INTEGRATING DATABASES AND WORDNET FOR SEMANTIC SEARCH

By

ZUBAIRU ABUBAKAR HUSSAINI

DEPARTMENT OF MATHEMATICS, FACULTY OF SCIENCE,

AHMADU BELLO UNIVERSITY, ZARIA

SEPTEMBER, 2014

INTEGRATING DATABASES AND WORDNET FOR SEMANTIC SEARCH

By

Hussaini Abubakar ZUBAIRU, B.Sc (ABU) 2008

M.Sc/Scie/05380/2009-2010

A THESIS SUBMITTED TO THE SCHOOL OF POSTGRADUATE STUDIES, AHMADU BELLO UNIVERSITY, ZARIA

IN PARTIAL FULFILMENT FOR THE AWARD OF MASTER DEGREE IN COMPUTER SCIENCE

DEPARTMENT OF MATHEMATICS,
FACULTY OF SCIENCE
AHMADU BELLO UNIVERSITY, ZARIA
NIGERIA

SEPTEMBER, 2014

DECLARATION

I declare that the work in this thesis entitled Integrating Databases and WordNet for semantic search has been performed by me in the Department of Mathematics. The information derived from the literature has been duly acknowledged in the text and list of references provided. No part of this thesis was previously presented for another degree or diploma at this or any other Institution.

Zubairu, Abubakar Hussaini

Name of student	Signature	Date

CERTIFICATION

This thesis entitled INTEGRATING DATABASES AND WORDNET FOR SEMANTIC SEARCH by ZUBAIRU, Abubakar Hussaini meets the regulation governing the award of the degree of M.Sc Computer Science of Ahmadu Bello University, and is approved for its' contribution to knowledge and literary presentation.

Chairman, Supervisory Committee Date

Member, Supervisory Committee Date

Head of Department Date

Date

Dean, Postgraduate School

DEDICATION

This thesis is dedicated to all the kindhearted people in the world.

ACKNOWLEDGEMENT

In the name of Allah, the most merciful. All gratitude is due to Him for giving me this wonderful opportunity to live up to this time and for bestow this honor on me. I exalt and glorify Allah-the Sustainer. I acknowledge first the effort of my patient supervisors; Dr. M. B. Hammawa (Chairman, Supervisory Committee) and Dr. A. A. Obiniyi (Member, Supervisory Committee) for sacrificing their time to see me through the completion of this thesis.

I also acknowledge the effort of the entire lecturers of the department of mathematics for their academic and intellectual support towards the success of this thesis. Special thanks go to the post graduate Coordinator - Dr. A.M. Ibrahim. I acknowledge with gratitude the effort of my parents, all my siblings for standing by me and with me in the course of my academic sojourn. A special thanks to my Uncle, Chief (Dr.) J.O. Omuya for his financial and moral support since the inception of this programme.

I cannot forget mentioning some of my friends/course mates whose company I enjoyed so much throughout this programme; Arc. Abdulwahab Etudaiye has been a very wonderful roommate, Sunday Adepoju is more than a friend, Bolatito Yusuf was instrumental to the completion of my course work and to my entire class members, I appreciate the quality time spent with you as course mates.

A special acknowledgment also goes to Mr. Donfack Kana for his patience and tolerant for my constant presence in his house in the course of this research. I am grateful Sir.

Finally, May the almighty Allah in His infinite mercy reward and bless whoever has contributed in one way or the other to the success of this work.

ABSTRACT

Semantic integration of heterogeneous databases requires that user queries are understood semantically and can be related to the real contents of underlying databases. The state of art on data integration is focusing on using ontologies which is described as a formal and explicit conceptualization, as a semantic foundation for data integration. Unlike the previous research that considered the databases to have a well structured ontologies, this thesis employs WordNet lexical database to mine for synonyms from WordNet synsets into cluster. A query reformulation algorithm is proposed to reformulate the user query based on the retrieved synonyms and then lookup for all the queries in the database(s). The implementation turned out to be effective in practical applications, with a particular set of data, as the system is able to identify terms correctly based on synset, while previous methods relying only on available terms in the WordNet, this thesis can handle an unknown term by the WordNet by inserting new term into the existing synset in the WordNet database. Though, the evaluation results show that the proposed approach improves recall, but the system incurs high execution time as the number of table increases.

Table of Contents

LIST OF FIGURES

LIST OF TABLES

CHAPTER ONE

GENERAL INTRODUCTION

1.1 INTRODUCTION

The technological advancement has lead to tremendous growth of data in the different databases and information sources available on the web. The web is the enormous sources of information ever conceived by man. There are different business data in a particular domain hosted on the web that can help users or organizations to make decisions in case they need to make choices. Hence, it is necessary and crucial that this information source is complete, precise and can be acquired on time (Abdul-Kareem and Hajmoosaei, 2007).

However, each of the information sources on the web is designed and use differently according to the requirements of the organization. This independency nature of individual databases causes heterogeneity in the databases. As long as the heterogeneity exists, access to all information available remains limited since information is stored separately without easy means to integrate them from different sources. The information society, however, requires a complete and easy access to all available information. These necessitate the need for effective methods to resolve heterogeneity and combine data from different data sources.

To retrieve, access and use information from the different data sources give rise to the data to be integrated because of heterogeneity within and among the various data sources (databases), opined (Kashyap and Sheth, 2006).

Data integration system is aimed at providing users with easy and efficient access to information from different data sources through common interface. Integrating data from distributed, heterogeneous, autonomous data sources is a fundamental requirement in many databases application domains (Amshakala and Nedunchezhian, 2011). A data integrated system should allow users to concentrate on what information is needed without the need to offer detail information on how to retrieved the require information.

In recent years, there has been considerable interest in integrated database, a system which attempt to logically integrate a number of independent distributed databases while allowing the local databases to maintain complete control of their operations. Thus, an integrated system is a distributed database system in which each site maintains complete autonomy. Database integration has been recognized as the solution to disparate data sources. It deals with the data transparency problem of distributed systems. That means it has to make users think that they are accessing a single information system with homogeneous data structures. But actually the data is physically distributed over heterogeneous data sources. An important requirement to achieve an integrated database is to be able to identify semantically equivalent data items in component databases.

Semantic heterogeneity is the major source of problems in integrated database design. These conflicts result from the use of different structures for the same information and the use of different names for the same entity in the same domain.

Ontology have been suggested as a cornerstone to solve the problem of semantic heterogeneity in database integration. The basis is that an ontology can offers a shared common understanding of the application domain. It provide the meaning of terms and their relationships that facilitates communication by providing precise notions that can be used to compose messages (queries, statements) about the domain (Li and Chow, 2009).

Previous research regarding the use of ontology to address the semantic problems was limited to well defined, well structured and domain specific ontology (Guarino and Giaretta, 1995;Gulati and Sharma, 2010; Ghawi and Cullot, 2009). In addition, most of these works are mere proposal, hand-coded and not implemented.

Hand-coded ontology has a low coverage of named entities, domain specific, and automatic building ontology is time-consuming and daunting task. Making uses of existing ontology and augmenting it with new term(s) could be of great benefit and time saving.

In this direction, this research proposes the use of WordNet (existing ontology) to serve as a controlled vocabulary and semantic similarity comparison mechanism between words. If semantic relation like synonyms could be incorporated in the search, then semantically related terms may be retrieved and could be presented to the user, even if it is not specifically present in the query. By replacing the query terms with a mechanism which can identify semantically similar terms, challenge posed by the use of different terms to define the same concept may be handled.

1. 2 Background of the study

Data integration is a research discipline that studies the mechanisms for a seamless access to autonomous and heterogeneous information sources (Amshakala and Nedunchezhian, 2011). Data integration and information exchangeable systems request a uniform interface that allows transfer and process that share data across multiple distributed systems. Current data integration systems are built upon the traditional database systems in which data are modeled in one of traditional data models.

Because each data source is modeled in different data models and available in different form and formats, integration problems have emerged. These different data models have result in to number of databases, which in turn give birth to incompatibility among themselves. This has led to the heterogeneity of the existing database systems.

There are different types of heterogeneity such as schematic, semantic and syntactic existing among the data sources that are to be integrated. Schematic emanated from different structures of sources schema, semantic occurs when different sources contain the same concept with different name and heterogeneity resulted from different language used for modeling the different sources. These heterogeneity issues necessitate data integration in order to enhance better decision making and function of associated information system. Thus, there is need to create a frame work that reconcile different databases system.

Data integration has been recognized as the solution to disparate data sources. It deals with the data transparency problem of distributed systems. That means it has to make users think that they are accessing a single information system with homogeneous data structures. But actually the data may be heterogeneous, autonomous and geographically distributed sources.

The process of heterogeneous database integration may be defined as the creation of a single uniform query interface to data that are collected and stored in multiple, heterogeneous databases, Raji (2010). The interoperability between databases should be supported without modifying the database or losing their autonomy, and in a way that is relatively transparent to user and the application. The state of the art on data integration is focusing on using ontologies, which is a formal and explicit specification of a shared conceptualization, Gruber (2009), to provide semantic support for data integration. Ontologies have been proven to be useful to capture the semantic content of data sources and to unify the semantic relationships between heterogeneous structures (Necib and Freytag, 2005).

With this vision, this thesis proposes a data integration system, using an ontology approach. To this end, the research attempt to use WordNet (an existing ontology) and adapt it with associated databases to overcome semantic heterogeneity with respect to ontology approach. The main focus is on the part of semantics related to the meaning of the terms used as identifiers in the various databases.

1.3 Heterogeneous Databases

Heterogeneous databases can be viewed as separate autonomous databases, independently created for unique purposes, with differences in both abstract and data models, which represent the underlying paradigm of the database (for example, extensible markup language, relational, object-oriented), and database schema, which represent the developer's conceptual model of the data structures and their interrelationships. It is important to recognize that the content of the databases must be considered "semantically" equivalent in a very general sense to be considered a heterogeneous database problem. Semantically similar databases reportedly contain the same “type” of information or constructs. Although heterogeneous databases broadly contain the same types of semantic content, this content can be represented in many different ways, often resulting in semantic differences that are difficult to identify and resolve. Thus, there is need to integrate the syntactic different but semantically the same object.

1.4 Problem Definition and Motivation

In an integrated system of a particular domain, the same entity may be stored with different names in different databases. Retrieving information from an integrated system, matches the user’s query with available databases, through syntax; this is due to the fact that the traditional method of processing query depends on syntax. It is common, however, that a system does not find an exact match, but there might be value(s) in the available database(s) that though syntactically different, but semantically equivalent to the user’s term that capture his or her intention. This leads to an interesting question; is it possible to return results when there are terms in the columns which are “related” to the term being search for?

For example, a synonym of a term found in the system, may be used in the query (TV and television, price and rate). Retrieving record from an integrated system often fail, because of syntactic and semantic differences. Consider the table 1.1 showing product information stored at a single data source.

Table 1.1: Product_table

```
mysql> use mysqldb;
Database changed
mysql> select * from item;
+------+-------------------+-------+
| id   | product_name      | price |
+------+-------------------+-------+
| 1    | cellular phone    | 5000  |
| 2    | television        | 20000 |
| 3    | sound system      | 5000  |
| 4    | personal computer | 30000 |
+------+-------------------+-------+
4 rows in set (0.06 sec)
```

A user query for a television using "TV" as search from table 1.1 return null, because, the tradition method of query processing depends on syntax, though, "TV" is implied by other term ("Television") which should be considered in processing the query.

Retrieving records that are semantically related to a user's query will yield better results in an integrated system that may have different databases of the same domain, describing the same entity in different forms. If semantic relation like synonyms could be included in the search, then semantically related terms may be retrieved and presented to the user. Using of synonyms terms in different databases is difficult for Database Management System (DBMS) to handle without semantic relation like ontology. By using WordNet which has the capability to identify semantically similar terms, challenge posed by the use of different terms to define the same concept may be handled. This form the basis of the thesis methodology.

1.5 Research aim and objectives

The major objective of this thesis is to design a platform in which the user needs not to formulate a query according to document structure of a particular database source in order to query several sources of information and produce and integrated view of their data. Modified version of phrase extraction phase of lingo algorithm is used to retrieve and cluster terms that are synonymous to the user query, and the propose query reformulation algorithm to reformulate the query based on the available synonyms.

The main objectives are;

i. To develop a model for heterogeneous database integration.

ii. To implement the designed model.

iii. To Test the designed model.

1.6 Research Methodology

The thesis approach to data integration is based on query reformulation; given a search term, the system need to find related terms and search for those related terms from WordNet. Modified phrase extraction phase of Lingo algorithm is used to retrieve and cluster the synonyms. The proposed query reformulation algorithm, exploit the retrieve synonyms to reformulate the user query based on the available synonyms and then lookup for all the queries in the database(s).

The step by step methodologies adopted to perform semantic integration of heterogeneous database are:

i. Study of heterogeneous database systems.

ii. Conduct thorough literature review to understand various semantic heterogeneity conflicts and semantic integration techniques.

iii. Study WordNet and its application in resolving conflicts in heterogeneous database

iv. Design and implementation of the integrated system, using WordNet as a controlled vocabulary.

1.7 Justification of the study

Based on the available literature, there is no evidence or report in literature that has directly update the WordNet database, therefore, this thesis present the possibility of extending WordNet database by plug-in new terms.

CHAPTER TWO

LITERATURE REVIEW

2.1 INTRODUCTION

Integration of data based on the semantics has been a long-standing challenge for the database community. It has received steady attention over the past two decades, and has now become a prominent area of database research (Wang and Xu, 2008). Integration of heterogeneous databases aims at accessing data from different databases in a consistent manner. The data access deals with retrieving the semantically related data present in different types of databases. (Kavitha *et al*., 2011).

Data integration and information exchangeable systems request a uniform interface that allows transfer and process to share data across multiple distributed systems (Rakesh, 2011). Several solutions have been proposed in literature to address the integration of heterogeneous databases based on semantic detection and reconciliation. Different methods tackle different aspects of the problem.

The following studies have been reviewed in the course of this research:

The research by (Kashyap and Sheth, 2006), proposed a model for capturing semantic similarities between objects (called semantic proximity) with which they represent semantic similarity between objects. Semantic similarity between objects is represented as $SemPro(O_1,O_2)=\{Context, Abstraction, (D_1,D_2),(S_1,S_2)\}$. First component, Context represents the context in which objects are being compared, second component Abstraction is the mapping used to relate the domains of O_1 and O_2, the third component enumerates the domain definitions of O_1 and O_2, and finally the fourth component enumerates the state of objects O_1 and O_2. The authors also provide taxonomy of schematic conflicts and introduced the concept of schema correspondences to capture structural similarities between objects. It is not clear in their work how the semantic similarity between two objects is inferred (manually, semi-automatic or automatic), and how the contexts are defined and built. Also the SemPro() function deals with two objects at a time. This seems impractical when dealing with a very large number of objects.

(Ma'ruf, 2010), in his work titled "Achieving heterogeneous database integration through database schema analysis". His works focused on alignment of heterogeneous database schema to achieve interoperability. The Author assumed the existence of both relational databases, and produces a set of relationships between the two databases. Any new construct that is unrelated to the mapping in the existing schema matching is considered to be out of scope of the merging. However, in an ideal database integration system, they could be databases in their thousands to be integrated, some which are unforeseen; hence this approach will not be feasible in an ideal system.

Suwanmanee *et al*., (2009) in their work titled "Wrapping and integrating Heterogeneous Database with OWL", they proposed an architecture that adopting mediator approach, by providing local ontologies for various data sources and global mediator that allows interoperability of the local sources. This method is very difficult to achieve, and thus unlikely. Aside the impracticability of this method, the Authors, did not provide the mechanism to query the system, and suggest that as a future work. Furthermore, it is very difficult to provide the translation to a global ontology.

Zhou *et al*., (2011) proposed an approach for database integration by semi-automatically building ontology from relational database schema and using wordNet to extend the ontology. It was not clear in their work how they integrate wordNet into the relational databases and how the semantic conflicts will be resolved. Furthermore, there is no provision to handle any unknown concept by the WordNet. This thesis is actually an improvement on Zhou *et al*., (2011), but the databases are considered in their original form without any modification.

Zhang and Li (2011), in their work proposed a method for automatic generation of ontology from relational database resources to improve the efficiency of ontology construction. They put forward some construction rules of ontology elements based on relational database, which are used to generate ontology web language, ontology concepts and properties. However, the correctness and feasibility of the system was a serious issue, as there is great disparity in quality between the ontology generated by this system and manual domain ontology.

The list of reference outlined is far from being exhaustive as there are numerous works that has been done in respect to data integrated system. However, many of the work available in the literature (actually all the one found during the research) suffer from at least one among the following deficiencies:

i. Scalability: Most of the existing works in the literature are not easily scalable, once the system has been put in use; it is difficult to introduce a new database to the system.
ii. Impracticability: Most of the system are actually a proposal and with one relational database and not feasible in real life, when there are thousands of databases to be integrated.

iii. Existing integration of data based on ontology cannot be reused easily, and have to be redeveloped from scratch.

iv. They are not implemented. Majority of the available works in the literature are mere proposal and never implemented to ascertain the feasibility of the proposal.

After going through the research done in the area of semantic heterogeneity and schema integration in heterogeneous databases systems, various semantic conflicts and challenge in schema integration have been understood. As an attempt to contribute in this direction, this thesis proposed an approach that uses WordNet to detect semantically related terms based on the user query and to search the available database using the available synonyms from the wordNet.

Unlike the previous work done in this area, the aim is not to speed up query processing but to provide users with additional meaningful answers.

2.2 Data Integration

Data integration deals with the data transparency problem of distributed systems. That means it has to make the users think they are accessing a single information system with homogeneous data structures. But actually the data is physically distributed over heterogeneous data sources. In this way, there is need for a semantic knowledge to reconcile the differences.

Data integration has evolved from structural to semantic integration. Traditional integration is based on relational and functional data model that integrate with one single global schema (Li, 2009). With the development of Internet and web applications, mediator and agent systems have become popular in the data integration. However, providing explicit and precise semantics is the critical problem of data integration. In the requirement of integration with heterogeneous data sources, the one single global schema and mediator or agent system is not possible to fulfill the needs.

Therefore, the ontology is introduced for providing explicit, formal, conceptualized definition of the data source. Compared with the former integration approach, the ontology based data integration reduced the semantic ambiguous by providing shared understanding. For example, the same syntax may have different meaning in two databases, but in the ontology, it will give more complete definition to each syntax to avoid the ambiguousness.

Though, ontology is promising in reconciling the heterogeneity issue on databases, it drawback is that, one ontology approach is only suitable for the integration within a single domain. It requires all the data sources mapping to the common ontology. As the multi-domain data integration the single ontology will have limited abilities to provide precise meaning of the data. Therefore, multi-ontology approach is introduced. Multi-ontology approach divides ontologies into a hierarchy. The lower, middle and upper ontology, the top level (upper) ontology is a highly abstract data model that provides meta-concept and meta-data for the lower ontology for easy reconciliation of heterogeneity.

2.3 Heterogeneity in Database System

The system heterogeneity concerns the differences in the low level architectural platforms, local component databases, such as hardware configuration, operating systems and communication facilities. At these level, different systems techniques is used by different DBMSs such as query processing strategy, concurrency control mechanism and transaction management facilities. The physical heterogeneity/syntactic heterogeneity refer to the differences in the data representation. It may concern the encoding of same data values into integers or real of different types, or, it may concern different implementation of the data model, for example tables in different relational systems. Finally, the data models themselves can differ. Data of interest may be in a relational database, extensible markup language (XML) database, and in an object-oriented database. The

Schematic and semantic heterogeneity refers to the discrepancies between data names, values, and the conceptual structures. Data heterogeneity refers to the scenario whereby difference sources might have different data models, require different access methods, and even be hosted by different hardware and operating system, Raji (2010). Schematic heterogeneity means, all component databases have different data model structure where equivalent and related data concepts are present and having conflicting structural representation. It can also be said that all local component databases model having same real world concept but structured in a different manner. Semantic heterogeneity refers to differences or similarities in the meaning of local data. For example, two schema elements in two local data sources can have the same intended meaning, but different names. Thus, during integration, it should be realized that these two elements actually refer to the same concept. Alternatively, two schema elements in two data sources might be named identically, while their intended meanings are incompatible. Hence, these elements should be treated as different things during integration. It is vital to compare local schema contents to detect type of schematic conflicts that may exist among these local schemas. Schematic heterogeneity arises when information that is represented as data under one schema, is represented as a value within the schema in another, for example, as relation or class names. Semantic relationships between the local databases determine how local classes should be integrated in the global schema. Various types of semantic relationships are possible between different schema objects. Schema objects in different databases are either semantically related or semantically incompatible. Semantically related schema objects represent the same real world concept. Semantically incompatible objects, on the other hand, are objects that cannot be semantically related and therefore cannot be merged into one component when building an integrated database, (Sugandha, 2010).

The main concern among the various heterogeneities highlighted is the semantic heterogeneity. That is same real word object that may be stored with different name in different database of the same domain.

2.4 Data Integration Approaches

There are different approaches to database integration that are well study in the literature, these are: Database translation, Multi-database language system, Federated database system, Data warehouse, and global Schema integration.

The database translation approach is a point-to-point solution based on direct data mappings between pairs of information systems. The mappings are used to resolve data discrepancies among the systems. This approach is appropriate when the number of participants in the interoperability environment is small. The number of data translators grows with the square of the number of components in the integrated system. The multi-database approach consists of a loosely coupled collection of databases in which a common query language (often SQL-like) is used to access the contents of the participating databases, Ghawi and Cullot (2009). The common language provides constructs that permit queries involving several databases at the same time, as well as operators for users to perform resolution of semantic conflicts. A multi-database language lacks the transparency for locating information, because users have to know a priori where the data are stored. In this approach, in contrast to the distributed and federated systems, no predefined or partial global schema is used. Instead, the users must discover and understand the semantics of other information systems. This approach lacks distribution and location transparency for users. A federation contains a federated schema that incorporates the data exported by one or more remote information systems (the federation participants/components), (Raji, 2010). The level of integration depends on how tightly or loosely coupled the databases are. A tightly-coupled architecture provides a stable interaction through the definition of a single federated schema controlled by the federation administrators. As with the global schema, whenever there are changes in the export schema of a tightly-coupled architecture, integration needs to be redone. A loosely- coupled architecture is a flexible approach that achieves interoperability by defining multiple views over databases. In this architecture, it is the user who has the control of the federation. A shortcoming of the loosely-coupled architecture is the assumption that users know exactly what they are looking for and what each database contains. A data warehouse approach of data integration, data originating from different sources are submitted to a process called ETL (Extraction, Transformation and Loading) and then stored into a new database with a single and usually denormalized schema. This final database is often structured to store various aggregations of the sources' data to speedup query processing. From an architectural point of view, Data warehousing can be seen as a tightly coupled approach because the integrated data reside in a single place at query time. In this approach, data originating from different sources are submitted to a process called ETL (Extraction, Transformation and Loading) and then stored into a new database with a single and usually denormalized schema. This final database is often structured to store various aggregations of the sources' data to speedup query processing, (Bernstein, 2008). From an architectural perspective, data warehousing can be seen as a tightly coupled approach because the integrated data reside in a single place at query time. This approach constrains the autonomy of databases and impractical as many databases are interconnected and databases update their local data. The inadequacy of the aforementioned approaches to tackle the semantic heterogeneity issues, gave rise to an alternative better means, hence the quest for an ontology approach to tackle heterogeneity in the databases, for effective data integration.

2.5 Ontology Based Approach for Database Integration

Data integration system aim at providing users with easy and efficient access to information from different data sources through common interface. As data to be integrated become large, the discrepancies in the concept interpretation, known as semantic heterogeneity have been major obstacles to achieve effective data integration (Amshakalla and Nedunchezhian, 2011). The move towards machine-understandable web resources has stress the need for semantic integration of various data sources in a particular domain (Zhixiao and Niu, 2012).

In recent time, focus has been on using ontologies for managing data in database and information systems. Ontology is formal and explicit specifications of a shared conceptualization, Gruber (2009), thus, provide supports for understanding the meaning of data. They are generally used in information integration systems to overcome problems caused by the heterogeneity of data among the distributed sources. In this thesis, ontology based integration of heterogeneous data is followed to create a single query interface for different databases in the same domain to improve the recall of query.

The research make use of an existing ontology (WordNet) within few relational databases from different platforms and present an approach of integrating databases using semantic knowledge from WordNet to reformulate a user query in such way that the query answer deals with the issue of finding all the relevant documents in the data collection for a given user's query, as stated in a natural language, includes those record that matches the user query semantically, even though, there is syntax discrepancy. To this end, modified phrase extraction phase of Lingo algorithm is used to retrieve and cluster the synonyms. The propose query reformulation algorithm, exploit the retrieve synonyms to reformulate the user query based on the available synonyms and then lookup for all the queries in the database(s).

Many researches have been done in database integration using ontology approach. But virtually all these works considered the databases involves to having a well defined (or) well structured ontology. However, the world is dynamic; the feasibility of structure a perfect ontology including the relations of all the resources in the various databases is highly impracticable. In Semantic Web, a class is view as a type of resource. This class possesses some properties and it can be a subclass to another class. Thus, if all these relationship is not well defined, it cannot be matched by the ontology based search engines for a web resource which is not defined as a class in the structured ontologies. In order to overcome the difficult inherent in designing ontology, this research proposed an approach of database integration that consider the databases as they are and make uses of WordNet to extracts potential relations among the databases by using the knowledge such as the synonyms set in the WordNet database. Unlike, previous research, effort is made to incorporate new terms into wordnet synset. The popular architecture for ontology-based integration is shown in the figure 2.1.

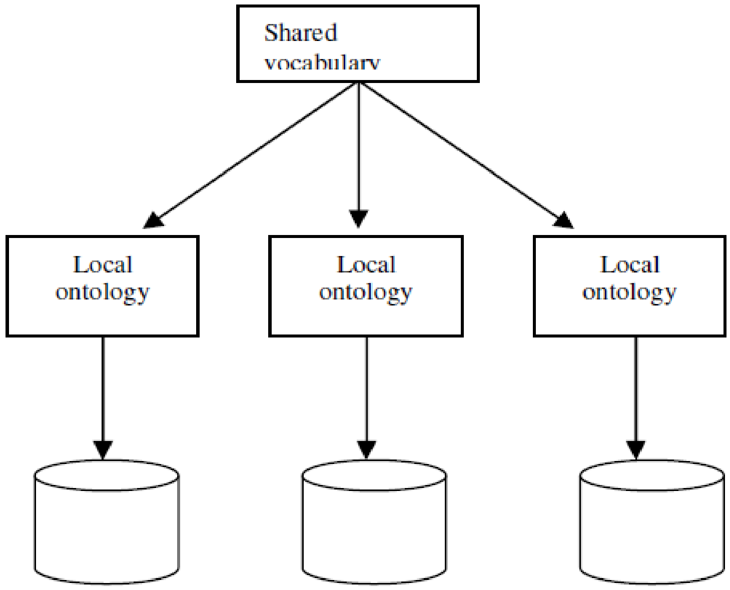


Figure 2.1: Architecture of an ontology-based integrated system(Akilandeswari and Subhashini, 2011)

2.6 Wordnet Overview

WordNet is a software system that has been developed by Princeton University that aims to be a lexical database. WordNet is basically an online lexical reference system whose design is inspired by current psycholinguistic theories of human lexical memory. English nouns, verbs, adjectives and adverbs are organized into synonym sets, technically called synset, each representing one underlying lexical concept. WordNet ontology is a useful resource to develop an ontology-based information retrieval system (Sameh and Amar, 2010). One use of WordNet is to preserve the distinction between the level of semantics and the level of concepts by enforcing different lexical relations for words and for concepts (Necib and Freytag, 2005). Relations between concepts and words in WordNet are made explicit and are labeled so that users can select a specific relation to guide them from one concept to the other. The synsets are interlinked by means of conceptual-semantic and lexical relations. The WordNet database can be read by the computer in an efficient manner and it would be useful to the user. The WorldNet will be incorporated into the propose system to enhance the performance of the propose system and support the scalability of the system. The decision to use WorldNet is motivated by the fact that it already defines relations between words. Relations between concepts and words in WordNet are made explicit and are labeled so that users can select a specific relation to guide them from one concept to the next. Another use of WordNet is that words express concepts in WordNet and the lexicon is constrained by the kinds of concepts that are available to human beings by virtue of perception of, and interaction with the world around us Bagiwa (2011).

However, despite the importance of WordNet for research and application development in natural language processing, WordNet does not cover all vocabularies and some terms that are from human understanding synonymous are not group in the same synset in the WordNet. Previous researches that make use of WordNet make little or no effort to insert new terms or words into the synset to cover more general vocabularies. This research is going to make effort in this direction.

2.7 The Lingo Algorithm

(Osinski and Weiss, 2005) proposed an algorithm called the Lingo algorithm that uses frequent phrases to identify candidate cluster labels, and then assigns snippet to these labels. This algorithm is extended by adding semantic identification to the frequent phrases extraction phase. The extension was achieved by finding the synonyms of the user entered key word in the WordNet database, and adding the synonyms to the user entered key word that comprises of the synset of the given key word. The detection of synonyms assists in clustering together terms that contain different but synonyms word. For more detail on Lingo, refer to (Osinski and Weiss, 2005) and (Osinski *et al*., 2004).

2.7.1 Overview of Lingo Algorithm Phases

Briefly discussed below are the phases involves in the Ling algorithm. For more detail, check (Osinski *et al*., 2004).

1. **Preprocessing**: Stemming and stop word are remove at these phase to improve the quality of the input snippet and frequent phrase detection and cluster labeling.
2. **Frequent Phrase Extraction**: This phase discover phrases and single terms that are recurring terms that can possibly explain the meaning behind concept. This phase is where this thesis actually modified, thus it give birth to the propose algorithm.
3. **Cluster Label Induction**: In these phases, Lingo algorithm use singular Value Decomposition to identify abstract semantic concept that binds the document together.
4. **Cluster Content Discovery**: In this phase, input snippets are matched to the cluster with the label selected in the previous phase.
5. **Final Cluster Formation**: The Cluster discovered in the preceding phase is sorted and the one with top scoring clusters are display.

Noteworthy is the fact that all five phases of LINGO are independent and easily separable. This allows to manipulate the quality and resource requirements of the algorithm by providing alternative implementations of some of its components(Osinski *et al*., 2004). This aspect of the algorithm makes it easier for modification to the second phase of the algorithm to suit the approach of the thesis.

CHAPTER THREE

DESIGN OF AN INTEGRATED DATABASE SYSTEM

3.1 INTRODUCTION

This chapter discusses the approach of the thesis to design and implement an integrated system, the detailed structure of the proposed architecture is discussed and the components of the proposed integrated system are explained in details. The architecture of the proposed system is shown in figure 3.1. It is based on the mediator-wrapper approach described in various literatures. This section also discussed the tools, and the algorithms the system implemented.

3.2 Tools Used in the System

i. Netbeans IDE: This is an integrated Development Environment (IDE) for java application, though other programming languages like PHP, C/C++ can also make use of the IDE. The netbeans was used to implement the processes in the application.

ii. **JWNL:** Java WordNet Library (JWNL), a Java application programming interface (API) to access WordNet database to retrieve terms that are semantically similar (synonyms) to the keyword term entered by the user. This API is simple and compatible with different versions of WordNet database files. This functionality can only be activated by installing the full version of WordNet database and java archive (JAR) that house the JWNL. The application receives a user query, which will be assumed to be noun. Using the JWNL class- PointerUtil to access the WordNet and generate a hierarchical hyponym tree for the first synset of noun term. Traveling down the tree, the system record the information of each synset, such as synset ID, gloss, and synonyms contained in the synset. This aspect is actually visible to the application.

iii. **WordNet:** is a software system that has been developed by Princeton University that aims to be a lexical database. WordNet is basically an online lexical reference system whose design is inspired by current psycholinguistic theories of human lexical memory. English nouns, verbs, adjectives and adverbs are organized into synonym sets, technically called synset, each representing one underlying lexical concept. The synsets are interlinked by means of conceptual-semantic and lexical relations.

iv. **JDBC:** is Java Database Connection that provides a uniform interface and high level of location transparency for retrieval of data from the local component databases. The JDBC make use of the database(s) driver in order to access their information.

v. **JAVA:** is an object oriented programming language that is used in designing the user interface.

3.3 Architecture of the Proposed System

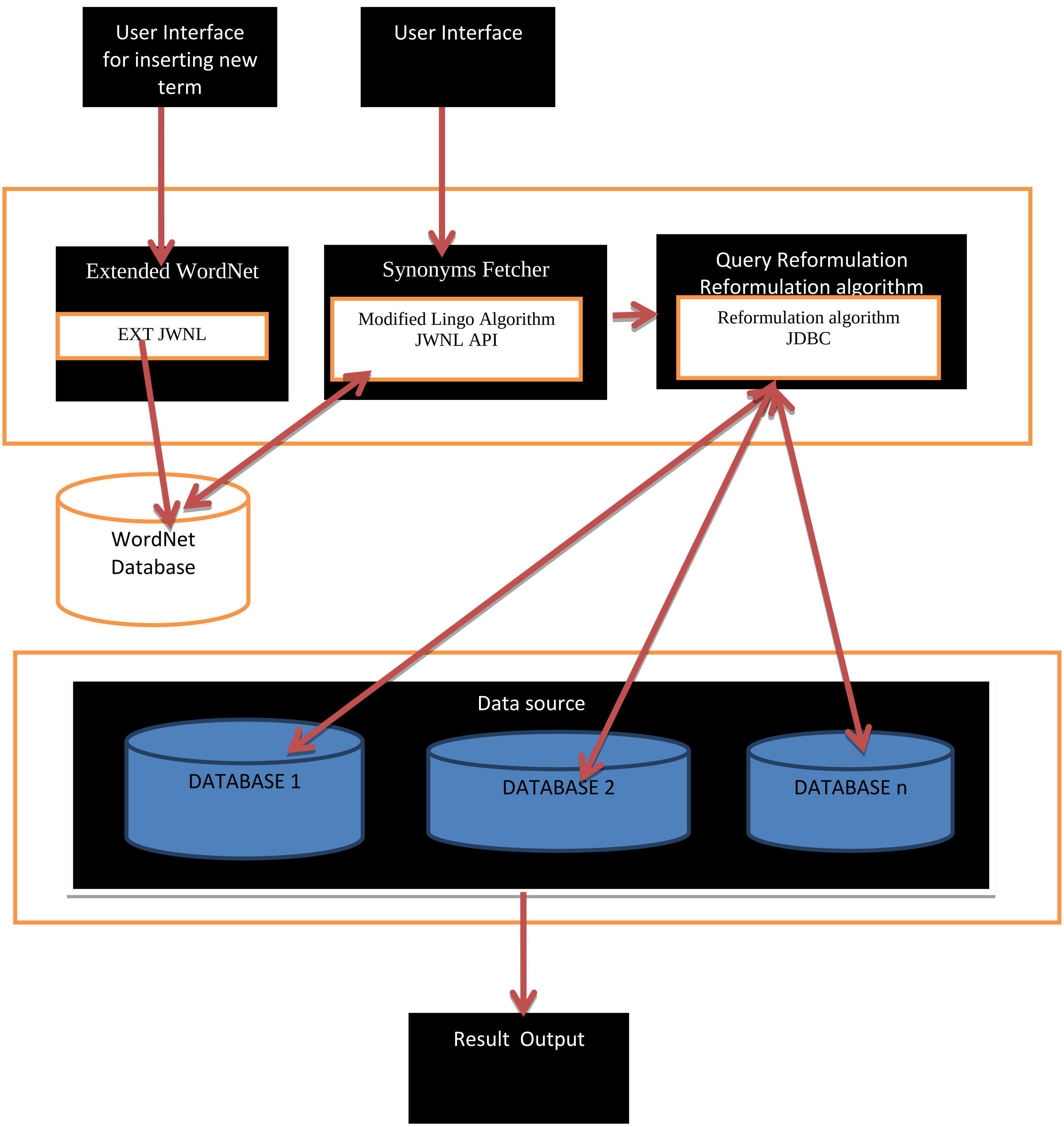


Figure 3.1: The system architecture

The proposed system architecture is as shown figure 3.1. The entire system comprises of different modules that interact together. The different modules are: The query interface, the semantic engine, result presentation and data source module.

3.3.1 The Query Interface

The user submits a query through the user interface; the user search query will be forwarded to synonyms fetcher to retrieve the corresponding synonyms or semantically related terms from the WordNet database. The user interface was designed using java programming language.

3.3.2 The Semantic Engine Module

This is the most important component of our system; it is actually this module that houses some components of the entire system. Other modules that are incorporated in this module are Extended WordNet view, synonyms fetcher and query reformation module.

3.3.3 Extended WordNet view module

This module is an interface through which new terms and synonyms are introduced into the original WordNet database.

3.3.4 Synonyms fetcher module

This module (Synonym Fetcher) was implemented to extract synonymous terms from the WordNet database. The module interacts with the user interface to collect the user input query in order to retrieve the semantically related concept from the WordNet. The module interacts with the WordNet using Java Application interface (Java API).

The Synonym Fetcher module use the modified Lingo algorithm to cluster the retrieved synonyms and pass it on to the query reformation module

The synonyms Fetcher module is written in Java to make use of the WordNet search API, which allows direct access to the WordNet database. This method is efficient because it only requires to be executed once for every query term. Then, query reformation module only needs to efficiently access this module in order to fetch synonyms. This module is composed of the modified lingo algorithm (synonym cluster algorithm) shown in section 3.4.

3.3.5 Query Reformation module

The Query Reformulation module is composed of the query reformulation algorithm. It interacts with the synonym fetcher to get the retrieved synonyms. The query reformulation algorithms contained in this module interpret and transform the retrieved synonyms into corresponding relational (SQL) queries. This module has query generator (not shown in the diagram) that generate queries internally. The mapping of terms to table is created automatically during the query processing to retrieve the available document(s) that stores the information about the user query.

The main goal of this module is to translate a user query into a set of queries which best suite the structure of the distributed sources. This reformulated query reduces loss of information, and extends the result of a given query in a semantically meaningful way. Speeding up query processing is not the intent of the thesis; rather the main aim is to provide users with more additional meaningful answers in respect to a particular query. This module is composed of the query reformulation algorithm as shown in section 3.4.

3.3.6 Wordnet module

This module consists of the WordNet database and plays the major role in this application. The WordNet houses by this module are uses to extend the meaning of any concept represented in the user query and to determine the semantic similarity between the concepts. WordNet is an online lexical resource for English language that classify synonym terms into synsets, each expressing a distinct concept. The WordNet database contains information on semantic relatedness for words, including relations such as synonymy (same meaning), antonymy (opposite meanings), hyponymy (hierarchical subordination), and meronymy(part-of relationship).

WordNet ontology is a good resource for mapping concepts (Fellbaum, 1998). The system extend the user query with the synonyms in WordNet belonging to the synset of each term contained in the user query. WordNet contains different relations, however, this thesis only exploit the synonyms part of the relations. It is used to generate semantically meaningful queries by deriving the synonyms related to the query terms.

The extended WordNet module is an interface through which new terms and sense could be introduce into the original WordNet database. This module contains the script that is responsible for inserting new term(s) into the original WordNet database files.

3.3.7 Result Presentation module

This module is responsible for the presentation of the result returned from the data sources to the user in the required form and format. An object-oriented programming language-java is used to design this interface.

3.3.8 Data source module

This module contains a set of autonomous databases as local data sources of a common domain. The data source module creates a uniform platform for all the available database(s). This module stored reasonable numbers of databases drivers, which the JDBC utilizes to understand the metadata of a particular table. The available databases in this platform are assumed to have modeled their data independently to their requirement and application. This is where the system queries to retrieve information. There is no interest in insert and update operations, as the flow of information is always from the databases to the system. For the system to have access to the databases, it must be in the data source module. Once, it is in this module, the application automatically detect all the available fields in the database-table using the Java Database Connectivity.

3.4 The Synonyms Cluster Algorithm

The pseudo code for the synonyms cluster is outline below:

```
Initialize empty list
   Searching(input-keyword){
        If not emptystring(input-keyword){
             Push stock of synonyms(keyword)  {
   While synonyms exist (keyword, wordNet){
        Push synonyms to list;
}
  }
Push keyword to list;
}
// Discover the data source
For each data in the data source {
Use the reformulation algorithm to reformulate query;
      If   list is not empty {
          For each item in list {
           Reformulate query;
          Get and store result set in result;
}
If empty result{
Return no result found
}
else
{
Return result set
          }
     }
}
// learning new term
```

```
If user propose similar word{
Store (keyword, synonyms, wordNet)
}
```

3.5 Query Reformulation

Query reformulation is a way in which the system is programmed to imitate the human in which user rewrite a particular queries in different form in order to get or increase the chance of hit. The system is designed to reformulate the use submit query base on the available synonyms from the WordNet. The basic assumption in the context of data integration is that, potential documents satisfying the user query may not contain the keywords as formulated by the user, but keywords either differently formulated or having a close meaning. The new list of synonyms mined from the WordNet and by a query reformulation process would allow the system to consider those relevant documents to the user query despite the difference in syntax.

The main goal of query reformulation is to translate a user query into a set of queries which best suite the structure of the distributed databases in the data source module (see fig 3.1). This reformulated query provides the user with more results to his/her query. The query reformulation module comprises of the query reformulation algorithm. The reformulation algorithm is given in section 3.6.

3.5.1 Query Reformulation Algorithm

There can be many different ways to refer to a single concept. This may occur when single entity are modeled differently in different databases to suit individual needs, particularly where spelling variations are common (for instance, Television can be indicated with its short form, Telly), acronyms (for instance, PC is sometimes use instead of the extended form, personal computer). Each one of these cases can be consider to be synonymy issues.

WordNet can help in resolving these issues, because it provides semantic relation and synonyms, called synset, for example, {TV, Telly, television, telecast } is the synset corresponding to the "*Television*". Taking these observations into cognizance, a query reformulation algorithm is developed to exploit these relationships. In essence, the query reformulation is performed by considering the synonyms of the query keywords retrieved from WordNet. The query reformulation is done in accordance to the following algorithm:

Define S = {s_1, s_2, s_3,…, s_n}, to be the set of synonyms retrieved by the extended lingo algorithm.

Define D = {d_1, d_2, d_3,…, d_n } as the set of databases

Result = dataset of possible output.

$\forall d_i \in D$ // d_i is in D, i=1 to n

Define T = {t_1, t_2, t_3,…,t_p} as the set of tables in d_i

$\forall t_i \in T$ // table t_i in T, i = 1 to p.

Define F = {f_1,f_2, f_3,…,f_m} as the set of field in table t_i

$\forall f_j \in F$ // field f_j in T, j = 1 to m.

$\forall s_k \in S$ // s_k is an element of S, where S is a set of synonyms, and k= 1 to n

Result = Result U SELECT * from T, where f_j contains s_k

End

End

End

End

Return result

3.7 Creating Set of Terms for Querying Sources

For the set of terms to be use, the set of tables proposed in (Amshakala and Nedunchezhian, 2011), was adopt because it comprises an already categorization of terms and synonyms, its domain is broad enough to be realistic, the content of the tables can be applicable in the domain of e-commerce and could be understandable for non-experts in order to be able to explain results. The dataset in each table were synthesized by duplicating the records based on their available synonyms in the WordNet, in order to test the effect of data integration based on semantic.

3.8 Concept Mapping

The attempt to integrating heterogeneous information sources called for concept mapping in order to establishes correspondences. The various information sources should not be perceived as a standalone model of the world, instead, it should be perceived as glue that put together information of various kinds.

Given the fact that there will be multiple databases in a particular domain, that participate in data integration, concept mapping aims to find semantic correspondences between similar elements of different databases.

Automatic ontology mapping has been the focus of much research for a number of years, and many tools have been developed for this purpose (Li and Chow, 2009). However, this thesis employs both automatic and manual mapping method. The system performs automatic mapping using synonyms extract from the WordNet to detect semantically similar concept from the available databases. However, if the given concept does not have a corresponding meaning in the WordNet, the system rely on hard linked approach (manual) for the mappings involve. The manual mapping method for unknown concept by the WordNet was chosen because of the relatively small size of our databases. This method will become ineffective as the databases increases.

In this context, the term ‘concept mapping’ refers to the connection or interlinked of different terms together. Technically, concept Mapping is the process of relating similar terms, concepts or relations of two or more information sources using synonymous relations. These relations are established by the system, so that the results are used to perform search on the available databases with little or no manual interference. The concept mapping in this thesis, do not include semantic consistency checking. The process only relates concepts and relations of one database with concepts and relations of the others. See the illustration in tables 3.1.

Table 3.1: Mapping of concept between two tables that store related information.

ID	**Product_Name**	**Price**	**Maker**
1	Television	500	HP

ID	**Item_Name**	**Cost**	**Model**
2	TV	550	Compaq

CHAPTER FOUR

SYSTEM IMPLEMENTATION

4.1 INTRODUCTION

The proposed system is a windows application. It is implemented in Netbeans Integrated Development Environment with java programming language. Java WordNet library (JWNL) is use to search given words in WordNet database. The JWNL includes handy data structures that make easy to compare two words. It provides methods to find least common subsumer or path length between words.

The data set used in the evaluation of the system and the actual implementation is presented in this chapter.

The system was designed and implemented using Java, as shown in Figure 4.1. For many reasons java was choosing for designing and implementing the system. First of all, some components of the WordNet are java compliant; hence the system could reuse some of its software components.

Secondly, with its emphasis on Object Oriented Programming (OOP), Java enforces the much needed good software practices such as the use of interfaces and proper code organization. Finally, Java comes with a package of optimized ready-to-use data structures (such as hash tables or lists) that make programming faster and less error-prone.

4.2 Semantic Similarity of Word

Semantic similarity and identification is very important for the functionality of the proposed system. Though, there are a large number of measures developed for finding the semantic similarity between words in the literature, which have been successfully applied to natural language tasks such as word sense disambiguation, synonym identification, text summarization, text annotation, information extraction and information retrieval. However, in this thesis, the lexical database of WordNet is used in measuring the semantic similarity because it provides is-a hierarchies for nouns. The similarity measures of the system use only the structure and the content of WordNet.

4.3 Synonyms Clustering Algorithm

The synonyms cluster algorithm was implemented in the synonyms fetcher module of the system. The algorithm is a modified version of the Lingo algorithm. The original Lingo algorithm was in five phases but independent and separable. The independent nature of each phases of the algorithm was the direct incentive to create a modified version of the phrase extraction phase, called synonyms cluster algorithm in this thesis.

However, unlike the main phrase extraction phase of Lingo algorithm, that use it to find cluster contents, the synonyms cluster algorithm is utilize to identify and group synonyms from WordNet that corresponding to the user input key word. The ability to retrieve and group synonyms is one of the major functionality of the integrated system. The pseudo code of the algorithm is shown in section 3.4.

4.4 Wordnet Database

WordNet (Fellbaum, 1998) Ontology is the major component of the proposed system. The WordNet serve as a controlled vocabulary to the system. The system utilizes the WordNet to mine for synonyms of a particular term, each time the user insert a query. The application interacts with WordNet database to retrieve synonyms, because one of the synsets is likely to be in the same synset group to the user entered keyword. This is the major inspiration of this thesis.

The WordNet is use as a source of control vocabulary to help clarify the sense of each query term and thus be able to generate the correct synonyms, which the system make use of to reformulate the query.

However, despite the importance of WordNet for research and application development in natural language processing, WordNet does not cover all vocabularies and some terms that are from human understanding synonymous are not group in the same synset in the WordNet. Previous researches that make use of WordNet make little or no effort to insert new terms or words into the synset to cover more general vocabularies. This thesis is able to update the WordNet database by adding a new term to a group of synset. This makes the system flexible and scalable due to its ability to allow user to add their own custom synonym(s) to the existing ones. Though, it takes quite a little time to add a single term. figure 4.1, show a WordNet interface with the term 'mota' added to the synset of car.

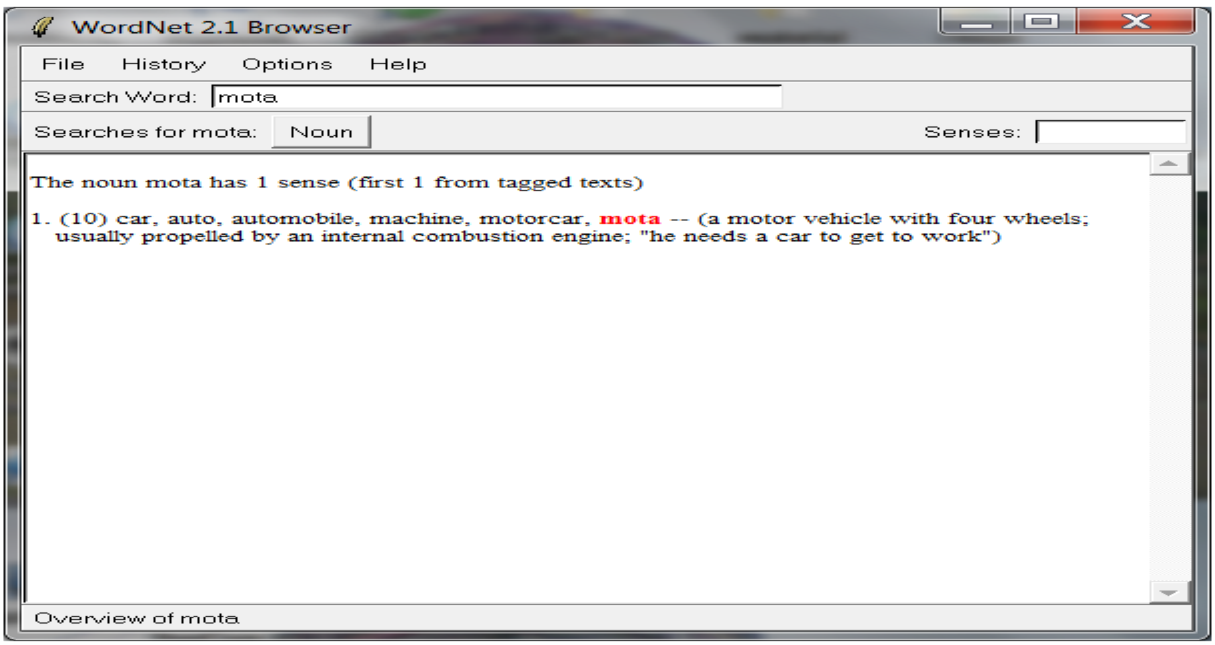


Figure 4.1: WordNet interface with the term mota added to the car synset.

4.5 Retrieved Synonyms from Wordnet Database

```
public ArrayList<String> syns(String wordForm, String path)
  {
        ArrayList<String> synonyms = new ArrayList<String>();
        Connection c = null;
         Statement s = null;
        ResultSet r = null;
        boolean stat = true;
```

```
        try
        {
                String pathe = System.getProperty("user.home")+"/.semantic_database data";
                Class.forName("org.sqlite.JDBC");
                c = DriverManager.getConnection("jdbc:sqlite:"+pathe+"/"+"database.db");
                s = c.createStatement();
                r = s.executeQuery("select * from rownum;");
                while(r.next())
                {
                stat = r.getBoolean(3);

                }
        }
        catch(Exception e)
        {
                JOptionPane.showMessageDialog(null, e, "Report", JOptionPane.ERROR);
        }

        if(stat)
        {
                System.setProperty("wordnet.database.dir", path);
                WordNetDatabase database = WordNetDatabase.getFileInstance();

                if(wordForm.equals(""))
                {
                    JOptionPane.showMessageDialog(null, "Enter a word!", "Report", JOptionPane.ERROR_MESSAGE);
                }
                else
                {
                Synset[] synsets = database.getSynsets(wordForm, SynsetType.NOUN);
                if(synsets.length > 0)
                {
                        for(int i = 0; i < synsets.length; i++)
                         {
                                        String[] wordForms = synsets[i].getWordForms();
                                        for(int j = 0; j<wordForms.length; j++)
                                {
                                         if(!synonyms.contains(wordForms[j]));
                                        {
                                                synonyms.add(wordForms[j]);
                                        }
                                        }
                        }
                }
                else if(synsets.length == 0)
                {
                        //do nothing, returns 0
                }

        }
    }
    else
    {

       synonyms.add(wordForm);

    }

    return synonyms;
  }
```

Figure 4.2: Codes segment for retrieving synonyms

WordNet ontology is actually a lexicon database that store words and its semantic relation. The WordNet database has a method that can access the WordNet files for a specified word form. The implementation of the retrieved synonym process was created with a segment of the code in figure 4.3. The retrieved synonyms are stored in a list form, which will be used in query reformulation.

4.6 How the Retrieved Synonyms are Mapped to Tables

```
String url = (String)datatable.getValueAt(i, 1);
                        String user = (String) datatable.getValueAt(i, 2);
                        String pass = (String) datatable.getValueAt(i, 3);
                        String table = (String) datatable.getValueAt(i, 4);
                        String field = (String) datatable.getValueAt(i, 5);
                        try
                        {
                                                Class.forName(driver);
                                                if(driver.equals("org.sqlite.JDBC"))
                                                     con = DriverManager.getConnection(url);
                                                else
                                                    con = DriverManager.getConnection(url, user, pass);
                                                stmtm = con.createStatement();
       StringBuilder sb = new StringBuilder("SELECT * FROM "+ table +" WHERE "+ field +" IN (");
                                                boolean added = false;
                                    //iterate through the list of synonyms and re-arrange into sql statement
                                    for (Iterator<String> it = syns(word, wordnetlocation).iterator(); it.hasNext();) {
                                    String s = it.next();
                                    if(added)
                                    {
                                                          sb.append(",");
                                    }
                                     sb.append("'");
                                    sb.append(s);
                                    sb.append("'");
                                    added = true;
                         }
                                    sb.append(")");

              rs = stmtm.executeQuery(sb.toString());//query  select * from TABLE where FIELD in (synonymns)

                                    if(rs != null)
                                    {
                                                while(rs.next())
                                                {

                                    ResultSetMetaData mdx = rs.getMetaData();
```

Figure 4.3: Mapped retrieved synonyms to table

Synonyms retrieved from the WordNet are used to reformulate the query. What is meant by mapping is actually, how the system sent all the reform queries to the available database.

4.7 Evaluation Criteria

4.7.1 Hardware Requirement

The evaluation test was performed using Processor 2.0GHz, RAM 2GB and Windows 7 on the standalone PC. The dataset for the experiments are set of tables shown in table (4.1 -4.3). On this test, queries are executed for different numbers of tables in the various databases and the results analyzed.

4.7.2 Software Requirement

The system used run on windows 7, and the application was designed on Netbeans environment with 2.1 version of Wordnet installed. For any system to run this application it must have java runtime environment and the WordNet installed. In addition a specific driver of the database to be integrated must be selected or added as the case may be.

4.8 Experimental Result and Discussion

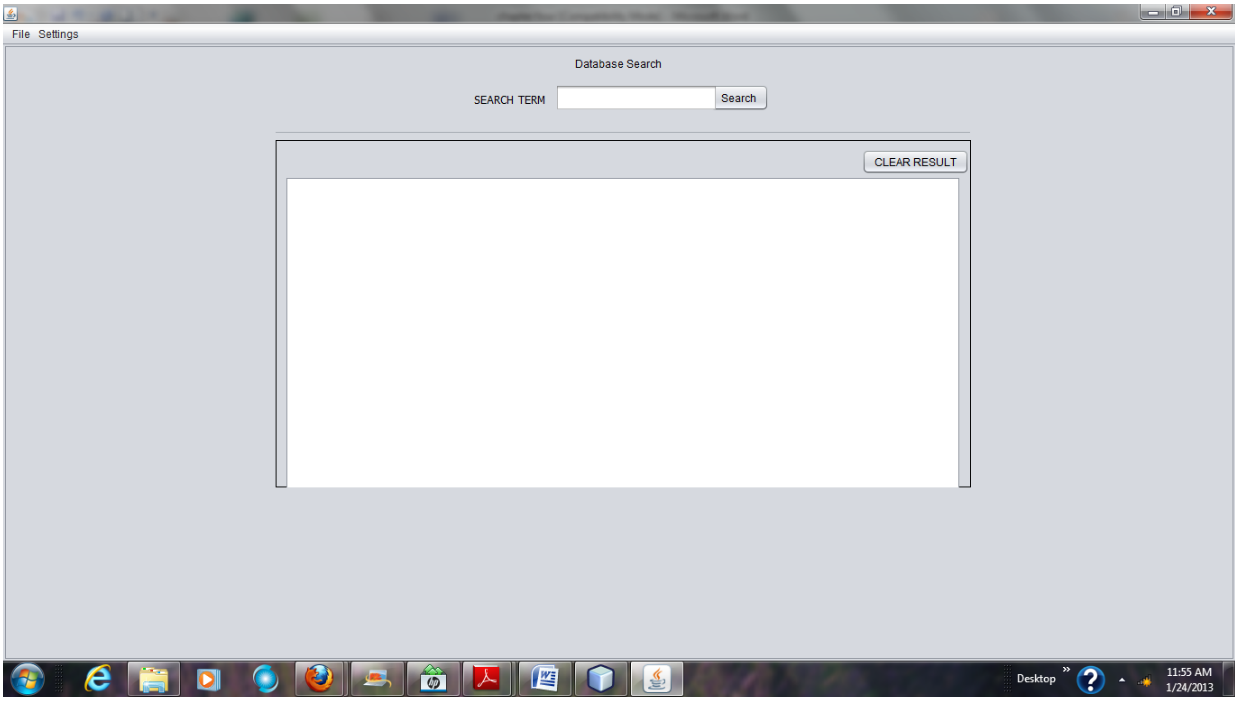


Figure 4.4: The main system interface showing presentation panel

Figure 4.4 shows the main GUI. Java swing is used to design the GUI. It has two menus; the first menu is file menu. It has just one submenu-the exit menu.

The second menu is the settings menu. It has two submenus; setup system parameters and add word menu. (i) Setup system parameters are used to enable connection to the WordNet database and the available databases. This menu is used to initialize the path to the directory where the WordNet database has been installed and that of the existing databases in the data source. (ii) Add word submenu is used to add customized synonyms. This menu would be use to manually link words that were absent in WordNet.

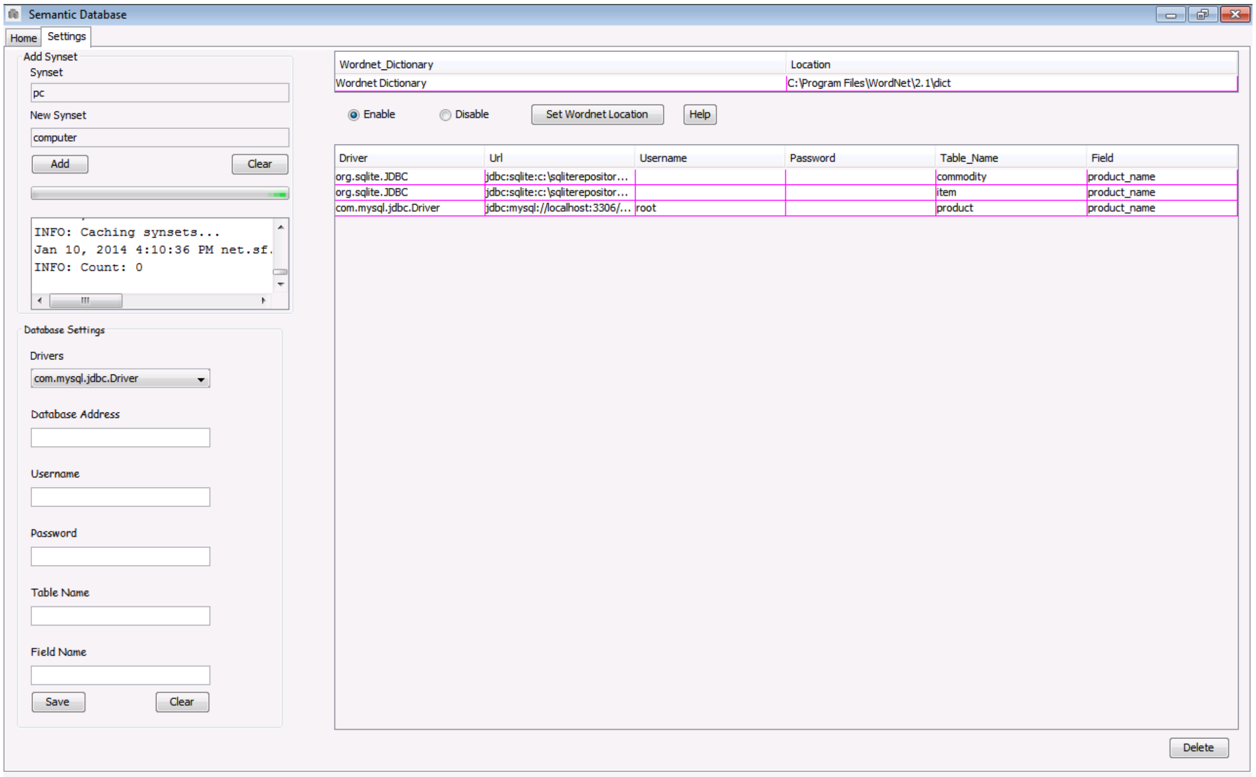


Figure 4.5: The system interface showing database and wordNet location

Figure 4.5 shows the setup parameters menu from which we can set the wordNet location and add the required database. The wordNet location is C:\program Files (x86)\WordNet\2.1\dic. If successfully set, the package stores internally a pointer to the WordNet dictionary which is used by the system.

4.9 Experimental Evaluation and Data input

For the experimental purposes of this thesis, the set of tables proposed in the research by (Amshakala and Nedunchezhian, 2011), is adopted, because it comprises an already categorization of terms and synonyms, its domain is broad enough to be realistic, the content of the tables can be applicable in the domain of e-commerce and could be understandable for non-experts in order to be able to explain results.

4.10 Tables Used in the Databases for Experimental Evaluation

The document collection used to build the tables in the different databases is from (Amshakala and Nedunchezhian, 2011). The table comprises an already categorization of terms and synonyms, its domain is broad enough to be realistic, the content of the tables can be applicable in the domain of e-commerce and could be understandable for non-experts in order to be able to explain results. The tables are used as they are taken from the original source.

The dataset in each table were synthesized by duplicating the records based on their available synonyms in the WordNet, in order to test the effect of the semantic integration. The tables in the different databases are display below. Format of the Tables and relations described in it has been taken as they are from their original source.

Table 4.1: Item table from mysql database

```
mysql> use mysqldb;
Database changed
mysql> select * from item;
+------+------------------+-------+
| id   | product_name     | price |
+------+------------------+-------+
| 1    | cellular phone   | 5000  |
| 2    | television       | 20000 |
| 3    | sound system     | 5000  |
| 4    | personal computer | 30000 |
+------+------------------+-------+
4 rows in set (0.06 sec)
```

Table 4.2: Commodity table from derby database

ID	PRODUCT_NAME	PRICE
1	mobile	3000
2	telecast	25000
3	music	4000
4	computer	25000

Table 4.3: Product table from sqlite database

id	product_name	price
1	mobile phone	4000
2	tv	25000
3	stereo	5000
4	pc	20000
5	washer	10000

4.11 Results and Discussion

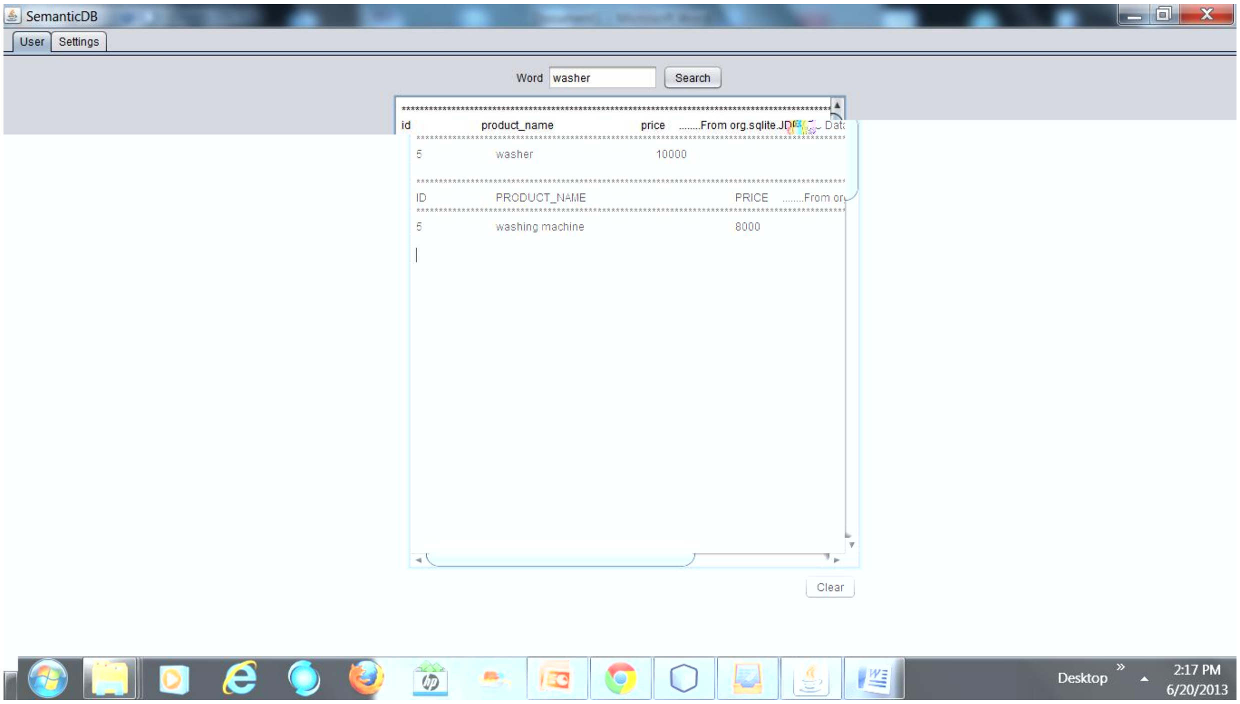


Figure 4.6: The system interface showing displaying result in the result panel.

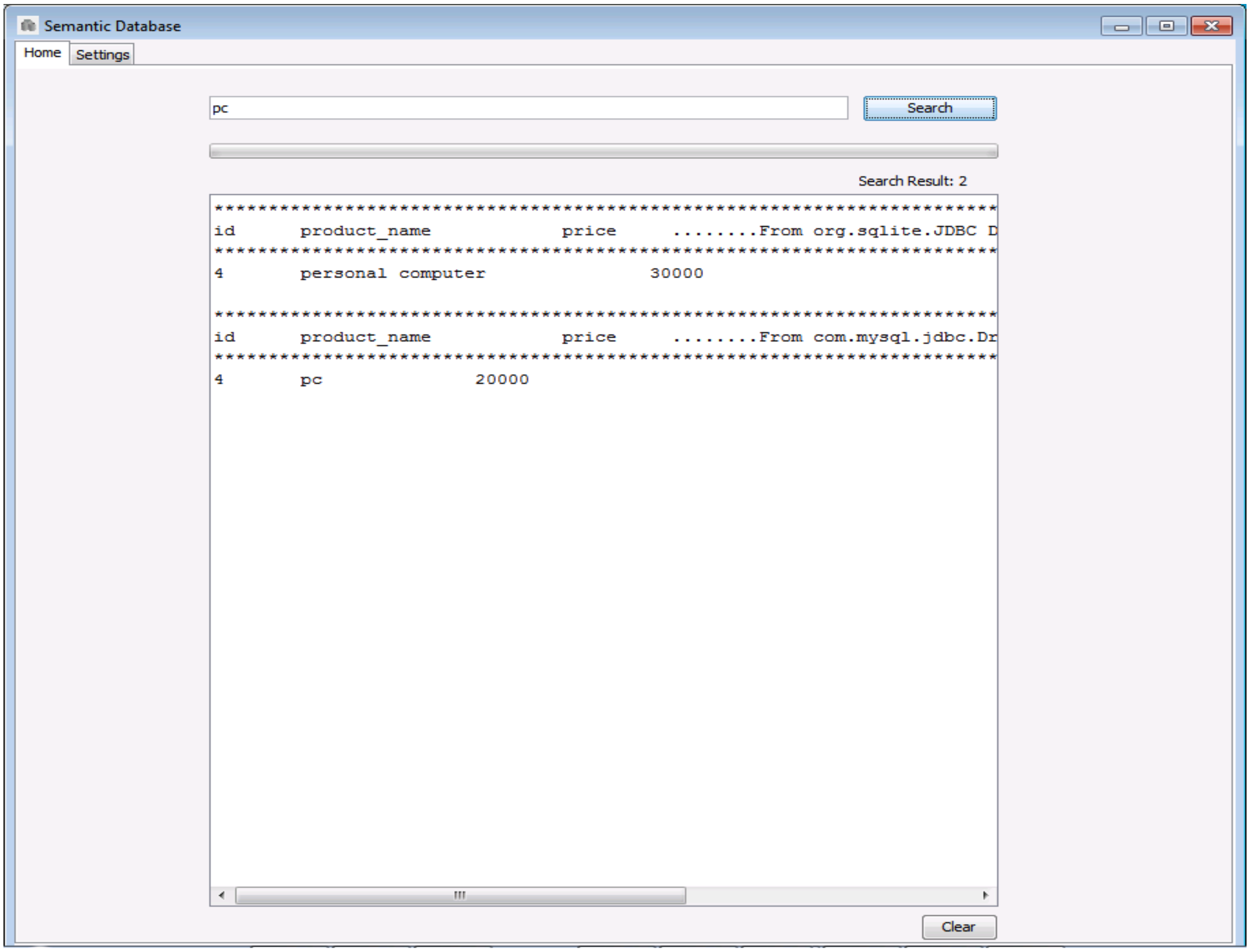


Figure 4.7: The system interface displaying result in the result panel.

As display in figure 4.6 and 4.7, the system was able to use Semantic knowledge to retrieve washing machine, using washer and personal computer using pc as the search term respectively. The result in figure 4.6 above was generated, with the word “washer”. This was possible because of the knowledge that “washer” and “washing machine” are synonymous.

The above results show an enhancement in the ability of retrieving concept based not only on exact meaning, but also on the existence of semantic relationships between the label terms.

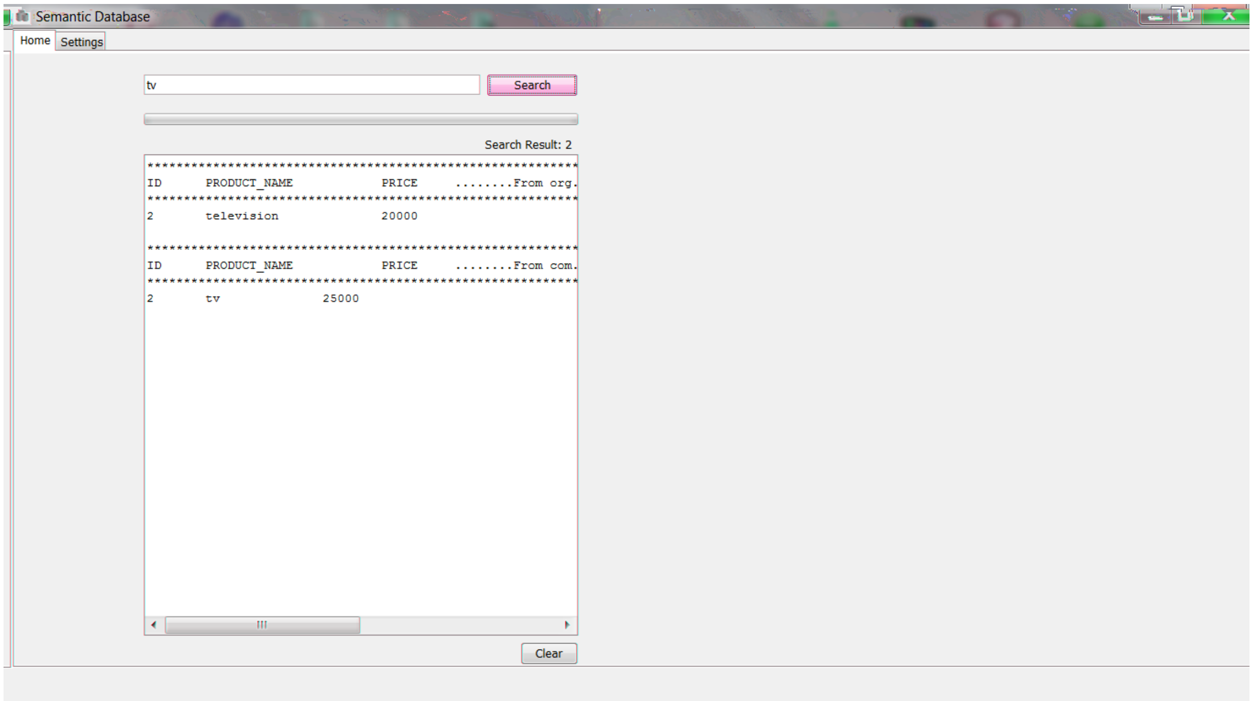


Figure 4.8: The system interface showing displaying result in the result panel using tv as search term.

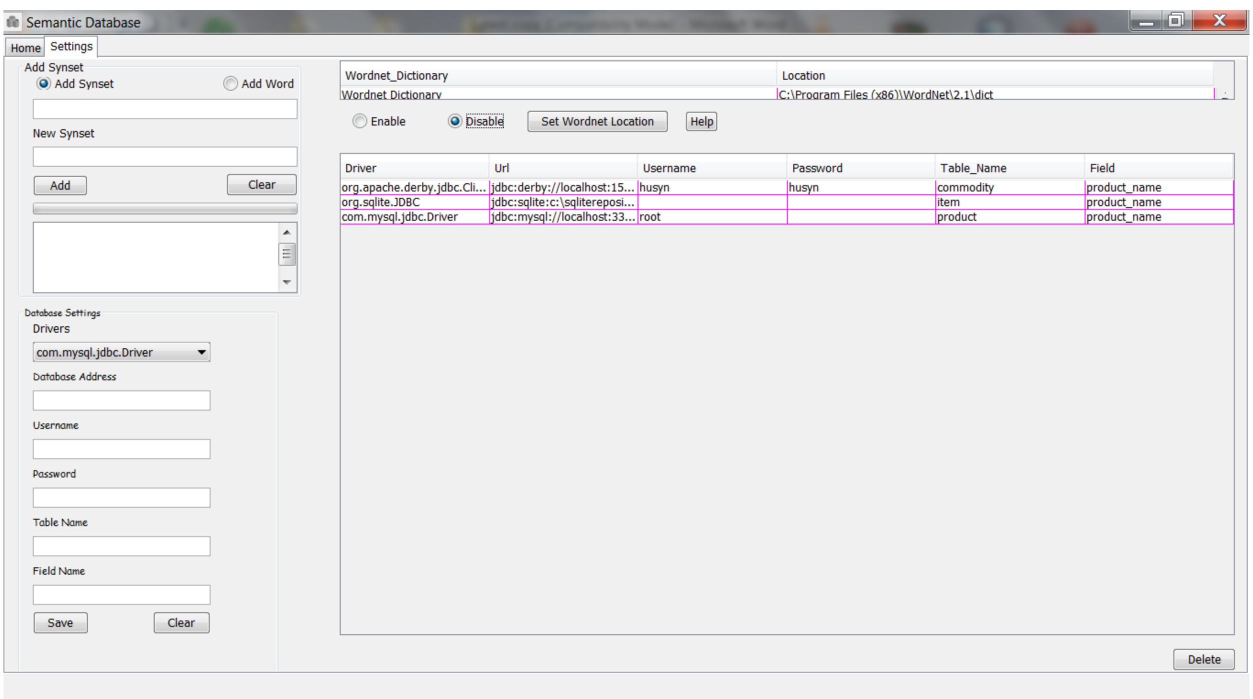


Figure 4.9: The system interface with the WordNet disable

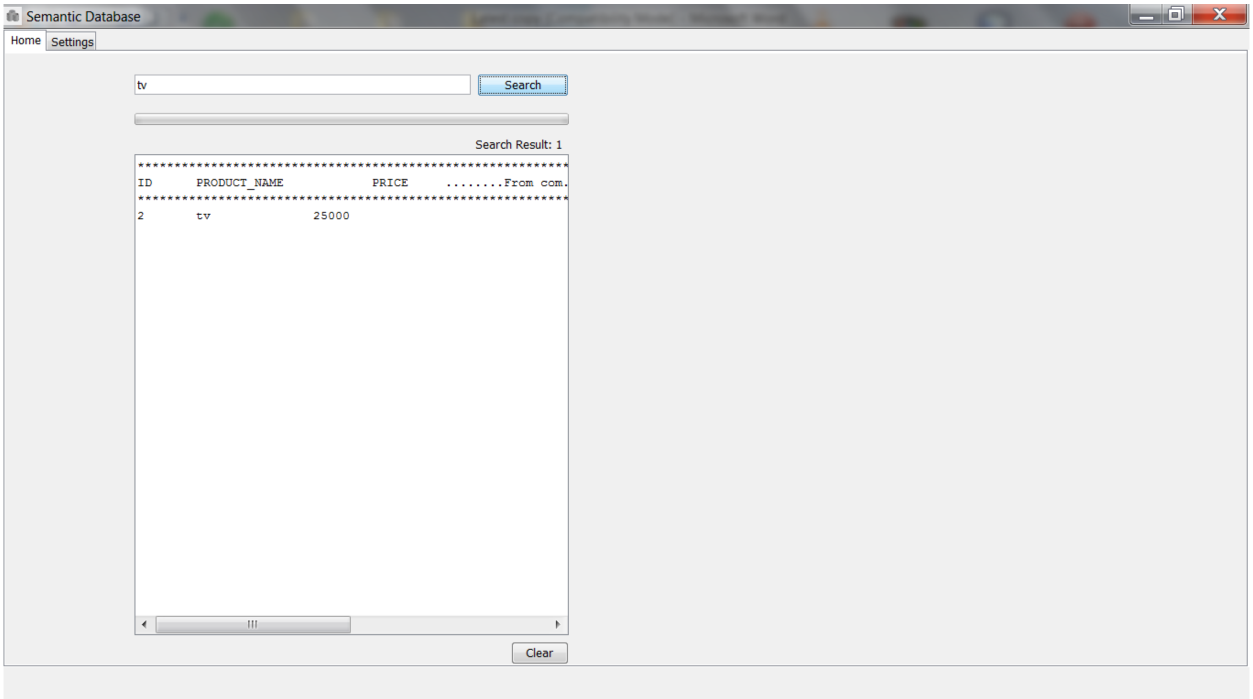


Figure 4.9: The system interface showing result of tv as WordNet was disable

The system was designed with so much flexibility in mind, so the WordNet component of the system can be disabled to test the effect of the WordNet in the system. Figure 4.9 show the interface of the WordNet disable in the application. With the WordNet disabled, the system was tested by searching for the term tv, and as expected only the term marching the search query was retrieved, thus, we have loss of information because there are other term(s) present in the system that matches the user intent but was not retrieved. Figure 4.8 and figure 4.9, displayed the result for search term tv, with and without WordNet respectively. The execution time for different number of tables is displayed in figure 4.10.

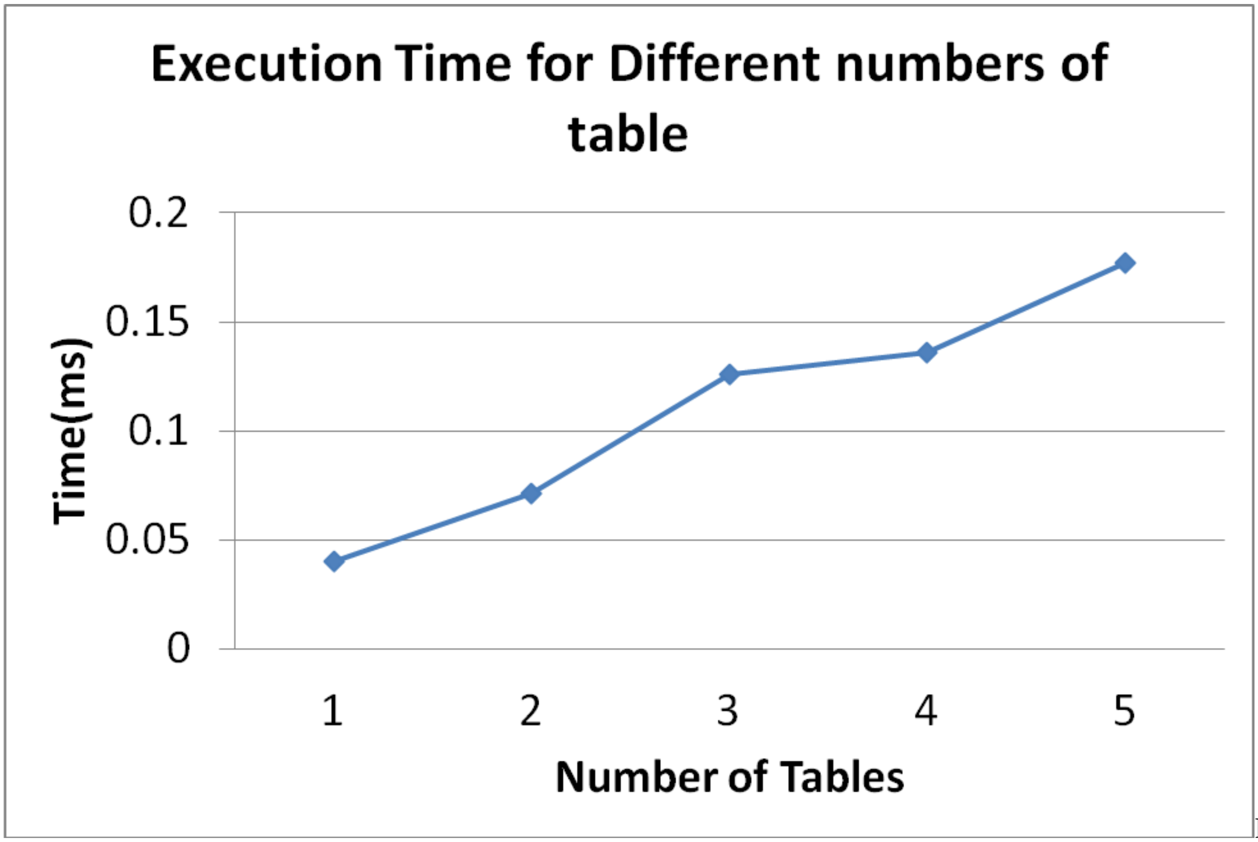
4.12 Further Experiment

Having had a successful result or so using the data set provided in (Amshakala and Nedunchezhian, 2011), the system was tested with data set provided in the thesis by Ma'aruf

(2010), to assess the effect of using WordNet in different domains. On the contrary to the data set in (Amshakala and Nedunchezhian, 2011), the effect of WordNet here is not apparent and indeed the result was worse compare to what is reported in the thesis.

One possible reason to this observation is that lots of technical terms are used in the tables provided in Ma'aruf (2010), which deals in school domain. These technical terms are not documented and classified specifically in accordance to their usage in the WordNet. For example, ''DOB'' has no entry in WordNet, in the case of ''Emp_Name'', Emp has no entry in WordNet either. These among other set of terms lead to the effect that cause WordNet to perform less satisfactory in this domain.

A possible way to overcome this limitation is to augment WordNet with domain specific term relations or to make use other domain specific ontology.



F

igure 4.10: Execution Time Graph

4.13 Conclusion

This thesis presents a prototype for data integration system, using WordNet as a controlled vocabulary without forcing any changing on the individual databases to conform to a particular ontology language.

The application designed in this thesis will expand the given query, using query reformulation module that makes use of the extracted synonyms and input query to modify the query semantically with the aid of query reformulation algorithm and map the concepts down to the appropriate databases.

The implementation turned out to be feasible in practical applications, as the system was able to retrieved results based on the available synonyms using knowledge of relations between words.

Nevertheless, the results obtained for using the data presented in Ma'aruf (2010), do not confirm the previous results. This was believed to be due to lots of technical terms used in the tables provided, which deals in school domain.

The execution graph reveal that the cost of execution increases with the number of tables that need to be accessed, this could be attributed to the query reformulation rule, and it could possibly be reduced if the system could be enhance to perform query reformulation into specific source.

CHAPTER FIVE

SUMMARY, CONCLUSION AND RECOMMENDATION

5.1 INTRODUCTION

This chapter presents the summary, conclusion and recommendation of the research in this thesis. The chapter summarize the entire work of the thesis, it contribution, limitations and the recommendation for future work.

5.2 Summary

The feasibility of structure a perfect ontology including the relations of all the resources in the various databases is highly impracticable in real integrated system; therefore, this thesis presented an approach for ontology-based data integration system, using an already existing ontology (WordNet) to facilitating seamless transparent access to different databases. The thesis considers the databases in their original form and format and makes use of WordNet to extract potential relation between the terms. The designed application that connects the databases and the WordNet adopting the three layer architecture. The WordNet module solves the problem of different name representations of same entity in the system, by providing terms that are synonymous to the user given query.

The implementation turned out to be fairly effective in practical applications, with a particular set of data using knowledge of relations between words. Nevertheless, the results obtained for using the data in school domain; do not confirm the previous results reported. That was believed to be due to lots of technical terms used in the tables provided, which are totally absent from WordNet database.

The execution time reveal high execution cost as the number of tables increase, reducing this execution time could enhance the system performance.

5.3 Conclusion

The objective of the thesis to design an integrated system for databases was achieved and the system has performed fairly satisfactory based on the data used in evaluating the system. Though, the prototype has only been tested with simple terms, and has the capability to answer single term query. This is possibly a weakness of the system, as certain terms may only be relevant when seen in connection with one or more other terms. The system does not take context into cognizance and the number of terms in the synonym set to be used in query reformulation was not specified. Adding these functionality and augment WordNet with domain specific term relations could possibly add more power to the system. The application designed in this thesis takes almost seven (7) minutes to add single term to group of synset in the WordNet database, and the execution time is observed to be the function of input tables.

Finally, evaluation of the designed application rely solely on English term available in the WordNet, therefore, cross-language processes could be considered later as an enhancement of the present work. Although, the application has certain limitations, however, it has been demonstrated that the idea of integrating databases using WordNet is viable and with some improvement and reduction of time to insert new term into a group of synset, it could serve as helpful tool for real time integration.

5.4 Recommendation

The major recommendation in this thesis is to stress the fact that our evaluation has relied on few test, and that a larger evaluation with domain knowledge, has to be done for more conclusive evidence of this approach. In addition, the proposed methodology revealed an overhead because of additional synonyms and query reformulation, reducing this overhead cost by reformulating queries into source specific and minimizes the time to add new term into WordNet databases could improve on the system.

LIST OF REFERENCES

Abdul-Kareem Sameem and Abdolreza Hajmoosaei. (2007). An ontology-based approach for resolving semantic schema conflicts in the extraction and integration of query-based information from heterogeneous web data sources. Paper presented in Australasian Ontology Workshop (AOW-07), Gold Coast, Queensland, Australia: pp. 310-314.

Akilandeswari J. and R.Subhashini (2011).International Journal of Enterprise Computing and Business Systems. 1(1). January 2011 (Online)http://www.ijecbs.com.

Amshakala K. and Nedunchezhian R., (2011). WordNet Ontology Based Query Reformulation and Optimization using Disjunctive Clause Elimination. *International Journal of Database management System (IJDBM)*, 3(4); pp. 55-62.

Bagiwa A. M. (2011). Improving Ontology Matching Towards Achieving Semantic Interoperability. Unpublished M.Sc. Thesis. Ahmadu Bello University, Zaria, Nigeria.

Bernstein Green. (2008). Data and database integration: A survey. Journal of Web Engineering, 5(3); pp. 210-215.

Fellbaum C. (1998). WordNet: an electronic lexical database. Cambridge, Mass: MIT Press.

Ghawi, R, and Cullot, N. (2009). Building Ontologies from XML Data sources. 1st International Workshop on modeling and visualization of XML and Semantic Web Data. Held in Linz-Austria:pp. 480 – 484.

Gulati Payal and Sharma A.K. (2010). Ontology Driven Query Expansion for Better Image Retrieval. *International Journal of Computer Applications*, 5(10); pp. 33-37

Gruber , T. (2009). Ontology, Encyclopedia of Database Systems. Springer-Verlag, in press.

Guarino, N and Giaretta, P. (1995). Ontologies and knowledge bases: towards a terminological clarification towards very large knowledge bases, Amsterdam, IOS Press: pp. 25-32.

Kavitha, C., Sadasivam, G. Sudha, and Shenoy, Sangeetha. (2011). Ontology Based Semantic Integration of Heterogeneous Databases. *European Journal of Scientific Research*, 64(1); PP.115.

Kashyap V. and Sheth A, (2006). Semantic and schematic similarities between database objects: a context based approach. *The Very Large Data Base Endowment Inc (VLDB)Journal*, pp. 277-304.

Li Kuang**.** (2009). Semantic Web for Data integration within Oil&Gas or maritime. Msc thesis. University of Agder Grimstad, Norway.

Li Xiao, Randy Chow. (2009). An Ontology-based Mapping Repository for Dynamic and Customized Data Integration. (CISE Technical Report), University of Florida, United State.

Ma'aruf L.M. (2010). Achieving heterogeneous database integration through database schema analysis. Unpublished M.Sc. Thesis. Ahmadu Bello University, Zaria, Nigeria.

Necib Chokri Ben and Johann-Christoph Freytag. (2005). Semantic Query Processing using Ontologies. Proceedings of the 17th conference on advanced information systems engineering (CAISE'05), held in Porto, Portugal, pp.167-186.

Osinski, Stanislaw, Leon and Dawid Weiss. (2004). Conceptual Clustering Using Lingo Algorithm: Evaluation on Open Directory Project Data. Advances in Soft Computing, Intelligent Information Processing and Web Mining, Proceedings of the International IIS: IIPWMÂ´04 Conference, Zakopane, Poland: pp. 359-368.

Osinski, Stanislaw, and Dawid Weiss. (2005). A Concept-Driven Algorithm for Clustering Search Results. *IEEE Intelligent Systems,* 20(6); pp. 48-54.

Raji Ghawi. (2010) "Ontology-based cooperation of information system", Ph.D. thesis de Bouegogne University, France.

Rakesh Jaitly. (2011). Implementation of Data Integration using Distributed Systems: A Review. International Journal Computer Technology Application, 2 (4); pp. 839-845.

Sameh**,** Ahmed and Kadray Amar. (2010). Modelling a semantic web system. *International* Journal of Research & Reviews in Computer Science, 1(2); PP. 71.

Sugandha Lohar, (2010). Semantic Integration of Heterogeneous Databases in Multidatabase System., Msc thesis, thapar University, India

Suwanmanee Seksun, Djamal Benslimane, Pierre- Antoine Champin and Philippe Thiran. (2009).Wrapping and Integrating heterogeneous Database with OWL. Proceed of International conference on Enterprise Information Systems: (ICEIS), Miami- U.S.A: pp.145-150.

Wang Peng and Baowen Xu. (2008). Debugging ontology mappings: a static approach. *Computing and Informatics*, 27(1); pp. 21-36.

Wordnet Documentation : http://wordnet.princeton.edu.

Zhang Lei and Jing Li. (2011). Automatic Generation of Ontology Based on Database, Journal of Computational Information Systems 7(4); pp. 148-154.

Zhixiao Wang and Q,iang Niu. (2012). Research on Hybrid Query Expansion Algorithm. *International Journal of Hybrid Information Technology*, 5(2); PP. 207-212.

Zhou Xu, Guoji Xu, and Lei Liu. (2011). An Approach for Ontology Construction Based on Relational Database. *International Journal of Research and Reviews in Artificial Intelligence,* 1(1); pp.16-19.

APPENDIX

Source Code Listing

```
import java.awt.BorderLayout;
import java.awt.Dimension;
import java.awt.FlowLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.SQLException;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.*;
public class Semantic_Database extends JPanel{

   JTextField text;
    JButton search, clear;
    JTextArea area;
    JScrollPane spane;
    JLabel result, time, word;
    JPanel panel, pane, panex, pan, pa, p, px, bar_pane;
    JProgressBar bar;
    static Settings ty = new Settings();;
    Data_base dt;
    public Semantic_Database()
    {
       bar_pane = new JPanel();
       bar_pane.setPreferredSize(new Dimension(300, 30));
        dt = new Data_base();
        bar = new JProgressBar(0, 100);
        bar.setToolTipText("Moves when  searching");

        bar_pane.add(bar);
        text = new JTextField(10);
        text.setToolTipText("Enter the word here");
        search = new JButton("Search");
        search.setToolTipText("Click here to search word");
        search.addActionListener(new ActionListener()
        {

            @Override
            public void actionPerformed(ActionEvent e)
            {
             String path = (String)ty.getTablew();

                        try {
```

```
                        if(ty.getTablexz().getRowCount() == 0)
                        {
                           JOptionPane.showMessageDialog(null, "Database list empty!", "Report",
JOptionPane.ERROR_MESSAGE);
                        }
                        else
                        {

                        dt.search(ty.getTablexz(), area, text.getText(),path );

                         }
                     } catch ( SQLException | ClassNotFoundException ex) {
                        Logger.getLogger(Semantic_Database.class.getName()).log(Level.SEVERE,
null, ex);
                     }
          }
      });
      area = new JTextArea(150, 50);
      area.setToolTipText("Search Results");
      spane = new JScrollPane(area);
      spane.setToolTipText("Scroll up and down");
      result = new JLabel("Search Result: ");
      time = new JLabel("Time: ");
      word = new JLabel("Word");
      clear = new JButton("Clear");
      clear.setToolTipText("Click here to clear search results");
      clear.addActionListener(new ActionListener()
      {
         public void actionPerformed(ActionEvent ev)
         {
            area.setText("");
            text.setText("");
         }
      });


      panel = new JPanel();
      panel.add(word);
      panel.add(text);
      panel.add(search);

      panex = new JPanel(new FlowLayout(FlowLayout.RIGHT));
      panex.add(clear);

      pa = new JPanel(new BorderLayout());
      pa.add(spane, BorderLayout.CENTER);
```

```
        pa.add(panex, BorderLayout.SOUTH);
        pane = new JPanel(new BorderLayout());
        pane.add(panel, BorderLayout.NORTH);

        pan = new JPanel(new BorderLayout());
        pan.setPreferredSize(new Dimension(500, 600));
        pan.add(pane, BorderLayout.NORTH);
        pan.add(pa, BorderLayout.CENTER);

        add(pan);
    }

    public static void main(String[] args)
    {
        java.awt.EventQueue.invokeLater(new Runnable() {

            @Override
        public void run() {
        try
        {
            UIManager.setLookAndFeel("com.sun.java.swing.plaf.nimbus.NimbusLookAndFeel");

        }
        catch(Exception e)
        {
            e.printStackTrace();
        }

        JTabbedPane tpane = new JTabbedPane();
        tpane.add("User", new Semantic_Database());
        tpane.add("Settings", new Settings());
        JFrame frame = new JFrame("SemanticDB ");
        ImageIcon img=new ImageIcon("icon.png");
        frame.setIconImage(img.getImage());
        frame.setContentPane(tpane);
        frame.setVisible(true);
        frame.setLocation(250, 50);
        //JOptionPane.showMessageDialog(null, "Read the Help Tab if is your first time of using this
software", "Information", JOptionPane.INFORMATION_MESSAGE);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.pack();
            }
        });
    }
}
```

```
package semantic_database;
import java.awt.BorderLayout;
import java.awt.Dimension;
import java.awt.Font;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.*;
import java.util.ArrayList;
import java.util.Vector;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import javax.swing.table.TableColumn;
public class Settings extends JPanel implements ActionListener
{
   public String[] urladdress = new String[50];
   ArrayList<String> uuu = new ArrayList<>();
   private JPanel datapanel, buttonpanel, delpanel, synpanel,synpanel1,syntotal, tablepanel, pan;
   private JButton register, clear, delete, add, refresh;
   private JTextField word, syn, url, table, field, user, pass, deletefield;
   private JComboBox driver, wordnet;
   private String[] drivers = {"com.mysql.jdbc.Driver", "org.apache.derby.jdbc.EmbeddedDriver",
"org.apache.derby.jdbc.ClientDriver","org.sqlite.JDBC",
"org.hsqldb.jdbcDriver","org.postgresql.Driver",                    "oracle.jdbc.driver.OracleDriver",
"COM.ibm.db2.jdbc.net.DB2Driver", "com.sybase.jdbc.SybDriver"};
   private JLabel urlL, userL, passL, tableL, fieldL,wordL, synL;
   JScrollPane scrollpanex;
   JPanel paz;
   public JTable tablexz = null;
   Vector datax = new Vector(), columnNamesx = new Vector();
   private JScrollPane pane, panex;
   public JTable tablex;
   public JTable tablew = null;
   /*
   Connection con = null;
   Statement st = null;
   ResultSet rs = null;
   */
   @SuppressWarnings("LeakingThisInConstructor")
   public Settings()
   {
      super();
      tablexz = new JTable();
      paz = new JPanel();
```

```java
tablex = new JTable();
tablex.setToolTipText("Registered Databases");
tablew = new JTable();
datapanel = new JPanel(new GridLayout(12, 1));
datapanel.setBorder(BorderFactory.createTitledBorder("Database Settings"));
datapanel.setPreferredSize(new Dimension(250, 100));
driver = new JComboBox(drivers);
driver.setToolTipText("Select the Database Driver here");
url = new JTextField(10);
url.setToolTipText("Enter the database url here");
user = new JTextField(10);
user.setToolTipText("Enter the database username here");
pass = new JTextField(10);
pass.setToolTipText("Enter the database password here");
table = new JTextField(10);
table.setToolTipText("Enter the database table here");
field = new JTextField(10);
field.setToolTipText("Enter the database field here");

driver.addActionListener(new ActionListener()
{
   public void actionPerformed(ActionEvent ev)
   {
      if("com.mysql.jdbc.Driver".equals(driver.getSelectedItem()))
      {
         url.setText("jdbc:mysql://<hostname:port>/<databasename>");
      }
      else if("org.apache.derby.jdbc.EmbeddedDriver".equals(driver.getSelectedItem()))
      {
         url.setText("jdbc:derby:<databasename>");
      }
      else if("org.apache.derby.jdbc.ClientDriver".equals(driver.getSelectedItem()))
      {
         url.setText("jdbc:derby://<hostname:port>/<databasename>");
      }
      else if("org.sqlite.JDBC".equals(driver.getSelectedItem()))
      {
         url.setText("jdbc:sqlite:<database location>\\<databasename>.db");
      }
      else if("org.hsqldb.jdbcDriver".equals(driver.getSelectedItem()))
      {
         url.setText("jdbc:hsqldb:file:<database location>\\<databasename>");
      }
      else if("org.postgresql.Driver".equals(driver.getSelectedItem()))
      {
         url.setText("jdbc:postgresql://<hostname:port>/<databasename>");
```

```
        }
        else if("oracle.jdbc.driver.OracleDriver".equals(driver.getSelectedItem()))
        {
           url.setText("jdbc:oracle:thin:@<hostname>:<port Number>:<database>");
        }
        else if("COM.ibm.db2.jdbc.net.DB2Driver".equals(driver.getSelectedItem()))
        {
           url.setText("jdbc:db2:<hostname>:<port number>/<databasename>");
        }
        else if("com.sybase.jdbc.SybDriver".equals(driver.getSelectedItem()))
        {
           url.setText("jdbc:sybase:Tds:<hostname>:<port number>/<databasename>");
        }
    }
});

urlL = new JLabel("Database Address");
urlL.setFont(new Font("comic sans ms", Font.ROMAN_BASELINE, 12));
userL = new JLabel("Username");
userL.setFont(new Font("comic sans ms", Font.ROMAN_BASELINE, 12));
passL = new JLabel("Password");
passL.setFont(new Font("comic sans ms", Font.ROMAN_BASELINE, 12));
tableL = new JLabel("Table Name");
tableL.setFont(new Font("comic sans ms", Font.ROMAN_BASELINE, 12));
fieldL = new JLabel("Field Name");
fieldL.setFont(new Font("comic sans ms", Font.ROMAN_BASELINE, 12));

datapanel.add(driver);
datapanel.add(urlL);
datapanel.add(url);
datapanel.add(userL);
datapanel.add(user);
datapanel.add(passL);
datapanel.add(pass);
datapanel.add(tableL);
datapanel.add(table);
datapanel.add(fieldL);
datapanel.add(field);

buttonpanel = new JPanel();
register = new JButton("Register");
register.setToolTipText("Click here the register the Database");
register.addActionListener(this);
clear = new JButton("Clear");
clear.setToolTipText("Click here to clear database fields");
clear.addActionListener(this);
```

```
refresh = new JButton("Refresh");
refresh.setToolTipText("Refresh database");
deletefield = new JTextField(5);
deletefield.setToolTipText("Write the table name of database you want to delete");
refresh.addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent exc)
    {
        try {
            reload();
        } catch (ClassNotFoundException ex) {
            Logger.getLogger(Settings.class.getName()).log(Level.SEVERE, null, ex);
        } catch (SQLException ex) {
            Logger.getLogger(Settings.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
});
buttonpanel.add(register);
buttonpanel.add(clear);

panex = new JScrollPane();
pan = new JPanel(new BorderLayout());
pan.setPreferredSize(new Dimension(150, 120));
datapanel.add(buttonpanel);
tablex.setModel(databaseinfo);
tablew.setModel(wordnetinfo);
tablew.setToolTipText("Edit the Path field to change the wordnet location if it doesn't match
yours");

panex.setViewportView(tablew);

pane = new JScrollPane();
pane.setViewportView(getTablexz());

delete = new JButton("Delete");
delete.addActionListener(this);
delete.setToolTipText("click here to delete highlighted database");

delpanel = new JPanel();
delpanel.add(deletefield);
delpanel.add(delete);
delpanel.add(refresh);

wordL = new JLabel("Word");
wordL.setEnabled(false);
synL = new JLabel("Synonyms");
```

```
        synL.setEnabled(false);
        add = new JButton("Add");
        add.setEnabled(false);
        wordnet = new JComboBox();
        wordnet.setEnabled(false);
        word = new JTextField(8);
        word.setEnabled(false);
        syn = new JTextField(8);
        syn.setEnabled(false);

        synpanel = new JPanel();
        synpanel.add(wordL);
        synpanel.add(word);
        synpanel.add(wordnet);

        synpanel1 = new JPanel();
        synpanel1.add(synL);
        synpanel1.add(syn);
        synpanel1.add(add);

        syntotal = new JPanel(new BorderLayout());
        syntotal.add(synpanel, BorderLayout.NORTH);
        syntotal.add(synpanel1, BorderLayout.SOUTH);
        pan.add(panex, BorderLayout.WEST);
        pan.add(syntotal, BorderLayout.CENTER);

        setLayout(new BorderLayout());
        add(datapanel, BorderLayout.WEST);
        add(pane, BorderLayout.CENTER);
        add(delpanel, BorderLayout.SOUTH);
        add(pan, BorderLayout.NORTH);

    }
    public javax.swing.table.DefaultTableModel databaseinfo = new
javax.swing.table.DefaultTableModel(
            new Object[][]{},
            new String[]{"Driver","Database URL", "Username", "Password", "Table Name", "Field
Name"}) {

        Class[] types = new Class[]{
            java.lang.String.class, java.lang.String.class, java.lang.String.class,
java.lang.String.class,java.lang.String.class, java.lang.String.class
        };
        boolean[] canEdit = new boolean[]{
            false, false, false, false, false
        };
```

```
        @Override
        public Class getColumnClass(int columnIndex) {
            return types[columnIndex];
        }

        @Override
        public boolean isCellEditable(int rowIndex, int columnIndex) {
            return canEdit[columnIndex];
        }
    };

    public javax.swing.table.DefaultTableModel createTable(String[] col, Object[][] row) {
        return new javax.swing.table.DefaultTableModel(row, col) {

            Class[] types = new Class[]{
                java.lang.String.class, java.lang.String.class
            };
            boolean[] canEdit = new boolean[]{
                false, true
            };

            @Override
            public Class getColumnClass(int columnIndex) {
                return types[columnIndex];
            }

            @Override
            public boolean isCellEditable(int rowIndex, int columnIndex) {
                return canEdit[columnIndex];
            }
        };
    }
    public javax.swing.table.DefaultTableModel wordnetinfo = createTable(new String[]{"Field",
"Path"}, new Object[][]{{"Wordnet Location", "C:\\Program Files (x86)\\WordNet\\2.1\\dict"}});


    @Override
    public void actionPerformed(ActionEvent e)
     {
       int increment = 0;
            if(e.getActionCommand().equals("Register"))
            {
                databaseinfo.addRow(new String[]{getDriver(), getUrl(), getUser(), getPass(),
getTable(), getField()});
                try
```

```
            /*int selected[] = tablexz.getSelectedRows();
            for (int i = selected.length - 1; i >= 0; i--) {
            String url = (String) databaseinfo.getValueAt(i, 0);
            try {
            databaseinfo.removeRow(selected[i]);
            JOptionPane.showMessageDialog(null, "Database Deleted!", "Report",
JOptionPane.INFORMATION_MESSAGE);
            } catch (Exception ex) {
            Logger.getLogger(Settings.class.getName()).log(Level.SEVERE, null, ex);
            JOptionPane.showMessageDialog(this, "Could not erase reference to database " + url);

               } catch (ClassNotFoundException ex) {
                  Logger.getLogger(Settings.class.getName()).log(Level.SEVERE, null, ex);
               }

               Connection conn =
DriverManager.getConnection("jdbc:sqlite:src/DB/semantics.db");;
               Statement stmt = conn.createStatement();
               //stmt.executeUpdate("drop table if exists database;");
               //stmt.execute("create table databases (drivers, urls, users, passwords, tables, fields);");
               //stmt.executeUpdate("create table people (driverss, urlss, userss, passwordss, tabless,
fieldss);");
               String sql = "INSERT INTO databases (drivers,urls,users,passwords,tables, fields) " +
"VALUES ('"+getDriver()+"', '"+getUrl()+"', '"+getUser()+"', '"+getPass()+"', '"+getTable()+"',
'"+getField()+"' );";
               stmt.executeUpdate(sql);
               JOptionPane.showMessageDialog(null, "Inserted", "Report",
JOptionPane.PLAIN_MESSAGE);


               conn.close();
            } catch (SQLException ex) {
               Logger.getLogger(Settings.class.getName()).log(Level.SEVERE, null, ex);
            }

        }
         });
    }
    public void delete() throws ClassNotFoundException, SQLException
    {
       Class.forName("org.sqlite.JDBC");
       Connection con = DriverManager.getConnection("jdbc:sqlite:src/DB/semantics.db");
       Statement stmt = con.createStatement();
       {
       Class.forName("org.sqlite.JDBC");
```

```
    Connection conn = DriverManager.getConnection("jdbc:sqlite:src/DB/semantics.db");;
    Statement stmt = conn.createStatement();
    //stmt.executeUpdate("delete from databases where users = 'SA';");
    ResultSet rs = stmt.executeQuery("select * from databases;");
            ResultSetMetaData mdx=rs.getMetaData();
            int columnsx = mdx.getColumnCount();
            for(int i=1; i <= columnsx; i++)
            {
                    columnNamesx.addElement(mdx.getColumnName(i));
            }
            while (rs.next())
            {
            Vector rowx=new Vector(columnsx);
            for(int i=1; i <= columnsx; i++)
            {
                    rowx.addElement(rs.getObject(i));
            }
                    datax.addElement(rowx);
            }
            rs.close();
            conn.close();
          } catch (SQLException ex) {
            Logger.getLogger(Settings.class.getName()).log(Level.SEVERE, null, ex);
          }
    catch(ClassNotFoundException c)
    {
      Logger.getLogger(Settings.class.getName()).log(Level.SEVERE, null, c);
    }

    DefaultTableModel dat = new DefaultTableModel( datax, columnNamesx);
    JTable tabl = new JTable();
        TableColumn colx;
              for(int i=0; i<tabl.getColumnCount(); i++)
              {
                    colx=tabl.getColumnModel().getColumn(i);
                    colx.setMaxWidth(250);
              }
          tabl.setModel(dat);
    return tabl;
}
public void reload()throws ClassNotFoundException, SQLException
{

    columnNamesx.clear();
    datax.clear();
    Class.forName("org.sqlite.JDBC");
```

```
        Connection conn = DriverManager.getConnection("jdbc:sqlite:src/DB/semantics.db");;
        Statement stmt = conn.createStatement();

        ResultSet rs = stmt.executeQuery("select * from databases;");
                ResultSetMetaData mdx=rs.getMetaData();
                int columnsx = mdx.getColumnCount();
                for(int i=1; i <= columnsx; i++)
                {
                        columnNamesx.addElement(mdx.getColumnName(i));
                }
                while (rs.next())
                {
                Vector rowx=new Vector(columnsx);
                for(int i=1; i <= columnsx; i++)
                {
                        rowx.addElement(rs.getObject(i));
                }
                        datax.addElement(rowx);
                }
                rs.close();
                conn.close();

        DefaultTableModel dat = new DefaultTableModel( datax, columnNamesx);
        JTable tabl = new JTable();
              TableColumn colx;
                  for(int i=0; i<tabl.getColumnCount(); i++)
                  {
                        colx=tabl.getColumnModel().getColumn(i);
                        colx.setMaxWidth(250);
                  }
              tabl.setModel(dat);
              pane.setViewportView(tabl);
              add(pane, BorderLayout.CENTER);
    }
    public Object getTablew()
    {
    return tablew.getValueAt(0, 1);
    }
}
```

```
package semantic_database;
```

```
import edu.smu.tspell.wordnet.Synset;
import edu.smu.tspell.wordnet.SynsetType;
import edu.smu.tspell.wordnet.WordNetDatabase;
import java.sql.*;
import java.util.ArrayList;
import javax.swing.JOptionPane;
import javax.swing.JTable;
import javax.swing.JTextArea;
public class Data_base
{
   public JTable tablex;
   public void search(JTable tab, JTextArea tarea, String word, String path) throws SQLException,
ClassNotFoundException
   {

      Connection con = null;
      Statement stmt = null;
      ResultSet rs = null;
      int rowcount = tab.getRowCount()/3;
       System.out.println(tab.getColumnCount());
       for(int i = 0 ; i < rowcount; i++)
       {
       String driver = (String)tab.getValueAt(i, 0);
       String url = (String)tab.getValueAt(i, 1);
       String user = (String) tab.getValueAt(i, 2);
       String pass = (String) tab.getValueAt(i, 3);
       String table = (String) tab.getValueAt(i, 4);
       String field = (String) tab.getValueAt(i, 5);
       try
       {
            Class.forName(driver);
            if(driver.equals("org.sqlite.JDBC"))
                con = DriverManager.getConnection(url);
            else
                con = DriverManager.getConnection(url, user, pass);
            stmt = con.createStatement();
            StringBuilder sb = new StringBuilder("SELECT * FROM "+ table +" WHERE "+ field
+" IN (");
            boolean added = false;
            for(String s: syns(word,path))
            {
                if(added)
                {
                    sb.append(",");
                }
                sb.append("'");
```

```
            sb.append(s);
            sb.append("'");
            added = true;
          }
          sb.append(")");

          boolean empty = true;
          rs = stmt.executeQuery(sb.toString());
          while(rs.next())
          {
            empty = false;
            if(empty)
            {
            tarea.append("");
            }
            else
            {
              ResultSetMetaData mdx = rs.getMetaData();

tarea.append("*****************************************************************
***********************************************\n");
              tarea.append(mdx.getColumnName(1)  +"\t"  +  mdx.getColumnName(2)+"\t\t"  +
mdx.getColumnName(3) +"     ........From "+driver+ " Database"+"\n");

tarea.append("*****************************************************************
************************************************\n");

              tarea.append(rs.getString(1) +"\t"  + rs.getString(2)+"\t\t" +rs.getString(3)+"\n");
              tarea.append("\n");

            }
            //empty = true;

          }
            //tarea.append("\n");
          rs.close();
          stmt.close();
          con.close();
         }
      catch(ClassNotFoundException c)
      {
        JOptionPane.showMessageDialog(null,  "Database  Driver  not  Found",  "Report",
JOptionPane.ERROR_MESSAGE);
      }
      catch(SQLException e)
      {
```

```
        JOptionPane.showMessageDialog(null, "Database Not Found!", "Report",
JOptionPane.ERROR_MESSAGE);
      }
      }
  }
  public ArrayList<String> syns(String wordForm, String path)
  {
      System.setProperty("wordnet.database.dir", path);
      WordNetDatabase database = WordNetDatabase.getFileInstance();
      ArrayList<String> synonyms = new ArrayList<>();
      if(wordForm.equals(""))
      {
        JOptionPane.showMessageDialog(null, "Enter a word!", "Report",
JOptionPane.ERROR_MESSAGE);
      }
      else
      {
      Synset[] synsets = database.getSynsets(wordForm, SynsetType.NOUN);
      if(synsets.length > 0)
      {
        for(int i = 0; i < synsets.length; i++)
        {
          String[] wordForms = synsets[i].getWordForms();
          for(int j = 0; j<wordForms.length; j++)
          {
            if(!synonyms.contains(wordForms[j]));
            {
              synonyms.add(wordForms[j]);
            }
          }
        }
      }
      }
      return synonyms;
  }
      }
```