sentences] # NLTK POS tagger
#Example: sentence = [("the", "DT"), ("little", "JJ"), ("yellow", "JJ"), ("dog", "NN"), ("barked","VBD"), ("at",
"IN"), ("the", "DT"), ("cat", "NN")] # a simple sentence with POS tags
#Simplified Part-of-Speech Tagset
#Tag    Meaning          Examples
#ADJ    adjective        new, good, high, special, big, local
#ADV    adverb           really, already, still, early, now
#CNJ    conjunction      and, or, but, if, while, although
#DET    determiner       the, a, some, most, every, no
#N      noun             year, home, costs, time, education
#NP     proper noun      Alison, Africa, April, Washington
#NUM    number           twenty-four, fourth, 1991, 14:24
#PRO    pronoun he, their, her, its, my, I, us
#P      preposition      on, of, at, with, by, into, under
#UH     interjection     ah, bang, ha, whee, hmpf, oops
#V      verb             is, has, get, do, make, see, run
#VD     past tense       said, took, told, made, asked
#VG     present participle making, going, playing, working
#VN     past participle  given, taken, begun, sung
#WH     wh determiner    who, which, when, what, where, how

#Define tag patterns to find NP-chunks; PP-Chunks (prepositional phrases chunks) ; terms/collocations etc
patterns1 = """
        NP: {<DT|PP\$>?<JJ>*<NN>}
        {<NNP>+}
        {<NN>+}
        {<DT>?<JJ>*<NN>}
"""“
patterns2 = """
        PP: {<DT|PP\$>?<JJ>*<NN>}
        {<NNP>+}
        {<NN>+}
"""“
patterns3 = """
        TP: {<DT|PP\$>?<JJ>*<NN>}
        {<NNP>+}
        {<NN>+}
"""“
NPChunker = nltk.RegexpParser(patterns1) # create a NP-chunk parser
PPChunker = nltk.RegexpParser(patterns2) # create a PP-chunk parser
TPChunker = nltk.RegexpParser(patterns3) # create a TP-chunk parser
result1 = NPChunker.parse(sentences) # parse the queries
result2 = PPChunker.parse(sentences) # parse the queries
result3 = TPChunker.parse(sentences) # parse the queries
print result1, result2, result3
```

Appendix 9: Query Sets and Results for Evaluation of English Wizzard, Easy Query and ELF (Reproduced from Bootra (2004))

	Query	ELF Response	EQ Response	EW Response
	where are the suppliers from Germany located	+	Sorry, I didn't understand that.	+
	show the names and complete address of the biscuit companies	+	Sorry, I didn't understand that. Please check your spelling or phrasing. -If you capitalize proper names, it will be easier for me to understand you.	INCORRECT: show employees, Discontinued 126 times (crosses with Order Details)
	at which company does Ian work	INCORRECT: work=>worker=> employee (crosses with Employee table)	Based on the information I've been given about this database, I can't answer: "At which companies Ian does works?". I haven't been given any information on companies.	INCORRECT: No rows returned.
	who handles the specialty items(Modify to: who supplies speciality items?)	+	INCORRECT: No appropriate choice	I'm not familiar with the word: handles
	show the domestic suppliers	+	Based on the information I've been given about this database, I can't answer: "How domestic are suppliers?". I haven't been given any information on domesticness.	I'm not familiar with the word: domestic
	show the New Orleans suppliers	+	INCORRECT: No answer because New Orleans is part of name, not whole name of company	+
	show the New England suppliers	+	INCORRECT: Same problem as New Orleans	I'm not familiar with the words: New England
	which company handles the specialty products	+	INCORRECT: No appropriate choice	I'm not familiar with the word: handles
	which companies have Product Managers	+	Based on the information I've been given about this database, I can't answer: "Which companies have Product Managers?". I haven't been given any information on companies.	+
	show the Product Managers	+	INCORRECT:	+
	show the orders by Leverling to Hanover Sq	+	I need to know how to interpret the name "Leverling to Hanover Sq"	You must specify 2 values to select a range of values.
	which products come in bottles	+	INCORRECT: No appropriate choice	I'm not familiar with the word: come
	What are the names of our Canadian customers?	+	Based on the information I've been given about this database, I can't answer: "Which customers have countries?". I haven't been given any information on countries.	I'm not familiar with the word: Canadian

	Give the name and location of suppliers from Germany.	+	Sorry, I didn't understand that.	INCORRECT: Gives address field only, 270 times
	Which are our Australian suppliers?	+	Based on the information I've been given about this database, I can't answer: "Which suppliers have countries?". I haven't been given any information on countries.	I'm not familiar with the word: Australian
	List the countries where suppliers are located, arranging the countries in alphabetical order.	INCORRECT:	Suppliers aren't there. Customers are there.	I'm not familiar with the word: arranging
	Products with names that start with "La".	+	INCORRECT: Wrong answer. Shows all containing, not starting with!	INCORRECT: Offers choice of Employee first or last name only
	Suppliers who are not located in Canada	+	Based on the information I've been given about this database, I can't answer	I'm confused by the word: Canada
	Find the products that have between 10 and 20 units in stock	+	Sorry, I didn't understand that.	+
	Records for customers who are located in Canada and whose names begin with the letter "M"	+	INCORRECT: Neither choice is correct.	I'm not familiar with the word: letter "M"
	Suppliers who are located in Canada and whose names begin with the letters A-N.	+	Sorry, I didn't understand that.	"letters" must be numeric.
	Suppliers who have a fax number	+	INCORRECT: show suppliers with or without faxes	+
	Show the employees hired between May 1, 1992 and June 1, 1993	+	+	+
	Employees who live in the United Kingdom or employees who live in Seattle	+	I don't understand the phrase: "t_or I list every employee that lives in Seattle".	I'm not familiar with the words: United Kingdom
	Orders placed before 1-Jan-93	+	Based on the information I've been given about this database, I can't answer: "Which orders are placed before 1-Jan-93?"."Which orders are placed?" doesn't depend on 1-Jan-93.	1' is not the expected type.
	Customers whose company names start with N-Z and who are located in either the United Kingdom or Paris	INCORRECT:	I don't understand the word "company" in the phrase "company start".	"N" must be numeric.
	Orders that were placed during the month of February 93	+	+	(' required after builtin function "month of".
	Find customers from Canada or the UK who have placed over 15 orders	+	I don't understand the phrase: "from UK".	15' is not the expected type.
	Suppliers who provide seafood products and who are from Singapore or Japan.	+	I don't understand the phrase: "from Singapore".	+
	Find the customers who ordered the "Chef Anton's Cajun Seasoning" product	+	+	+

	information on orders that were placed after 31-Mar-92, including the employee who made the sale and the customer who placed the order	+	Sorry, I didn't understand that. Please check your spelling or phrasing. If you capitalize proper names, it will be easier for me to understand you.	31' is not the expected type.
	What's the average price of all our products	+	I haven't been given any information on prices.	I'm not familiar with the word: price
	Give the name and id for each category.	+	INCORRECT: gives names but not Ids	INCORRECT: Offers only [Region ID] choices.
	List the customers	+	+	+
	Count the orders that have been placed for each seafood product	+	+	By "address", do you mean Customers, Employees, Suppliers?
	Show the ship date and order subtotals since March of 1994	+	I don't understand the phrase: "since March, 1994".	I cannot connect the table "Order Subtotals" to the other tables in your request.
	Display the subtotal and shipping date of all orders	+	Sorry, I didn't understand that.	I cannot provided (sic) both summary and detail information in the same request.
	List the suppliers in alphabetical order	+	I don't understand the phrase: "in orders".	+
	Find the total number of Northwind suppliers	+	Sorry, I didn't understand that.	I'm not familiar with the word: Northwind
	orders that were shipped today	+	+	What date does "today" refer to?
	orders that were shipped during the past ten years	+	+	+
	The number of orders that were shipped within the past 3100 days	+	+	What date does "last" refer to?
	Find the total value of orders that have been shipped to each country	+	I haven't been given any information on values.	I'm not familiar with the word: value
	Which products cost between $3 and $6?	+	Products don't have net costs. Line items have net costs.	I'm not familiar with the word: cost
	Give the order id, product name, product id, price, quantity, discount and extended price for each purchase	+	Sorry, I didn't understand that. Please check your spelling or phrasing. If you capitalize proper names, it will be easier for me to understand you.	I'm not familiar with the word: price
	Show catalog information for the active products.	INCORRECT: "catalog" and "active" are not defined	I haven't been given any information on catalogs.	I'm not familiar with the word: catalog
	the minimum price of all products in the Products table	+	I haven't been given any information on prices.	I'm not familiar with the word: price
	all records with the current date	+	I don't understand the word "current" in the phrase "current date".	INCORRECT: show no records, only a count
	What's the total number of orders we received this month	+	I haven't been given any information about people.	I'm not familiar with the word: received
	all employees who have birthdays today	+	+	INCORRECT: interprets this as "who was born today"!
	all employees who have birthdays on July 2	+	+	+
	All employee records that contain photos	INCORRECT:	I don't understand the word "employee" in the phrase "employee record".	INCORRECT: Wrong, interprets Photo as a True/False, which shows all records

	Find the total number of customers in Canada or the United Kingdom who have placed orders, and group them by country	+	INCORRECT: Neither choice is correct.	I'm not familiar with the words: United Kingdom
	Find the total value of orders shipped to each customer within each country	+	Sorry, I didn't understand that.	I'm not familiar with the word: value
	Which employee sold the most units of tofu?	INCORRECT:	Based on the information I've been given about this database, I can't answer: "Which employees sold products?".	I'm not familiar with the word: units
	Subtotal and customer for orders shipped between 10/1/91 and 12/31/91, sorting on the value	+	Sorry, I didn't understand that. Please check your spelling or phrasing. If you capitalize proper names, it will be easier for me to understand you.	I'm not familiar with the word: value
	photos of employees whose last names start with "B"	+	I haven't been given any information on photos.	+
	show photos of employees hired during 1991	+	I haven't been given any information on photos.	+
	which customers have ordered both Konbu and Filo Mix?	+	+	INCORRECT
	which products are more expensive than chai	+	Sorry, I didn't understand that.	I'm not familiar with the word: expensive
	how much does chai cost	+	Products don't have net costs. Line items have net costs.	I'm not familiar with the word: cost
	customers that ordered both chai and filo	+	Sorry, I didn't understand that. Please check your spelling or phrasing. If you capitalize proper names, it will be easier for me to understand you.	I'm not familiar with the word: filo
	how many products are there in each category	+	Products aren't in categories. Products are in orders.	Warning: due to a limitation of Microsoft Access the count displayed may include duplicates.
	which customers have ordered every meat/poultry product	+	+	INCORRECT
	which customers have never ordered seafood	+	+	I'm not familiar with the word: never
	which customers ordered Longlife tofu but not filo mix	+	+	INCORRECT
	which customers always use Federal Shipping	+	Sorry, I didn't understand that.	I'm not familiar with the words: always use
	which product costs the most	Sorry, unable to interpret the question.	I could not find a meaning for the noun "more".	I'm not familiar with the word: costs
	which customers have placed more orders than average	+	+	I'm confused by the word "orders".
	show the seafood products in reverse price order	+	I haven't been given any information about prices.	I'm not familiar with the words: reverse price
	customers that have ordered from both Ma Maison and Tokyo Traders	+	Suppliers have not had customers orderring from them. Employees have had customers orderring from them.	INCORRECT: No records; shows orders from any company which is named both Tokyo Traders and Ma Maison, which is pretty darned unlikely
	show company names of the suppliers that have more than 3 products	+	I don't understand the word "company" in the phrase "company name".	+

	which orders were neither shipped to Canada nor sent via Speedy Express	+	INCORRECT: Neither choice is correct.	I can't relate "ship" to a seach value.
	which orders were not both shipped to Canada and sent via Speedy Express	+	I don't understand the phrase: "via Speedy Express".	I'm confused by the word "ship".
	how many customers have ordered every meat/poultry product	+	+	Misinterprets as "how many customers ordered EACH meat product" we'd give them that, but then they answer that question wrong by counting each customer once for each meat order, inflating the numbers
	what percentage of customers have ordered every meat/poultry product	+	+	Warning: due to a limitation of Microsoft Access the count displayed may include duplicates. (Percentages inflated as in above query)
	which customers bought products from every category	+	Based on the information I've been given about this database, I can't answer	INCORRECT: just shows every order
	which customers ordered the fewest items	INCORRECT: shows who placed the fewest orders	Based on the information I've been given about this database, I can't answer:"Which items did customers order?". I haven't been given any information on items.	I'm not familiar with the words: fewest items
	show the names and complete address of the pear companies	+	I don't know what the companies are.	I'm not familiar with the word: pear
	which of the clients that purchased tofu have also purchased chai?	+	Based on the information I've been given about this database, I can't answer	Error in CreateEWQueryDef: join expression not supported
	Show the ship date and subtotals for all orders since March of 1991	+	I don't know how to connect subtotals to orders or ship dates, so I can't answer this question.	I cannot provided both summary and detail information in the same request.
	how many customers in each country have ordered tofu?	+	Based on the information I've been given about this database, I can't answer. I haven't been given any information on countries.	Warning: due to a limitation of Microsoft Access the count displayed may include duplicates. (As warned, it incorrectly includes duplicates)
	which customers exclusively use Federal Shipping	+	Based on the information I've been given about this database, I can't answer	I'm not familiar with the word: use
	which customers use Federal Shipping exclusively	+	Based on the information I've been given about this database, I can't answer	I'm not familiar with the word: use
	customers that work at 12 Orchestra Terrace	+	Based on the information I've been given about this database, I can't answer	INCORRECT: crosses customers with employees table
	customers in the t2f area	+	INCORRECT: No appropriate choice	I'm not familiar with the word: t2f
	count the orders for tofu versus those for chai	+	Sorry, I didn't understand that.	I'm not familiar with the word: versus
	graph the number of tofu or chai orders	+	I didn't understand the meaning of "number of order".	I'm not familiar with the word: graph
	graph the number of Seattle employees against London	+	Based on the information I've been given about this database, I can't answer	I'm not familiar with the word: against
	graph the sum of subtotals for seafood against beverages	+	Sorry, I didn't understand that.	I'm not familiar with the word: against

	graph the average subtotal for each category	+	Sorry, I didn't understand that.	Error processing query
	graph the sum of subtotals for tofu, chai and konbu	+	I haven't been given any information on subtotals.	Error processing query
	show the average number of products sold by each employee sales representative	INCORRECT	I don't understand the word "sales" in the phrase "sales representative".	Error processing query
	compare the average unit price showing employee and product	+	I don't understand the words "unit_price showing" in the phrase "unit_price showing employee".	INCORRECT: shows one number, not a crosstab
	which products were shipped by Federal in the last 5 years	+	INCORRECT: "Federal" is not an Employee's firstname!	I'm not familiar with the word: Federal
	list employees with home phones = (206) 555-8122 (206) 555-8122	+	INCORRECT: replaces '(206) 555-8122' with '(206)555-8122' which leads to no rows retrieved	I'm not familiar with the word: 206
	Find the total number of different customers in Canada or UK who have placed orders	+	Sorry, I didn't understand that.	Error processing query
	find the total number of DISTINCT customers in Canada or the United Kingdom who have placed orders, and group them by country	+	INCORRECT: Neither choice is correct.	I'm not familiar with the words: United Kingdom
	which suppliers have order dates that are newer than 600 months old	+	Sorry, I didn't understand that.	I'm not familiar with the words: newer
	show the difference between discount and unit price	+	Customers don't have unit prices. Products have unit prices.	I'm not familiar with the word: difference

Appendix 10: Some of the Databases Used in Nomenclature Analysis

A: HUMAN RESOURCE MANAGEMENT (Oracle, 2008)

HR_DB Schema
Object Type and Objects
Index: COUNTRY_C_ID_PK, DEPT_ID_PK, DEPT_LOCATION_IX, EMP_DEPARTMENT_IX, EMP_EMAIL_UK, EMP_EMP_ID_PK, EMP_JOB_IX, EMP_MANAGER_IX, EMP_NAME_IX, JHIST_DEPARTMENT_IX, JHIST_EMPLOYEE_IX, JHIST_EMP_ID_ST_DATE_PK, JHIST_JOB_IX, JOB_ID_PK, LOC_CITY_IX, LOC_COUNTRY_IX, LOC_ID_PK, LOC_STATE_PROVINCE_IX,REG_ID_PK
Procedure: ADD_JOB_HISTORY, SECURE_DML
Sequence: DEPARTMENTS_SEQ, EMPLOYEES_SEQ, LOCATIONS_SEQ
Tables: COUNTRIES, DEPARTMENTS, EMPLOYEES, JOBS, JOB_HISTORY, LOCATIONS, REGIONS
Trigger: SECURE_EMPLOYEES, UPDATE_JOB_HISTORY
View: EMP_DETAILS_VIEW
Table HR.COUNTRIES
COUNTRY_ID NOT NULL CHAR(2)
COUNTRY_NAME VARCHAR2(40)
REGION_ID NUMBER
Table HR.DEPARTMENTS
DEPARTMENT_ID NOT NULL NUMBER(4)
DEPARTMENT_NAME NOT NULL VARCHAR2(30)
MANAGER_ID NUMBER(6)
LOCATION_ID NUMBER(4)
Table HR.EMPLOYEES
EMPLOYEE_ID NOT NULL NUMBER(6)
FIRST_NAME VARCHAR2(20)
LAST_NAME NOT NULL VARCHAR2(25)
EMAIL NOT NULL VARCHAR2(20)
PHONE_NUMBER VARCHAR2(20)
HIRE_DATE NOT NULL DATE
JOB_ID NOT NULL VARCHAR2(10)
SALARY NUMBER(8,2)
COMMISSION_PCT NUMBER(2,2)
MANAGER_ID NUMBER(6)
DEPARTMENT_ID NUMBER(4)
Table HR.JOBS
JOB_ID NOT NULL VARCHAR2(10)
JOB_TITLE NOT NULL VARCHAR2(35)
MIN_SALARY NUMBER(6)
MAX_SALARY NUMBER(6)
Table HR.JOB_HISTORY
EMPLOYEE_ID NOT NULL NUMBER(6)
START_DATE NOT NULL DATE
END_DATE NOT NULL DATE
JOB_ID NOT NULL VARCHAR2(10)
DEPARTMENT_ID NUMBER(4)
Table HR.LOCATIONS
LOCATION_ID NOT NULL NUMBER(4)
STREET_ADDRESS VARCHAR2(40)
POSTAL_CODE VARCHAR2(12)
CITY NOT NULL VARCHAR2(30)

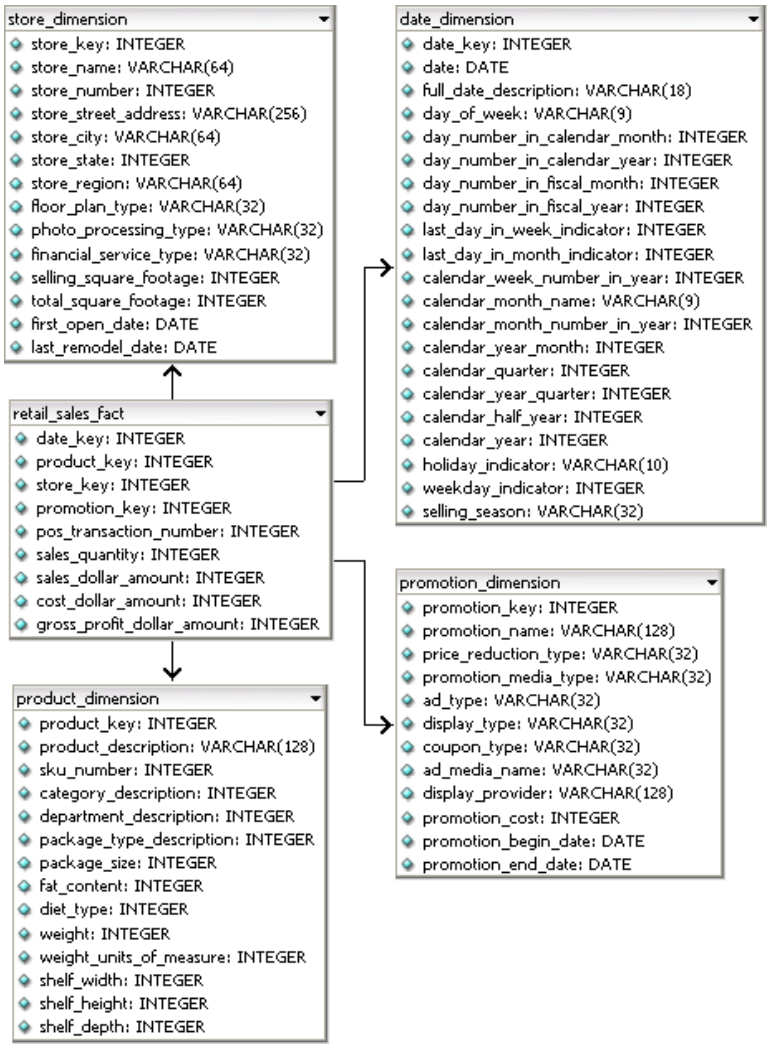
STATE_PROVINCE VARCHAR2(25)
COUNTRY_ID CHAR(2)
Table HR.REGIONS
REGION_ID NOT NULL NUMBER
REGION_NAME VARCHAR2(25)

B: ORDER ENTRY DB (Oracle, 2008)

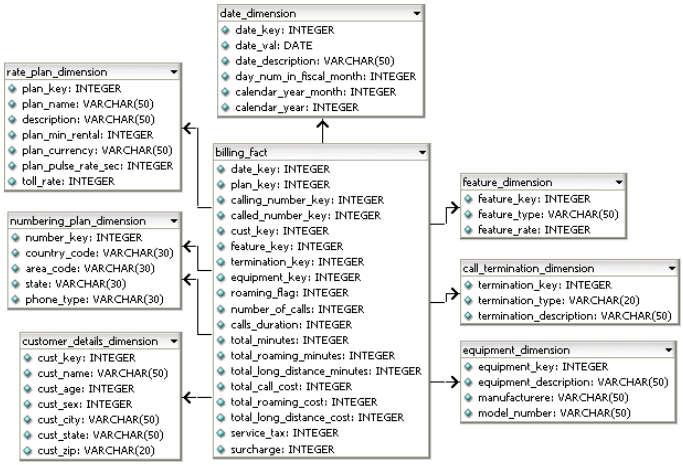
Order Entry Db-SCHEMA
Object Type and Objects
Index: CUSTOMERS_PK, CUST_ACCOUNT_MANAGER_IX, CUST_EMAIL_IX, CUST_LNAME_IX, CUST_UPPER_NAME_IX, INVENTORY_IX, INV_PRODUCT_IX, ITEM_ORDER_IX, ITEM_PRODUCT_IX, ORDER_ITEMS_PK, ORDER_ITEMS_
UK, ORDER_PK, ORD_CUSTOMER_IX, ORD_ORDER_DATE_IX, ORD_SALES_REP_IX, PRD_DESC_PK, PRODUCT_INFORMATION_PK, PROD_NAME_IX, PROD_SUPPLIER_IX, PROMO_ID_PK
Tables: CUSTOMERS, INVENTORIES, ORDERS, ORDER_ITEMS, PRODUCT_DESCRIPTIONS, PRODUCT_INFORMATION, WAREHOUSES
Triggers: INSERT_ORD_LINE, ORDERS_ITEMS_TRG, ORDERS_TRG
View: ACCOUNT_MANAGERS, BOMBAY_INVENTORY, CUSTOMERS_VIEW, DEPTVIEW, OC_CORPORATE_CUSTOMERS, OC_CUSTOMERS, OC_INVENTORIES, OC_ORDERS, OC_PRODUCT_INFORMATION, ORDERS_VIEW, PRODUCTS, PRODUCT_PRICES, SYDNEY_INVENTORY, TORONTO_INVENTORY
Table OE.CUSTOMERS
CUSTOMER_ID NOT NULL NUMBER(6)
CUST_FIRST_NAME NOT NULL VARCHAR2(20)
CUST_LAST_NAME NOT NULL VARCHAR2(20)
CUST_ADDRESS CUST_ADDRESS_TYP
PHONE_NUMBERS PHONE_LIST_TYP
NLS_LANGUAGE VARCHAR2(3)
NLS_TERRITORY VARCHAR2(30)
CREDIT_LIMIT NUMBER(9,2)
CUST_EMAIL VARCHAR2(30)
ACCOUNT_MGR_ID NUMBER(6)
CUST_GEO_LOCATION MDSYS.SDO_GEOMETRY
DATE_OF_BIRTH DATE
MARITAL_STATUS VARCHAR2(20)
GENDER VARCHAR2(1)
INCOME_LEVEL VARCHAR2(20)
Table OE.INVENTORIES
PRODUCT_ID NOT NULL NUMBER(6)
WAREHOUSE_ID NOT NULL NUMBER(3)
QUANTITY_ON_HAND NOT NULL NUMBER(8)
Table OE.ORDERS
ORDER_ID NOT NULL NUMBER(12)
ORDER_DATE NOT NULL TIMESTAMP(6) WITH LOCAL TIMEZONE
ORDER_MODE VARCHAR2(8)
CUSTOMER_ID NOT NULL NUMBER(6)
ORDER_STATUS NUMBER(2)
ORDER_TOTAL NUMBER(8,2)
SALES_REP_ID NUMBER(6)
PROMOTION_ID NUMBER(6
ORDER_STATUS NUMBER(2)
ORDER_TOTAL NUMBER(8,2)

SALES_REP_ID NUMBER(6)
PROMOTION_ID NUMBER(6)
Table OE.ORDER_ITEMS
ORDER_ID NOT NULL NUMBER(12)
Table OE.WAREHOUSES
WAREHOUSE_ID NOT NULL NUMBER(3)
WAREHOUSE_SPEC SYS.XMLTYPE
WAREHOUSE_NAME VARCHAR2(35)
LOCATION_ID NUMBER(4)

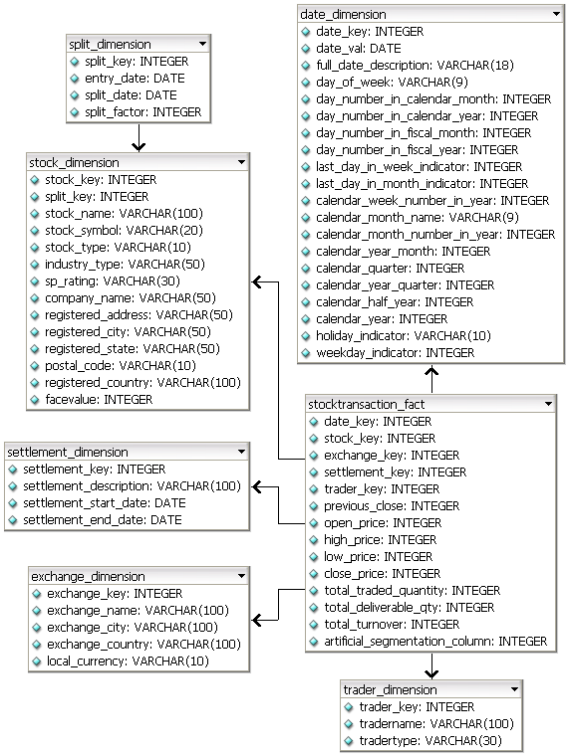
C: RETAIL MANAGEMENT (Vertica Systems, 2011)



D: PHONE COMPANY (Vertica Systems, 2011)



E: STOCK EXCHANGE (Vertica Systems, 2011)



Appendix 11: Foreign Key Attribute Processing

Examples from the Northwind Database and related Ontology (See section 3.73 and Table 3.47)

Sample Query 1: Show the 'names' and complete 'address' of the 'chai companies'?

Worked Example with Explanations

.....

The concepts to be picked are indicated.
'chai company' implies a 'company'>> that 'supplies' etc>> 'product named', 'chai'.
'Supplies' suggests use of <supplies Table> and 'product name' suggests <products table>.
Presence of 'names' suggests use of <~name>. Possible combinations include 'company name, product name etc, thus <ProductName> and < CompanyName>.
The concept 'address' implies the use of attribute <address>
Additionally we include <SupplierID> to uniquely identify supply company and <product name> to identify chai. To say we only want chai products, we include a filter for product name

==========

```
PREFIX chema: <http://www.owl-ontologies.com/NewNorthwind#>
SELECT ?SupplierID ?CompanyName ?Address ?ProductName
WHERE { ?suppliers db:SupplierID ?SupplierID.
        ?suppliers db:CompanyName ?CompanyName.
        ?suppliers db:Address ?Address.
         ?products db:ProductName ?ProductName
          FILTER(?ProductName = "chai") }
```

Results:

SupplierID	CompanyName	Address	ProductName
2	Charlotte Coopermaners	666663777	chai
1	Exotic Liquids	49 gilbert street	chai

>>>>>>>>>>>>>>>>

Sample Query 2: "Which products come in bottles?"

```
PREFIX chema: <http://www.owl-ontologies.com/NewNorthwind#>
SELECT DISTINCT ?ProductID ?ProductName
?Description ?CategoryID
WHERE { ?products db:ProductID ?ProductID.
        ?products db:ProductName?ProductName.
        ?products db:CategoryID ?CategoryID.
        ?categories db:CategoryId ?CategoryID.
        ?categories db:Description ?Description.
 FILTER( ?Description = "bottled") }
```

Appendix 12: Sample Results that Demonstrate Query Transformation

A. The NL QUERY

Give me the ids of order and the date goods were shipped pertaining customer 1. Give me also his full name.

B. THE CONCEPTS IN THE NL QUERY (Identified by OCM PROTOTYPE)

"ids of order" ➔ 'id' 'ord' ➔ Maps to OrderID
"date goods were shipped"➔ 'dat' 'good' 'ship' ➔ maps to shippedDate
"customer 1" ➔ 'cust' '1'; 'id'=implicit ➔ Maps to customerID = '1'
"full name" ➔ 'nam' which implies a relationship in 'orders' or 'customer' (implicit) table ➔ CompanyName, ShipName ('his' in the query refers to subject 'customer' therefore 'shipName is dropped and only CompanyName is considered). These are organized into triples as per OCM algorithm.

C. THE ONTOLOGY DERIVED FROM NORTHWIND DB (Derived using Mysql db, Datamaster, Protégé)

```
<db:orders rdf:about="&db;orders_Instance_1">
        <db:orders.CustomerID rdf:datatype="&xsd;int">1</db:orders.CustomerID>
        <db:orders.EmployeeID rdf:datatype="&xsd;int">1</db:orders.EmployeeID>
        <db:orders.Freight rdf:datatype="&xsd;string">mv nyayo</db:orders.Freight>
        <db:orders.OrderDate rdf:datatype="&xsd;date">2010-04-29</db:orders.OrderDate>
        <db:orders.OrderID rdf:datatype="&xsd;int">1</db:orders.OrderID>
        <db:orders.RequiredDate rdf:datatype="&xsd;date">2010-04-29</db:orders.RequiredDate>
        <db:orders.ShipAddress rdf:datatype="&xsd;string">italiano</db:orders.ShipAddress>
        <db:orders.ShipCity rdf:datatype="&xsd;string">rome</db:orders.ShipCity>
        <db:orders.ShipCountry rdf:datatype="&xsd;string">japan</db:orders.ShipCountry>
        <db:orders.ShipName rdf:datatype="&xsd;string">the wrecker</db:orders.ShipName>
        <db:orders.ShippedDate rdf:datatype="&xsd;date">2010-04-20</db:orders.ShippedDate>
        <db:orders.ShipPostalCode rdf:datatype="&xsd;string">777</db:orders.ShipPostalCode>
        <db:orders.ShipRegion rdf:datatype="&xsd;string">europe</db:orders.ShipRegion>
        <db:orders.ShipVia rdf:datatype="&xsd;string">mombasa</db:orders.ShipVia>
</db:orders>
```

Nb. The ontology triples are formed from the class-names, Attribute-names and instances

D. THE SPARQL QUERY (Generated by the OCM PROTOTYPE)

```
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
PREFIX moon: <http://www.owl-ontologies.com/NewNorthwind#>
SELECT  ?orderID ?CustomerID ?CompanyName ?shipDate
WHERE { ?orders db:OrderID ?OrderID.
    ?customers db:CustomerID ?CustomerID.
    ?orders db:CustomerID ?CustomerID.
        ?customers db:CompanyName ?CompanyName.
        ?orders db:OrderID ?orderID.
        ?orders db:ShippedDate ?shipDate
FILTER(?CustomerID = 1)}
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
```

E. THE ACTUAL RESULTS (Generated by the owl-reasoner)

orderID	CustomerID	CompanyName	shipDate
3	1	Alfreds Futterkiste	2010-05-24
1	1	Alfreds Futterkiste	2010-04-20