Semoogle - An Ontology Based Search Engine

Nooshin Aghajani

Master in Information Systems
Submission date: June 2012
Supervisor: Torbjørn Skramstad, IDI
Co-supervisor: Jingyue Li, Det Norske Veritas (DNV)

Norwegian University of Science and Technology
Department of Computer and Information Science

NTNU – Trondheim
Norwegian University of
Science and Technology

Semoogle - An Ontology-Based Search Engine

Nooshin Aghajani

Master of Science in Computer Science
Submission date: June 2012
Supervisor: Torbjørn Skramstad, IDI
Co-Supervisor: Jingyue Li, DNV

Norwegian University of Science and Technology
Faculty of Information Technology, Mathematics and Electrical Engineering
Department of Computer and Information Science

Preface

This report is a Master's Thesis at Department of Computer and Information Science(IDI), Faculty of Information Technology, Mathematics, and Electrical Engineering at the Norwegian University of Science and Technology(NTNU).

The thesis is continued the specialization project in previous semester(Fall 2011) and is a part of research carried by the DNV (Det Norske Veritas) about designing an ontology-based search engine in the area of safety and security. It contributes towards development of an application named as *Semoogle* for this purpose. The author hopes that this study will provide better insight into search engines, and specifically semantic search engines.

Acknowledgements

I would like to first thank my supervisor, Professor Torbjørn Skramstad for his support and guidance throughout the work with the thesis. Secondly, I would like to acknowledge Dr.Jingyue Li from DNV Research and Innovation for his help and guidance and interesting discussions.

Many people at NTNU have contributed to this work and supported me. Stig Ole Johnsen has helped me with the evaluation and testing of the search process and has given valuable feedbacks and comments. My special thank goes to one of my friends, Aryan Taheri Monfared, who helped me during programming part of this project and his support is valuable for me forever. Also, I would like to acknowledge all my other friends and colleagues, who motivated me in my work by keeping the work atmosphere positive and inspiring.

I also express my gratitude to my family and close friends for tirelessly supporting and cheering on me.

Last – and largest – I would like to thank my beloved husband Hossein for his love and great support in all steps of this thesis, and for showing interest in and patience with my study. This book is dedicated to you.

Nooshin Aghajani
Trondheim, June 2012

Problem Description

Semantic search helps the user queries to be understandable for electronic agents searching. In this way, ontologies play a main role to define the semantic and the relations between user queries.

Therefore, by adding paradigm ontology in order to interpret user queries and their relevant documents in safety and security domain, we enable user with using fewer numbers of terms to gain the desired information. Then, the retrieved results are categorized based on four categories. It is worth to note that, the domain of proposed ontology is flexible to be modified to the other specific domains in other fields of study.

The main objective of this study is to improve the usability and efficiency of semantic search using an ontology in order to enrich the user queries and gain user satisfaction in result search. Four categories include: History, Mechanism, Prevention and Case study. The resulting search application is evaluated versus traditional search engines and, the improvement is demonstrated in the efficiency of new application (Semoogle).

Submitted: 27. June 2012
Supervisor: Professor Torbjørn Skramstad, IDI, NTNU
Co-Supervisor: Dr. Jingyue Li, DNV

Abstract

Much effort is needed to develop a search engine that provides effective and efficient search functionalities with a retrieved high-quality collection of documents. Manual creation of the collection and indexing the terms for searching will require a lot of time and effort. Another alternative is to automate these tasks using software tools. By using semantic search and ontology techniques, the automation of this process is achievable based on user needs. Semantic search is an area of research, which focuses on mapping the meaning of user queries to electronic agents searching. The requirements to define the meaning of user queries are to disambiguate and interpret the meaning of the query and relations between the queries. By enriching the semantic of the queries, the documents in the Web will be meaningful both for the users and search applications.

In semantic search, ontologies play an important role to define the concepts, their properties and the relations among domains. Since the interpretation of concepts is domain specific, ontologies are dependent on the specific domain. According to this argument, the meaning of safety's queries in oil and gas domain can be different in other domains such as medical. Within the scope of this research, safety and security terms are selected as the domain of study and queries are specified for this domain.

In this thesis, we present a prototype for search engine to show how such a semantic search application based on ontology techniques contributes to save time for user, and improve the quality of relevant search results compared to a traditional search engine. This system is built as a query improvement module, which uses ontology and sorts the results search based on four predefined categories. The first and important part of the implementation of search engine prototype is to apply ontology to define the meaning and the relations between the queries in default domain of the study. Next, categorization of the results is carried out in order to improve the quality of result search presentation based on categorization-list. The ontology used in this search engine prototype includes sample of terms in safety and security domain, which is capable to be modified in this domain, or can be substituted by another ontology in the other fields of study. The process is continued by searching the enriched query through the Web using Google interface application search engine. The application uses ranking algorithms to categorize and organize the results of Google search in four categories, i.e. History, Mechanism, Prevention, and Case study. The predefined categories can be substituted to the other categories based on user preferences in other studies using different categorizes.

The evaluation of the implemented prototype reveals that the search engine works as it has been expected. However, due to the fact that the domain is improving dynamically within the research time frame, it is difficult to propose a definite semantic search engine that can cover all user's needs.

Contents

Preface iii

Problem Description v

Abstract vii

Abbreviations xiii

1 Introduction 1
- 1.1 Background 1
- 1.2 Project Goals 1
- 1.3 Approach 3
- 1.4 Expected Results 3
- 1.5 Outline of this Document 5

2 Related Work 7
- 2.1 Semantic Web 7
- 2.2 Information Retrieval 8
- 2.3 Ontology 8
- 2.4 Web Ontology Language–OWL 8
- 2.5 Semantic Search 9
- 2.6 Development of Ontology 9
- 2.7 Related Work 10
  - 2.7.1 Ontology-based Traditional Chinese Medicine 10
  - 2.7.2 INSPIRE ontology approach to critical infrastructures protection 12

3 Search Engine Technologies 15
- 3.1 Search Engine 15
- 3.2 Semantic Search Engine 18
- 3.3 Google Search Engine 19
  - 3.3.1 Functional Process in Google Search Engine 19
- 3.4 Recovery Information Systems 20
- 3.5 Semantic Web Architecture for Search Engine 21
- 3.6 Searching and Browsing Linked Data with SWSE Architecture 23

3.7 Techniques/Algorithms in Search Engine . . . 24
3.7.1 Design of Clustering Algorithm . . . 25
3.8 State-of-the-Art . . . 27
3.8.1 Ontology-based Search Engines . . . 27
3.8.2 *hakia*-Semantic Search Engine . . . 28
3.8.2.1 Technologies in *hakia* . . . 30

4 Design and Implementation of Semoogle 35
4.1 Methodology . . . 35
4.1.1 Computing Term-Weight Based on Term-Frequency (TF) . . . 37
4.1.1.1 Horizontal Weight(Linear) . . . 38
4.1.1.2 Vertical Weight(Heuristic) . . . 39
4.2 Algorithm and Functions of Semoogle . . . 40
4.3 Suggested Ontology . . . 41
4.4 Implementation and Design . . . 45
4.4.1 Frameworks . . . 45
4.4.2 Implementation and Design Strategy . . . 46

5 Evaluation 51
5.1 Evaluation of Data . . . 51

6 Discussion 57
6.1 Advantages of Semoogle . . . 57
6.2 Limitation of Semoogle . . . 58
6.3 Future Work . . . 59

7 Conclusions 61

Bibliography 63

A OWL Ontology in XML 67

B Python Code Listing 81

List of Figures

1.1 The Typical Search Engine Process [1] 2
1.2 Execution Process of the Study 4
1.3 Overview of the Suggested Approach 4

2.1 The upper-level framework of the TCM ontology [2]. 11
2.2 The semantic system of the TCM ontology [2]. 11
2.3 Ontology: Class Hierarchy - General View of INSPIRE Ontology [3]. 13

3.1 High Level Google Architecture [4] 21
3.2 Search Process and Representations [5] 23
3.3 System Architecture of SWSE [6] 24
3.4 Flowchart of Algorithm [7] 26
3.5 Smart Data Continuum [5]. 28
3.6 Query Detection and Extraction - QDEX - System [8] 31
3.7 SemanticRank Algorithm of *hakia* [8]. 32

4.1 Suggested overview of the system 36
4.2 Semoogle search user interface 36
4.3 Semoogle result schema 37
4.4 Suggested Algorithm in order to Category Ranking 38
4.5 Sample case of Category Ranking Algorithm–Horizontal Weight 39
4.6 Sample case of Category Ranking Algorithm–Vertical Weight 40
4.7 Main classes of INSPIRE and Our Ontology [3] 42
4.8 Sub-classes of Security Domain with Source-of-Attack Main Class 42
4.9 Sub-classes of Security Domain with Assets Main Class 43
4.10 Sub-classes of Security Domain with Threats Main Class 43
4.11 Sub-classes of Security Domain with Vulnerability Main Class 44
4.12 Sub-classes of Safety Domain with Physical-assets Main Class 44
4.13 Sub-classes of Security Domain with Safeguards Main Class 44
4.14 Mock-up of enter user query in Semoogle interface 47
4.15 Mock-up of matching user query with existing terms in the ontology list . . 48
4.16 Mock-up of searching expanded query using Google search engine 48
4.17 Mock-up of categorizing the Google search results based on four caregory . 49
4.18 Mock-up of final results for user 49

4.19 Pseudo code for the whole process of implementation strategy 50

5.1 Compare the relevancy of returned URLs for “Virus” query in Semoogle and Google in the domain of study 53
5.2 Compare the relevancy of returned URLs for “Worm Virus” query in Semoogle and Google in the domain of study 54
5.3 Compare the relevancy of returned URLs for “SCADA” query in Semoogle and Google in the domain of study 55
5.4 Compare the relevancy of returned URLs for “Stuxnet Virus” query in Semoogle and Google in the domain of study 55

Abbreviations

Abbreviation	
API	Application Programming Interface
ALS	Amyotrophic Lateral Sclerosis
CO	Commercial Ontology
GERD	Gastroesophageal reflux disease
GUI	Graphical User Interface
HTML	Hyper Text Markup Language
ICT	Information and Communications Technology
IR	Information Retrieval
KB	Knowledge Base
MGED	Microarray Gene Expression Data
NLP	Natural Language Processing
OWL	Web Ontology Language
QDEX	Query Detection and Extraction
RDF	Resource Description Framework
RDFS	Resource Description Framework Ű Schema
SCADA	Supervisory Control And Data Acquisition
SOAP	Simple Object Access Protocol
SWSE	Semantic Web Search Engine
Semoogle	Semantic Google
TCM	Traditional Chinese Medicine
TF	Term-Frequency
UMLS	Unified Medical Language System
UNSPSC	United Nations Standard Products and Services Code
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
VSM	Vector Space Model
W3C	World Wide Web Consortium
XML	eXtensible Markup Language

Chapter 1

Introduction

1.1 Background

By increasing the amount of Web pages and documents on the Web, without using the search engines, it will be almost impossible to obtain relevant and required information from the Web. The huge amount of documents on the Web has caused challenges for searching through the Web and information retrieval. As an answer to this problem, the computer engineers have developed “search engine”. A search engine is an application for searching through documents on the Web where the keywords are given by the user. Also, the engineers implement an interface called browser to search through the documents by search engines. Usually the users will find their information by using some keywords in browsers and the search engine will perform search of those keywords and rank the results by their relevance. Top ten search results have higher quality of content and relevance in most of the search engines. The typical process of a search engine is shown in Figure 1.1

It is difficult for the users to understand the whole process behind the search engine. Understanding existing relationship among terms in a specific domain in order to enter relevant terms and hence having better result search is crucial. Therefore, the user may get into trouble with term matching for the queries to find relevant results. Ontology driven Information Retrieval (IR) has an important role which lets the user to search concepts rather than keywords. This thesis contributes to enrich the search engine by using the ontology in a relevant domain.

1.2 Project Goals

The main goal of this project is to develop and test a search engine prototype. The work is inspired from “INSPIRE ontology” to utilize the semantic search system on a specific domain. The INSPIRE ontology focuses mainly on telecommunication, energy and transportation sectors [3]. The main classes of our ontology are adopted according to the original INSPIRE ontology. The domain of proposed ontology is defined as safety and

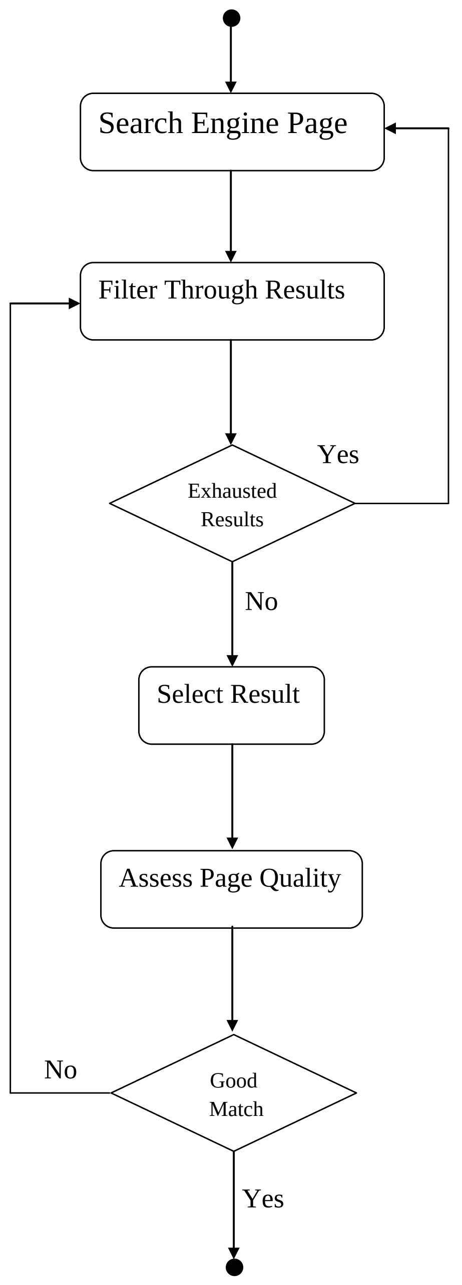


Figure 1.1: The Typical Search Engine Process [1]

security, and is capable of being modified or substituted by other domains of study. The motivation for this project is to investigate the interest of using the ontology to semantic search in search engines.

The objectives that should be achieved stepwise during this research are outlined in the following steps:

- Investigating the related works on search engines in literature survey.
- Building an ontology based on safety and security domain in order to define the relations between the terms.
- Creating Graphical User Interface (GUI) platform to receive the user's query and publish the result search.
- Searching Google using the Google API search application.
- Categorizing the results at predefined categories.
- Evaluating the results of application versus traditional Google search results.
- Concluding the important remarks based on concrete results of evaluation.

The activity diagram depicted in figure 1.2 shows the execution process in order to conduct the above steps.

1.3 Approach

This project aims towards developing a semantic search prototype based on a semantically enriched ontology. The basic knowledge required for the research on semantic enrichment is based on the work done by the author in the project titled "A Systematic Literature Review on Software Security and SCADA Security Taxonomy and Ontology" as a specialization project in fall 2011 [9].

The main part of the project focused on exploring ways to use the semantic enriched ontology for transparently representing linguistic sense. The aim was to draw attention to the conceptual difference of the terms interpretation within the field of study. Figure 1.3 illustrates an overview of the suggested approach.

1.4 Expected Results

The outcome of this thesis is a semantic search engine prototype, which uses an ontology to enrich the user queries. The retrieved results search are categorized based on categorized-list. Through the evaluation of the implemented prototype, the query proposed strategy

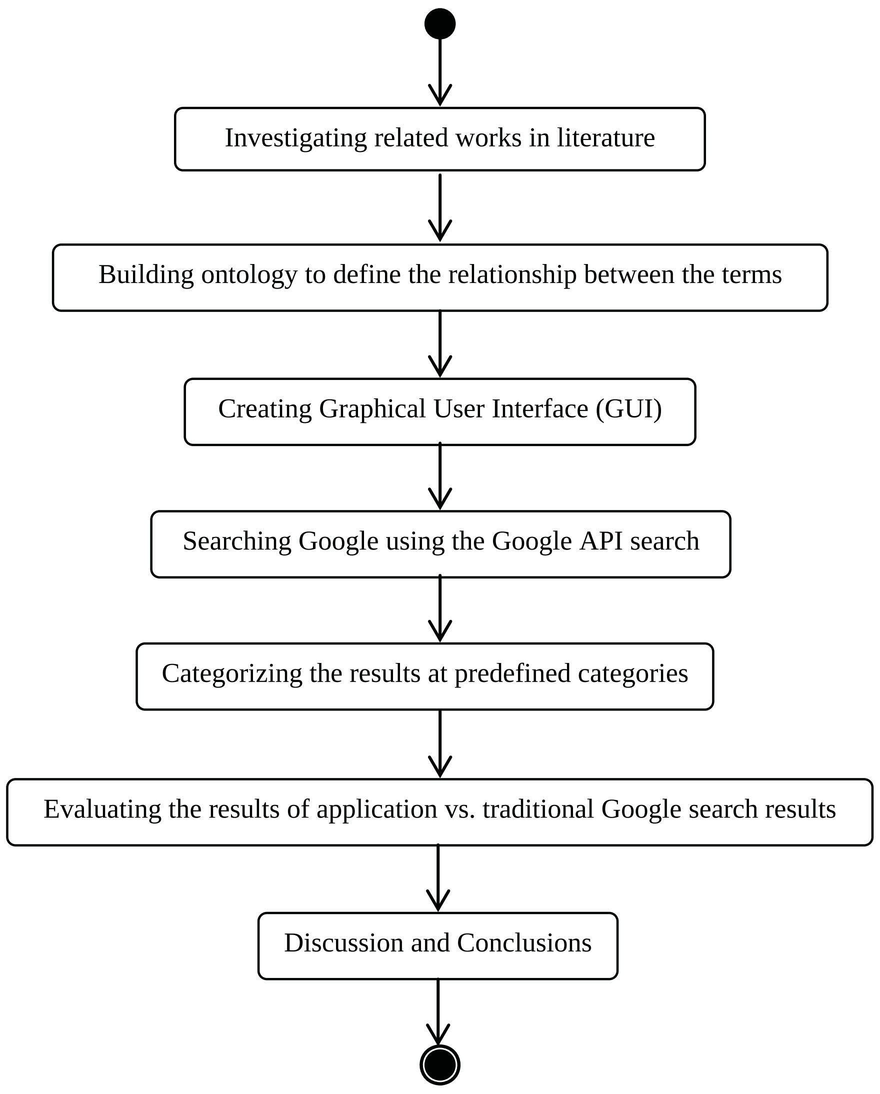

Investigating related works in literature
Building ontology to define the relationship between the terms
Creating Graphical User Interface (GUI)
Searching Google using the Google API search
Categorizing the results at predefined categories
Evaluating the results of application vs. traditional Google search results
Discussion and Conclusions

Figure 1.2: Execution Process of the Study

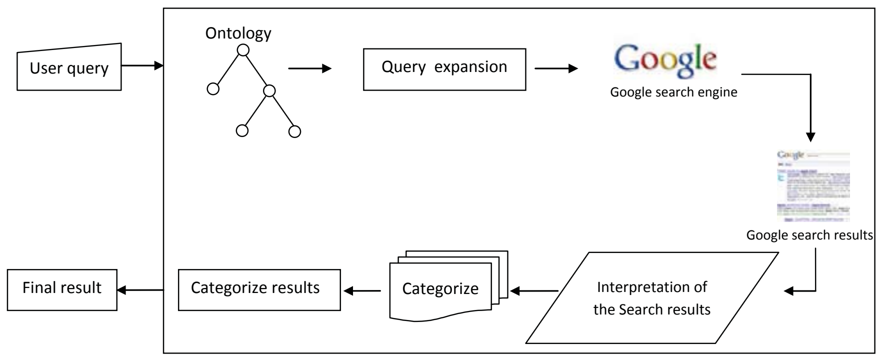


Figure 1.3: Overview of the Suggested Approach

is compared with standard keyword search, shedding light on the possible improvement of search made by our approach with respect to keyword search.

1.5 Outline of this Document

The reminder of the thesis is organized as follows:

Chapter 2 gives an overview of the relevant work, which are prerequisites to understand this project. It defines all the key concepts dealt with in the rest of this thesis.

Chapter 3 includes the literature survey, which prevails the idea and provides required definitions is presented.

Chapter 4 presents the design and implementation of *Semoogle* search engine and will illustrate every stage of the design cycle.

Chapter 5 outlines the evaluation of proposed search engines, which presents an evaluation of *Semoogle* semantic search engine, and compares the usability and efficiency of Semoogle search results with the results of Google search engine.

In **chapter 6**, our findings, advantages and the limitation of the Semoogle search engine are discussed. It presents ideas for further research on this topic.

Finally, **chapter 7** presents conclusions from the results obtained in this thesis.

Chapter 2

Related Work

This chapter provides an overview of the relevant literature works, which are prerequisites to understand concepts used in this thesis. It defines all the key concepts dealt with in the rest of this thesis.

2.1 Semantic Web

"The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in co-operation"

Tim Berners-Lee, James Hendler, Ora Lassila
The Semantic Web, Scientific American, May, 2001 [10]

Moreover, the structure of the web pages is constructed by Semantic Web. The Semantic Web is not only a separate part of the Web, but also it is an extension of the current one. To present the semantics in the Semantic Web, the semantics must be processable for machine. The machines should have access to the set of information, which is structured to make automatic reasoning possible [10].

Semantic Web is the next generation of the World Wide Web, which includes the machine processable meta data describing the meaning of the Web resources. In simple words, Semantic Web enables the search engines and software agents to interpret web contents in the same way as a human being, while it is fast and accurate [11]. The Semantic Web is not only a Web of document, but also a Web of relation between resources denoting real world objects, i.e., objects such as people, places and events.

The ontologies have a main role in the Semantic Web vision. The main goal of ontologies is to give the conceptual description of a domain and, models the domain with their concepts, relations and properties. Defining the complex reasoning will be possible by the ontology concept.

2.2 Information Retrieval

The Information Retrieval has variety of concepts. However, in an academic field of study, Information Retrieval might be defined as:

Information Retrieval (IR) finds out material (usually documents) of an unstructured nature (usually text) that satisfies an information required within large collections (usually stored on computers). In first place, the Information Retrieval concept began with scientific publications and library records, but spread very soon to the other forms of content, particularly those related to information professions, such as journalists, lawyers, and doctors in order to fulfill the users' information needs [12].

2.3 Ontology

Ontology is a basic concept in many different fields, and has different meaning across different domains, e.g. library science, philosophy, and knowledge representation.However, in general, it can be the basis for communication ground across the gaps between people and computers [13].

The philosophers presented the meaning: *existence* and *being* for the ontology. Today's use of ontology on the web has a different slant from the previous philosophical notions. In computer science, an ontology is a description of knowledge about a domain of interest, the core of which is a machine-processable specification with a formally defined meaning [14].

Some uses of simple ontologies are as controlled shared vocabulary such as search engines, authors, users, databases, programs/agents, site organization and navigation support, browsing support, search support and many other roles can have ontologies in different concepts [15].

2.4 Web Ontology Language–OWL

Both OWL and RDFS are W3C recommended standards for the modeling of ontologies. OWL is on the top of RDFS, which provides limited expressive means and it is not designed to present the complex knowledge. The main point to design OWL was not only to find a reasonable balance between expressivity of the language , but also efficient reasoning, i.e. scalability. In order to give the user a choice between different degrees of expressivity, three sub-languages of OWL - species of OWL- have been designed expressed as: OWL Full,OWL DL and OWL Lite [14]. OWL Full contains OWL DL and OWL Lite. OWL DL contains OWL Lite. OWL Lite is less expressive than OWL Full and OWL DL ,but OWL Full is very expressive.

2.5 Semantic Search

In general, Semantic Search attempts to extend and improve traditional search results. It is based on Information Retrieval (IR) technology by using data from the Semantic Web interpretation. Semantic Search can be considered as an application of the Semantic Web to search. The main characteristic of the semantic web which is different from other generations of the World Wide Web is matching the concepts and their meaningful variations. The traditional IR is implemented based on the occurrence of words in documents. Under this context, the search engines, e.g. Google, search with information about the hyper link structure of the Web. Thereby, the structured and machine understandable information about a wide range of objects on the Semantic Web may offer possibilities for improving traditional search [16].

2.6 Development of Ontology

In recent years, development of ontology has become very common in World Wide Web in different industries such as medicine, electrical, oil and gas, etc. Most of the usage of the ontologies in industry is to categorize and organize the products to be ready to use. In fact, the motivating factors to develop ontologies are listed as following [17]:

- To share common understanding of the structure of information among people or software agents
- To enable the reuse of the domain knowledge
- To make domain assumptions explicitly
- To separate the domain knowledge from the operational knowledge
- To analyze the domain knowledge

Ontology is required to develop a semantic search engine. The main advantage of the semantical ontology is to describe the domain knowledge, for example the relations between categories in different views can be defined. Also, ontologies can be used to create semantics more accurate annotations in terms of the domain knowledge. By ontologies, the user will be able to express the queries more precisely and unambiguously, which leads to better precision and recall rates. Finally, through ontological class definitions and inference mechanisms, such as property inheritance, instance-level metadata can be enriched semantically [18].

The more common goal of developing ontologies is sharing common understanding of the structure of information among people/software agents [17].

In order to clarify the discussion, we present works related to ontologies, being relevant to our work in next section.

2.7 Related Work

In this section, we present an overall overview of the relevant approaches and related works in the area of semantic ontology.

2.7.1 Ontology-based Traditional Chinese Medicine

One of the effective practical approaches of ontology is defined from the field of medical terminology. Traditional Chinese medicine has clinical effectives and characteristics in disease diagnosis and treatment, Chinese medical formula and drug therapies. The goal of this ontology is to resolve ambiguities in polysemous terms and organizing a very large scale domain. In order to encode domain knowledge in a reusable format, well designed ontology will be required. This ontology which is named as "Traditional Chinese Medicine Ontology" is a cooperation of many disciplines in this area. It contains more than 10,000 classes and about 80,000 instances.

Traditional Chinese Medicine (TCM) has an old history as a complex medical science. Based on the advantages of ontologies, they play a critical role as a key technology for enabling semantic-driven knowledge processing in order to share and integrate knowledge for critical systems. In [2], authors propose a framework for TCM. They argue that the proposed ontology provides a strong foundation that leads toward a vision of domain knowledge management for knowledge-intensive domains such as TCM. Figure 2.1 shows this framework. There are some available ontologies in this field which can be useful as a road map for the other fields. For example UMLS (Unified Medical Language System) is one of the several on-line large-scale ontologies in biology and medicine which is used for integrating biomedical terminology. Also, MGED ontology for microarray experiment, and Gene ontology is used for gene product.

Such as the other ontologies, in TCM ontology, the major work is to define the concept in hierarchy order and to find the relations between those concepts for modeling the ontology. From the semantic structure point of view, the TCM structure consists of two components: *concept modules* and *semantic module*, which are shown in Figure 2.1. "The concept module contains content class that represents the concrete domain knowledge of the TCM discipline. The semantic module concerns the basic semantic type and semantic

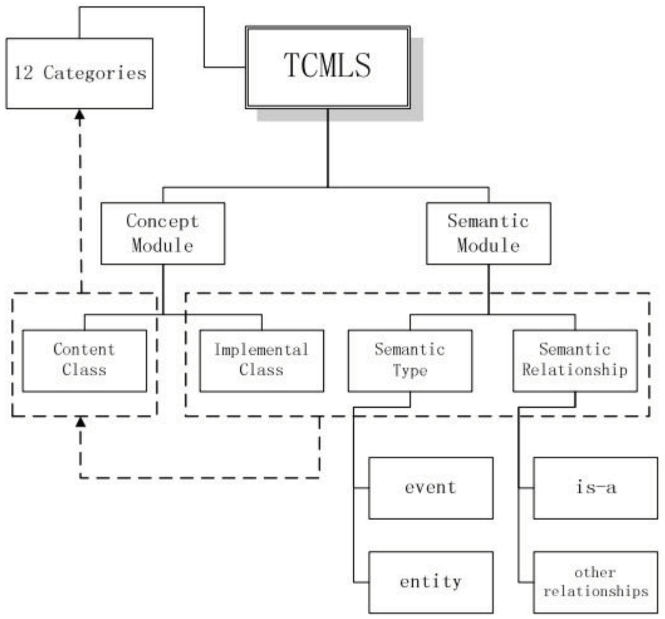


Figure 2.1: The upper-level framework of the TCM ontology [2].

relationship of content class, which form ontology classes themselves." (see figure 2.2).

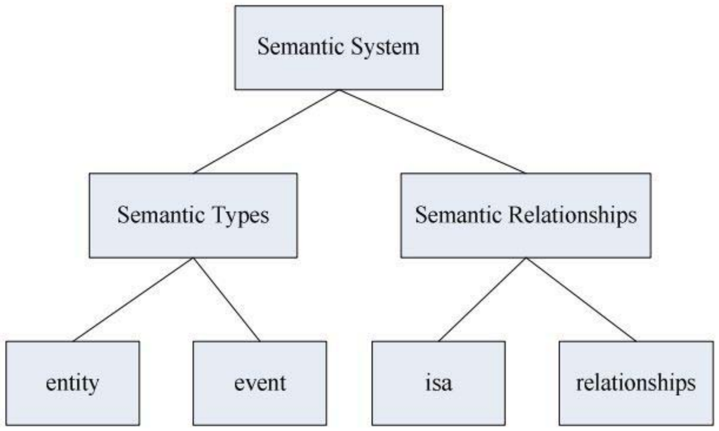


Figure 2.2: The semantic system of the TCM ontology [2].

Further details such as concept structure and their categories of this ontology can be found in [2].

2.7.2 INSPIRE ontology approach to critical infrastructures protection

INSPIRE is a two-year STREP EU Project within the Joint Call FP7-ICT-SEC-2007-1 (Critical Infrastructure Protection). The project started in November 2008 and ended in October 2010. INSPIRE research challenges were as follows [3]:

- Analysis and modeling of dependencies between critical infrastructures and underlying communication networks
- Designing and implementing traffic engineering algorithms to provide SCADA traffic with quantitative guarantees
- Exploiting peer-to-peer overlay routing mechanisms for improving the resilience of SCADA systems
- Defining a self-reconfigurable architecture for SCADA systems
- Development of diagnosis and recovery techniques for SCADA systems

From the application point of view, the INSPIRE project focuses mainly on telecommunication, energy and transportation sectors. INSPIRE project aims at developing innovative automated reconfiguration techniques such as P2P (Peer-to-Peer) and multi-agent reconfigurations. In particular, INSPIRE project aims at investigating the characteristics of P2P for the purpose of strengthen SCADA systems against a cyber-attack. INSPIRE aims at improving the definition of a distributed diagnosis and reconfiguration framework. A diagnostic system can be able to understand the nature of errors occurring in the system, judge whether and when some actions are necessary, and finally trigger the recovery/reconfiguration/repair mechanism to perform the adequate controlling actions. In INSPIRE ontology, vulnerabilities are considered as a property of a network security system, and SCADA resources and components have weak points named as vulnerabilities. These vulnerabilities can be exploited by threats, leading to attacks. The security system within this ontology can be considered as a form of classification with properties and relationships among security issues. The main classes of INSPIRE ontology compose of the following concepts [3]:

- Asset(anything that has value to the organization)
- Vulnerabilities (include weaknesses of an asset or group of assets which can be exploited by threats)
- Threats (potential cause of an unwanted incident which may result in harm to a system or organization)
- Source of attacks
- Safeguards (practices, procedures or mechanisms that reduce vulnerabilities)

Each class has its own properties. Properties describe relations, dependence of one class on another or can represent some attributes. The interconnections between particular Vulnerabilities, and between Vulnerabilities and Assets are modeled based on properties. The relation between Vulnerabilities and Safeguards is modeled by the Safeguard's properties [3]. The proposed hierarchy of the ontology classes is presented in Figure 2.3:

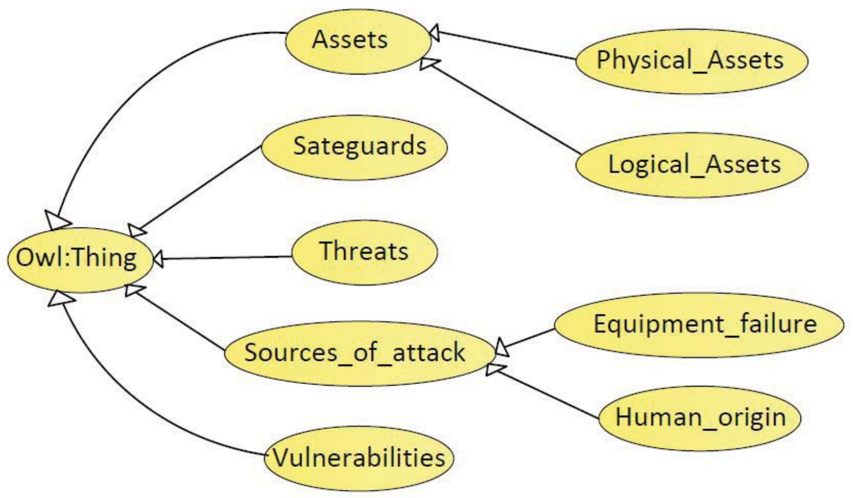


Figure 2.3: Ontology: Class Hierarchy - General View of INSPIRE Ontology [3].

Next chapter focuses on search engines such as Google and the relevant techniques and technologies used in the current search engines.

Chapter 3

Search Engine Technologies

This chapter presents a review of the relevant literature on search engine technologies and provides required definitions used in this study..

3.1 Search Engine

One of the most useful and high-profile resources on the Internet are search engines. Since many documents were spread on the net and it was difficult for the users to find their relevant information, the search engines have appeared to help users to find the information that they need and retrieve the data from the Web. According to the territory of search engines, optimization of the search engines based on semantic Web plays an important role to develop the semantic search engines. By optimization of the search engines, search results can be improved and more relevant document can be retrieved than the traditional results search. This optimization can be done in a semantic search format in order to improve the results search based on users' needs.

The fundamental differences between traditional and semantic search engines, are described in below [19].

Traditional search engine:

- Keyword(s) are entered in the engine prompt.
- There are not any relationship between the keywords (no polysemy and no synonymy).
- The terms don't have meaning for the system.
- The stop words such as “a”, “and”, “is”, “on”, “of”,... are not taken into account and will be removed in the search process.
- Unable to handle the long-tail queries.

Semantic search engine:

- You are allowed to raise questions.
- There are relationships between words (synonym and polysemy are meaningful).
- The keywords have meaning for the system.
- The stop words are taken into account.
- Able to handle the long-tail queries.

As mentioned in the above items, the traditional search engines focus on keywords where there are no relationship between the keywords or terms. Such aspect imposes limitation to traditional search engines. Whereas in semantic search engines, there is relation among keywords providing a specific meaning of the keywords to the system.

Moreover, when the user searches for long-tails queries, semantic search engine can easily support it because of the pre-defined relation between queries via ontology. Traditional search engines, however, are not capable of capturing this affect due to lack of relation between keywords.

In the traditional search engines, there is a routine function for finding the documents, which is matched by user's query. Search engines match queries against an index that they create. The index consists of the words in each document, and then pointers are added to their locations within the documents. This is called an "inverted file". A search engine or IR system comprises four essential modules [20]:

- **Document processor:** This module prepares processes and inputs the documents, pages, or sites for which users search. The document processor performs some or all of these steps: Normalizes the document stream to a pre-defined format; Breaks the document stream into desired retrievable units; Isolates and meta-tags sub-document pieces; Identifies potential indexable elements in documents; Deletes stop words; Stems terms; Extracts index entries; Computes weights; Creates and updates the main inverted file against which the search engine searches in order to match queries to documents.
- **Query processor:** This module has seven possible steps. The steps in query processing are as follows (with the option to stop processing and start matching indicated as "Matcher"): Tokenize query terms; Delete stop words; Stem words; Create query representation; Expand query terms; Compute weights. Though a system can cut these steps short and proceed to match the query to the inverted file at any of a number of places during the processing. Document processing shares many steps with query processing. More steps and more documents make the process more expensive for processing in terms of computational resources and responsiveness. However, the longer the wait for results, the higher the quality of results. Thus,

search system designers must choose what is more important to their users - time or quality. Publicly available search engines usually choose time over very high quality, having too many documents to search against.

- **Search and matching function:** Searching the inverted file for documents to meet the query requirements is referred simply as “matching”. It is typically a standard binary search, no matter whether the search ends after the first two, five, or all seven steps of query processing. When the computational processing requires simple, unweighted, non-Boolean query, matching is far simpler than when the model is an Natural Language Processing (NLP)-based query using a weighted Boolean model. It also follows that the simpler the document representation, the query representation, and the matching algorithm, the less relevant the results, except for very simple queries, such as one-word, non-ambiguous queries seeking the most generally known information [20].

  Having determined which subset of documents or pages matches the query requirements to some degree, a similarity score is computed between the query and each document/page based on the scoring algorithm used by the system. Scoring algorithms rankings are based on the presence/absence of query term(s), term frequency, tf-idf (term frequency Ű inverse document frequency), Boolean logic fulfillment, or query term weights. Some search engines use scoring algorithm which is not based on document contents, but rather, on relations among documents or past retrieval history of documents/pages [20].

- **Ranking capability:** After computing the similarity of each document in the subset of documents, the system presents an ordered list to the user. The sophistication of the ordering of the documents again depends on the model the system uses, as well as the richness of the document and query weighting mechanisms. For example, some search engines only require the presence of any alpha-numeric string from the query occurring anywhere in any order in a document. These search engines would produce a very different ranking than the ones which performs linguistically correct phrasing for both document and query representation and that utilized the proven tf-idf weighting scheme [20]. Whenever the search engine determines rank, the ranked results list goes to the user, who can then simply click and follow the system’s internal pointers to the selected document/page. More sophisticated systems can go even further at this stage and allow the user to provide some relevance feedback or to modify their query based on the results they have envisaged. If either of these are available, the system will then adjust its query representation to reflect this value-added by feedback and re-run the search with the improved query to produce either a new set of documents or a simple re-ranking of documents from the initial search. Those interested in further detail should turn to [21].

  There are some key features in helping to retrieve a good representation of documents/pages. Some of these key features, which determine the upper documents, are more relevant, such as: Location of terms; Link analysis; Popularity; Date of Publication; Length; Proximity of query terms; Proper nouns.

While users focus on “search”, the search and matching function is only one of four modules. Each of these four modules may cause the expected or unexpected results that consumers get when they use a search engine.

Based on that, the engineers create the Web crawling and index systems in search engines. Web crawling is mainly used to create a copy of all the visited pages for later processing by a search engine that will index the downloaded pages to provide fast searches. Crawling is required not only for gathering the information on documents, but also it keeps the information up to date. At the first step, computer engineers created the automatic search engine which returns too many non-relevant information and junk results for the users, since its search was exclusively based on keyword matching. Afterwards, engineers moved to create Google, which is extract of *googol* (10100), with the purpose of covering what they have dreamed of a search engine [4]. Compared to the all Web crawling, Google has the fastest Web crawling system so far. Google is a large scale search engine and the goal of this search engine is “to address many of the problems, both in quality and scalability [4].”

In general, search engines consist of many components in their architecture, which works together but have own functionality. The existing architectures are based on architecture from 1998, when Sergey Brin and Lawrence Page from Stanford University posed the architecture for the search engine and their functionalities. In the first generation of search engines, the query terms are matched against an index of all terms in the documents. A result page with ranked links to all the retrieved documents is presented to the user [5].

3.2 Semantic Search Engine

In semantic search engine, semantic information about Web resources is stored and semantic Web is able to solve complex queries and considering the context where the Web resource is targeted. By gathering the technologies and techniques of semantic Web, the search engines attempt to improve the results of searching. In general, the whole process of semantic search engine is divided into the following parts [19]:

1. The user question is interpreted, and the relevant concepts from the sentence are extracted.
2. Set of concepts is used to build a query that is launched against the ontology,
3. The results are presented to the user.

In order to apply semantic search, we have to be familiar with the ontology term. The term of ontology comes from philosophy, that is “the science of what is, of the kinds and structures of objects, properties, events, processes and relations in every area of reality” [18].

There are some Websites in the Internet with examples of ontology, such as Yahoo! Categories, DMOZ Directory, Amozon.com product catalogue, WordNet, GO (Gene Ontology) (www.geneontology.org), Unified Medical Language System (UMLS), and UNSPSC - terminology for products and services. There are many kinds of ontologies which can be divided in three main types [18]:

- Terminological ontologies where concepts are word senses and instances are words,
- Topic ontologies where concepts are topics and instances are documents,
- Data-model ontologies where concepts are tables in a data base and instances are data records (such as in a database schema).

The benefits of semantic search is to make it easy to locate relevant information to the user’s subject of interest, also to save the user a lot of time to read non-relevant Web pages. The other advantage is that semantic search engine can handle long-tail queries. Since the semantic search uses an ontology to infer information about objects, it can tackle the limitations of keyword searches. This will possibility help the semantic search system to correctly identify objects [19].

3.3 Google Search Engine

Google is the most famous and powerful search engine in the world at the present time. It was introduced in 1996 by Larry Page and Sergey Brin. Google is a dominant search engine out of all major search engines (it is also known as Web directory) [22]. This section focuses on the functional process in the Google search engine.

3.3.1 Functional Process in Google Search Engine

In this section, we describe the functional components in Google search engine to give the readers an overview of how the search engine in Google works. As figure 3.1 shows, the web crawling (downloading of web pages) is performed by several distributed crawlers. Lists of URLs to be fetched by the crawlers are sent by an URL server. The fetched Web pages are then sent to the storeserver where the web pages are compressed and stored into a repository. There is an ID number (docID) for every web page which is assigned whenever a new URL is parsed out of a Web page. The indexing function is done by the indexer and the sorter. The indexer is in charge of a number of functions including reading the repository, uncompressing and parsing the documents. Every document is converted into a set of word occurrences called hits. The word, its position in document, an approximation of its font size and capitalization is recorded by hit. These hits are then distributed into a set of barrels, creating a partially sorted forward index. Another important function of the indexer is parsing out all the links in each web page and storing important information about them in an anchors file. Based on information stored in this file, it is determined where each link points from and to, and the text of the link. The anchors file are now read by the URL resolver. Also relative URLs are converted into absolute URLs and in turn into docIDs by the URL resolver. This is done through putting the anchor text into the forward index, associating it with the docID that the anchor points to. The URL also generates a database of links which are pairs of docIDs. The links database is used to compute PageRanks for all the documents.

The sorter takes the barrels, which are sorted by docID. This is a simplification and resorts them by wordID to generate the inverted index. Leading to little temporary space for this operation. The sorter also produces a list of wordIDs and offsets into the inverted index. A program called DumpLexicon takes this list together with the lexicon produced by the indexer and generates a new lexicon to be used by the searcher. The searcher is run by a web server and uses the lexicon built by DumpLexicon together with the inverted index and the PageRanks to answer queries [4].

Since ranking the relevant pages is a vital task for any search engine, information retrieval is the main aspect of any search engines [23]. Hence, performance of a search engine will be highly important to evaluate whether the search engine is better than the other search engines or not. In [23], performance is categorized into two perspectives: 1) User Perspective 2) Search engine perspective. The former describes the user's needs and whether search engine covered the user expectations. The latter is the search engine foundation by respect to the speed, which means the time elapsed to find the result for user's query.

3.4 Recovery Information Systems

The first generation of search engines consists of three types of systems for information recovery. They are directories (indexes), search engines and meta-search engines.

Second generation of search engines is a developed version of the first generation and is employed currently. Nowadays, pages with similar features are clustered into same bunch. One example is Google Scholar, which provides scientific articles for the users without forcing the user to change the browser.

The difference between the first generation and the second generation is highlighted by an following example. In the first generation, for searching an image file, the user had to enter the required keywords into the browser to retrieve the respected results on a

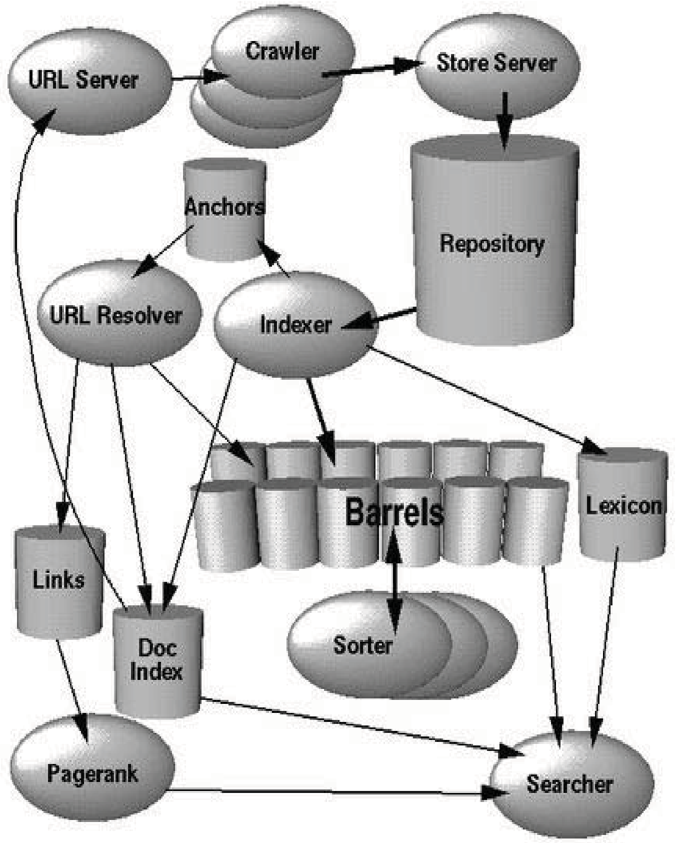


Figure 3.1: High Level Google Architecture [4]

Website which contains the information about that image. That means the user had to look through the Website to find the image. In second generation, this process has become much easier. The user can find the result in milliseconds by some few keywords, and they do not need to look for the information in the Web pages. The reason is that the search engines has become powerful and every textual document or image is understandable by search engines. The search engine can directly have access to the address of the images or every words of a textual document.

3.5 Semantic Web Architecture for Search Engine

Across growing the rate of the Web, search engine technology has developed. Google covers both this aspect very well. However, some valuable results may be neglected because of the semantics of the keywords, consequently Google is not able to use the semantics of the keywords for finding more relevant results. However, in recent years, Google is improved by providing direct answers to prose queries matching certain common templates [6]. Google is a keyword-based search engine. So, it is not suitable for complex information gathering tasks. Semantic-based subject is the most interesting subject in the area of search engines today. For example, some commercial companies have developed their search engines by semantic technologies which their search engines are able to search the query by conceptual meaning instead of just keywords matching.

There is a limitation in Google structure which is related to lack of structure in HTML documents. Since machine interpretability has limitation for using generic markup- tags mainly concerned with document rendering and linking [6].

Now a days, the basic architecture is as old version of Sergey Brin and Lawrence Page's, but there is an ongoing discussion on semantic and syntactic query searching based on basic architecture [5]. Some commercial companies have achieved these technologies (semantic searching) in their search engines, but Google has not introduced semantic searching technologies in its search engine so far. The idea behind the semantic search is to make the search engine intelligent. This idea will help the user to find more relevant results just in millisecond. Semantic search applications offer mechanisms to deal with the content of the documents rather than matching keywords and trying to capture and find the keywords in documents and queries. A term used as a keyword may refer to several concepts, on the other hand, a concept may be used from the meaning of several terms [5].

Figure 3.2 describes a high level overview of the components used in semantic search engine architecture. This structure of the main components, according to [5], includes: indexing, querying, searching and ranking, result presentation, and result navigation. In this figure, documents have their own indexes which means that they have semantic structures, and queries have reference to the concepts and their relationships. Also, the retrieved documents have semantic concept. Moreover, navigational links can be improved on classification and refinement of the results.

In this architecture, which is proposed in [5], there is a possibility to have similar search processes at semantic layers as well. Textual documents are indexed which means they have semantic structure, and therefore every document is understandable by the computer. Moreover, the queries, which are used for searching documents, will be understandable and meaningful by the computer. Also, the relationship between documents is considered in this structure. The communication between every document and their relationships are based on ontology and the taxonomies, which are used to facilitate this possibility for the semantic Web.

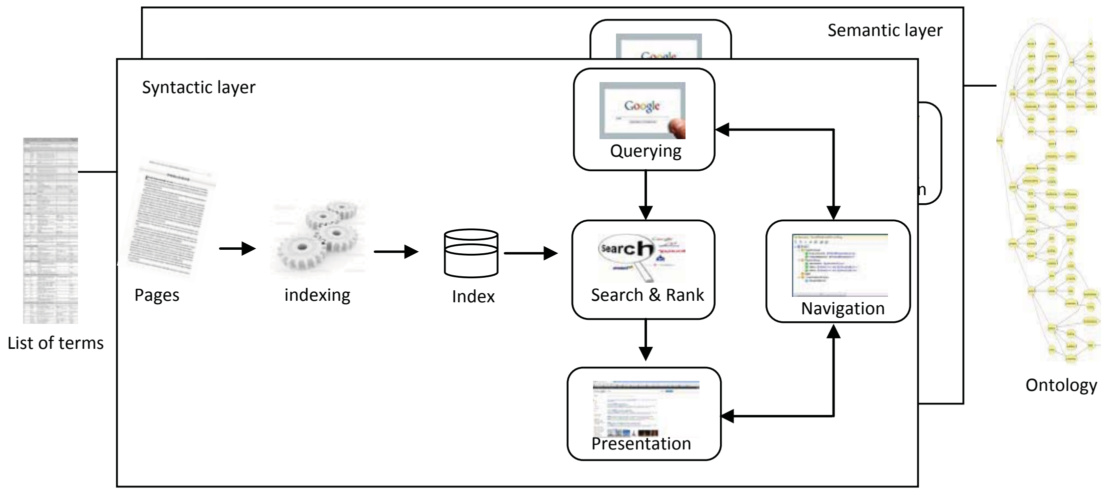


Figure 3.2: Search Process and Representations [5]

3.6 Searching and Browsing Linked Data with SWSE Architecture

In [6] the authors claimed that the realization of Semantic Web Search Engine(SWSE) implies two main research challenges:

1. The system must scale to large amounts of data,
2. The system must be robust in the face of heterogeneous, noisy, impudent, and possibly conflicting data collected from a large number of sources.

In the design and implementation of SWSE, semantic Web standards and methodologies are not naturally applicable in such an environment. Unlike frequent document-centric Web search engines, SWSE operates over structured data and holds an entity-centric perspective on search, which means instead of return links to documents containing specified keywords, SWSE returns data representing the real-world entities. In SWSE structure, the users are allowed to specify keyword queries in an input box and responds with a ranked list of result snippets. Thereby, results refer to the entities instead of documents. Figure 3.3 shows a high level overview of SWSE architecture by showing the main components. This architecture loosely follows that of traditional HTML search engines. In details, this figure shows the pre-runtime architecture of SWSE system showing the components involved in achieving a local index of RDF Web data amenable for search.

The functional process of this system is as follows [6]:

- The crawler accepts a set of seed URIs (Uniform Resource Identifier) and retrieves a large set of RDF data from the Web.

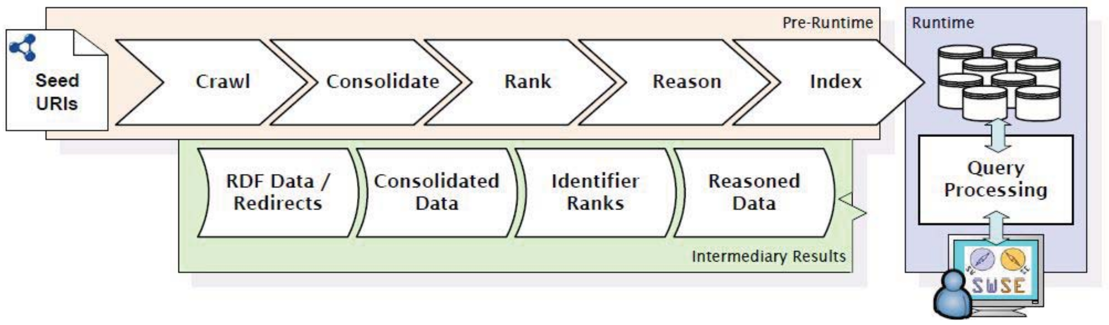


Figure 3.3: System Architecture of SWSE [6]

- The consolidation component aims to find synonymous (i.e., equivalent) identifiers in the data, and canonicalises the data according to the equivalences found.
- The ranking component performs links-based analysis over the crawled data and derives scores indicating the importance of individual elements in the data (the ranking component also considers URI redirections encountered by the crawler when performing the links-based analysis).
- The reasoning component materializes new data which is implied by the inherent semantics of the input data (the reasoning component also requires URI redirection information to evaluate the trustworthiness of sources of data).
- The indexing component prepares an index which supports the information retrieval tasks required by the user interface.

Subsequently, the query-processing and user-interface components service queries over the index built in the previous steps.

This architecture has some similarity with traditional search engines such as crawling, ranking and indexing data, but there is some other components related to handling RDF data, named as consolidation component and the reasoning component [6].

3.7 Techniques/Algorithms in Search Engine

The best way for evaluating the search engines is measuring the quality of performance of the results of the search engines. Since information on the Web should be up to date and major changes of the system should be tested quickly, efficiency of crawling and indexing is very important for search engines. Especially for Google, the most important tasks are crawling, indexing, and sorting. By improving techniques and algorithms, search engines can have better performance and more relevant results in minimum time [4].

One of the problems regarding information retrieval is predicating the relevant documents as a result search. This kind of decisions is usually made by ranking algorithms. The ranking algorithms are the core of information retrieval systems [21].

In addition to the ranking algorithms and developing ranking algorithms, which improve the "quality" of the answer set, there is another way to group (cluster) documents based on the terms that they contain and to retrieve from these groups using a ranking methodology.

Automatically clustering the results can be very helpful for users to choose the search results which they really need. Recently, engineers achieved success in order to clustering the results of search engines. These achievements are divided into three parts [7]:

First- Using the advantages of Hits and PageRank algorithms [24], as mentioned in section (3.3.1), they analyzes the hyperlinks among the Web pages and cluster the pages with the similar features into the same bunch.

Second- This approach, which is proposed by Huajun Zeng [25], is for applying the Text Clustering into clustering search engine results. By this method, the engineers will be able to encode each result into a vector set and then use text-clustering algorithms to cluster the results.

Third- This approach, which is proposed by Po-Hsiang Wang [26], by taking advantage of users' response presenting an algorithm to optimize the order of clustering results. The larger number of users interested in the specific results, the higher merit order the results will be placed in.

New retrieval algorithms are based on new retrieval models. Once a new algorithm for information retrieval is designed, the performance of the algorithm should be tested and measured [21].

3.7.1 Design of Clustering Algorithm

Most clustering algorithms are based on the document Vector Space Model (VSM), which is used for ranking the Web documents. The VSM algorithm creates a space in which both documents and queries are represented by vectors. Although these algorithms are easy to implement, the meaning of each cluster is not clear to user. If the similarity of search results is larger than a certain threshold, they merely vectorize the search results, and cluster the search results to the same bunch. So, the results will be inaccurate, also the algorithm is inefficient. Consequently, they are seldom applied for commercial search engines.

Authors in [7] proposed an approach, which ensures that with a lower complexity algorithm, they can have more efficient clustering. We suppose that the topic of document will represent the frequency terms in the document. So, based on this frequency the documents can be clustered in different groups by different topics. The similarities among different documents are computed based on the appearances of the terms in the documents and documents are clustered using heuristic rules. This is shown in figure 3.4.

In every search engine, the input is represented by a very large number of features which many of them are not needed for predicting the labels. Feature selection is the task of choosing a small subset of features that is sufficient to predict the target labels well. Feature selection reduces the computational complexity of learning and predicting algorithms and saves the cost of measuring non selected features [27].

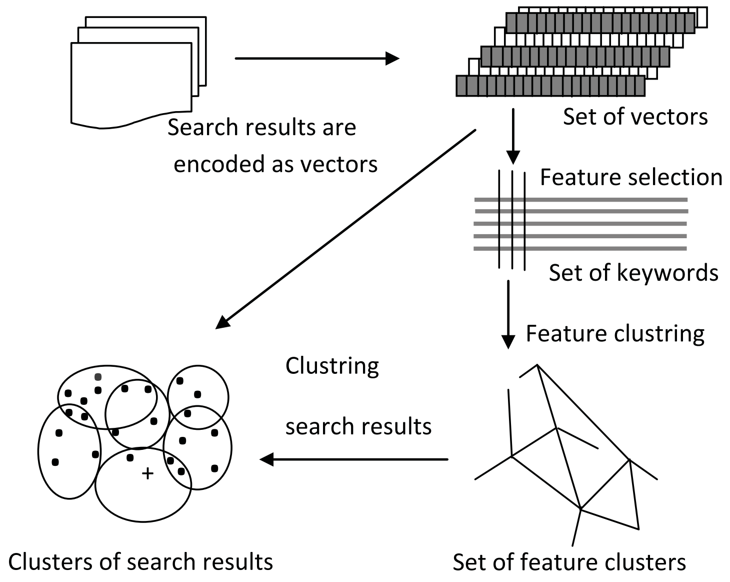


Figure 3.4: Flowchart of Algorithm [7]

In figure 3.4, each search result generated by the search engine for a certain query is assumed to be an independent cluster unit. Higher occurrence frequency of one word in a result search indicates its higher importance [7]. These high-frequency words can be regarded as the key characteristics of a document and hence they can be used to feature the documents.

Two examples of characteristics of a single feature are: the keyword's frequency inside a search result as well as the frequency of the search result that the keyword occurs in the search result collection. Moreover, two different are associated to each other through certain relationships, such as the number of search results in the result search collection and the frequency in the results search in which both features appear. Feature selection, feature clustering, weight computation and search result clustering will be applied according to the clustering algorithm, after the selected search results have been encoded as vectors [7].

Among many measures of retrieval effectiveness, there is also another traditional way for measuring the performance of a search engine, which is called *recall* and *precision*. Recall is "the number of relevant documents retrieved divided by the total number of existing relevant documents that should have been retrieved", while precision is "the number of relevant documents a search retrieves divided by the total number of documents retrieved." In other words, when the concentration is on recall, the question is based on "Have all the relevant documents been retrieved?" However, when the concern is on precision, the focus is according to this question: "Are all the retrieved documents relevant?" This combined measure of recall and precision is proposed and developed by van Rijsbergen (1979) [28]. Eq. (3.1)

$$recall = \frac{|\{relevant\ documents\} \bigcap \{retrieved\ documents\}|}{|\{relevant\ documents\}|} \tag{3.1}$$

$$precision = \frac{|\{relevant\ documents\} \bigcap \{retrieved\ documents\}|}{|\{retrieved\ documents\}|} \tag{3.2}$$

It is clear from the Eqs. (3.1 and 3.2) that in recall, all relevant documents are extracted, while in precision all retrieved documents are considered.

In the point of optimistic view, a perfect model of search engine is the one that finds the precise documents always on the Web for the user.

3.8 State-of-the-Art

This chapter presents the recent achievements in the area of search engine techniques.

3.8.1 Ontology-based Search Engines

At the first stage, when Tim Berners-Lee and their group decided to spread the documents on the Web, they did not think about the semantic Web. Thus, after the volume of the documents increased on the Web they found that searching through the Web for finding specific document is a very hard task. Since the machine cannot understand the semantic of the documents which is available on the Web, Tim Berners-Lee and their groups decided to define the documents and their meanings for the machine. They thought of the semantic Web as a replacement of the current Web. This decision required to add semantic to all information on the Web. By this act, data on the Web become smarter and will be accessible in searching process [5]. Figure 3.5 shows the data progress along a continuum of increased intelligence.

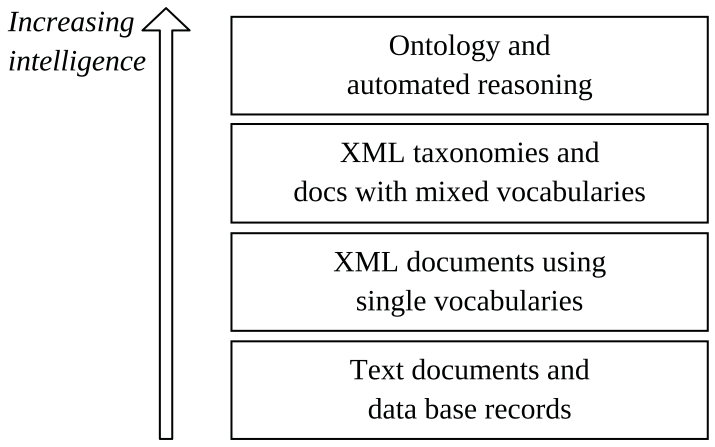

Increasing intelligence
Ontology and automated reasoning
XML taxonomies and docs with mixed vocabularies
XML documents using single vocabularies
Text documents and data base records

Figure 3.5: Smart Data Continuum [5].

In final stage which is shown by arrow, new data can be deduced from existing data without human involvement. In this stage, the data is smart and have their relationships with other data which make it easily understandable by machine and human. In between, ontologies play an important role in the semantic Web. Ontology presents common understanding of a domain the relevant terminologies of that domain and the relationship between the concepts.

By learning ontology and creating the ontology for a specific domain, there is a possibility to semi automate or even fully automate creation of ontologies from representative domain texts. The previous works in this area named as text mining and computational linguistics for extracting terms from the text. After that the focus moved to extracting the relationships and properties using statistical methods, such as association rules and single phrasal searches.

Nevertheless, the number of available ontologies has increased today and evaluation of ontologies becomes difficult. There is a great need to develop the semantic aspects in order to select the best ontology for a specific domain.

3.8.2 *hakia*-Semantic Search Engine

This section introduces a semantic Web search engine on the Web called **hakia**. This semantic search engine is designed by a private institute, Headquartered in Manhattan, New York, in 2004 [8]. The aim of this search engine is to focus on the quality of the results in all segments including Web, News, Blogs, *hakia* Galleries, Credible Sources, Video, and Images. Among these segments, News, Blogs, Credible, and galleries are processed by *hakia*'s proprietary core semantic technology called QDEXing. Web, video, and images are processed by *hakia*'s SemanticRank technology using third party API feeds. Moreover, they launched three new products in 2011 named as AeroHakia, NewPubmed, and MoodTRADE.

More information about this semantic search engine is presented in [8]. Also, there are 10 issues which represent the differences between semantic indexing and keyword indexing are considered in *hakia* search engine structure. These issues are discussed in following.

10 points of differentiation from keyword indexing and semantic indexing [8]

1. **Handling morphological variations:** A semantic search engine is expected to handle all morphological variations, e.g. tenses, plurals, etc. On a consistent basis. In other words, the results should not change whether you type "improving quality of life" illustrates that *hakia* results contain morphological variations of the query.

2. **Handling synonyms with correct senses:** A semantic search engine is expected to handle synonyms, e.g. cure, heal, treat,.. etc. In the right context and with correct word senses. For example, the word "treat" can mean doing social favors as in trick and treat, which would not be correct in the medical sense. The example query "is there a cure for ALS" shows that *hakia* brings results with synonyms with the correct senses. The level of sense disambiguation in a semantic search engine is a sign of its progress.

3. **Handling generalizations:** A semantic search engine is expected to handle generalizations, e.g. disease = GERD, ALS, AIDS, etc. Where the user's query is expressed in generalized form, the result is expected to be specific. The example query "Which disease has the symptom of coughing?" brings a result set in *hakia* such that GERD is recognized by the system as the specific answer.

4. **Handling concept matching:** Perhaps the most challenging functionality among all, a semantic search engine is expected to recognize concepts and bring relevant results. Usually, the depth of this capability is increased in verticals of operation, and it would be unrealistic to expect coverage in all subjects under the sun. The example query "what treats headache" brings a result set in *hakia* including concept matching such that migraine belongs to the concept of headache in the medical sense.

5. **Handling knowledge matching:** Very similar to the previous item, a semantic search engine is expected to have embedded knowledge and use it to bring relevant results , e.g. swine flu = H1N1, flu=influenza. Knowledge match and concept match are similar in principle, yet different in practice in the way the capability is acquired. The example query "swine flu virus" brings a result set in *hakia* where these kinds of matches are visible.

6. **Handling natural language queries and questions:** A semantic search engine is expected to respond sensibly when the query is in a question form , e.g. what,

where, how, why, etc. Note that a "search engine" is different than a "question answering" system. Search engine's main task is to rank search results in the most logical and relevant manner whereas a question answering system may produce a single extracted entity. The example query "how fast is swine flu spreading?" brings a result set in *hakia* to shed light to this capability.

7. **Ability to point to uninterrupted paragraph and the most relevant sentence:** Unlike conventional search engines where a query points to documents, semantic search is expected to do much better. A query must point not only to documents but also to relevant sections of them. This eliminates 2nd search where the user is supposed to open the documents to find the relevant sections.

8. **Ability to enter queries freely, no special formats like quotes, or Boolean operators:** When entering a query, special format requirements are becoming a thing of the past even with today's non-semantic search engines. These formats perform gross approximations to substitute meaning match, and are signs that unveil the underlying weaknesses of the search technology.

9. **Ability to operate without relying on statistics, user behavior, and other artificial means:** A semantic search engine is expected to bring relevant results by analyzing the content of a page (or document), its source, authors, and the credibility of the results in response to a query. Relying on link referrals, user behavior/tagging, and other artificial means may produce good results when such data is available, but are outside the realm of semantic search. By not relying on artificial input, semantic search technology is more universal, applicable to any situation especially to enterprise documents and real-time content where such data does not exist.

10. **Ability to detect its own performance:** When there is no semantic content analysis in a search algorithm, relevancy scores refer to artificial measurements, e.g. how popular the page is. A semantic search engine is expected to produce a relevancy score reflecting the degree of meaning match. This capability provides flexibility for the developers to apply meaning thresholds. Accordingly, the search engine can understand its poor performance to automatically flag areas of improvement that is needed.

3.8.2.1 Technologies in *hakia*

In this section, some aspects of the technology used in *hakia* search engine and the basic elements of this search engine is explained:

- Query Detection and Extraction(QDEX): This is a new way to analyze and store page content in terms of knowledge bits inverted by *hakia*. It is a replacement of the inverted index method most commonly used today. The need for such replacement

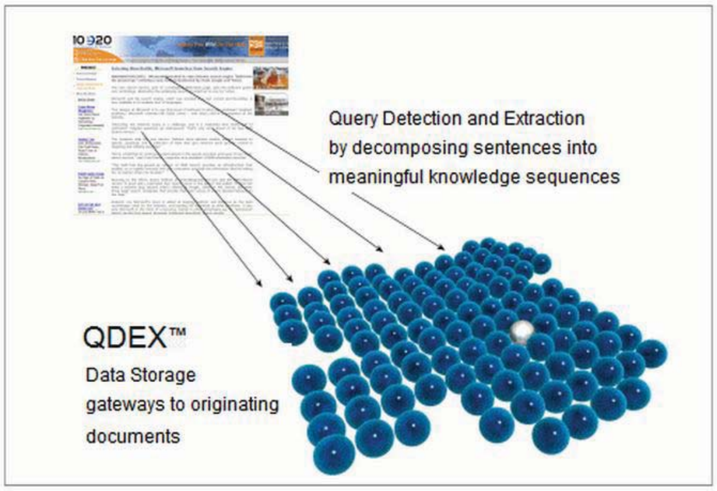

Query Detection and Extraction
by decomposing sentences into
meaningful knowledge sequences
QDEX™
Data Storage
gateways to originating
documents

Figure 3.6: Query Detection and Extraction - QDEX - System [8]

emerges when semantically rich data must be handled at high speeds for Web search or in enterprise search [8].

Figure 3.6 shows the QDEX system analysis of the entire content of a page. Then, the QDEX algorithm extracts all possible queries that can be asked to this content, at various lengths and forms. These queries (sequences) become gateways to the originating documents, paragraphs and sentences during the retrieval mode. Note that this process takes place off-line before the users enter any query.

Decomposing content in this way provides a great flexibility to utilize semantically rich data and to deploy multiple-thread processing of equivalent queries. Otherwise, deep semantic analysis is virtually impossible over vast amount of textual data.

- SemanticRank Algorithm of *hakia*: It is comprised of innovative solutions from the disciplines of Semantics, Fuzzy Logic, Computational Linguistics, and Mathematics. The purpose of the algorithm is to rank search results in the order relevancy.

  Figure 3.7 shows a pool of relevant paragraphs coming from the QDEX system for a given query terms. Then, the final relevancy is determined by the SemanticRank algorithm based on an advanced sentence analysis and concept match between the query and the best sentence of each paragraph. Among other things, morphological and syntactic analyses are also performed. In this operation, there is no keyword matching or Boolean algebra involved.

- Commercial Ontology: The term "understanding" mainly refers to a cognitive process in the human brain where bits and pieces of information are collectively analyzed and identified to belong to a certain category of the world knowledge.

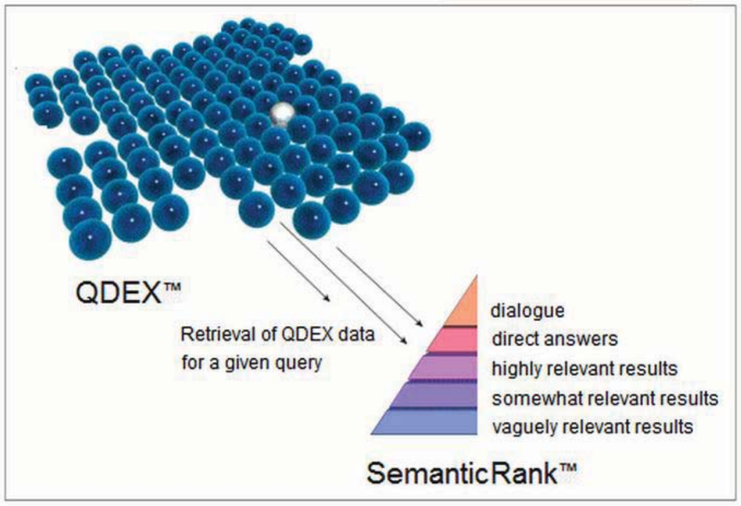


Figure 3.7: SemanticRank Algorithm of *hakia* [8].

Computerized understanding is being able to emulate this complex and mostly unknown cognitive process using algorithms. No reasonable person should expect that computerized understanding can be anything close to its biological counterpart for the next thousand years.

Therefore, computerized understanding only makes sense to emulate particular functions of the human brain for a given, highly specific task. Otherwise, if we attempt to emulate the entire process it may easily become a thousand-year project.

The term "commercial ontology" (CO) is used to emphasize this mere fact that the development of *hakia*'s ontology focused on designing a structure suitable for the search function, using manageable and reasonable body of the world knowledge. Most other ontologies developed to date, especially those originating from academic research, seem to be thousand year projects attacking the general problem by encapsulating the entire world knowledge. Hence, their application to practical problems suffers from lack of resources.

CO can be viewed as a building with a number of floors. The very first floor contains a map of objects (nouns) including tangible things as well as conceptual nouns. The top of the ontology is the parent of all things. For a given word, phrase, sentence, or text, the first layer of CO identifies the elements and where they belong to in the world knowledge of nouns.

The second floor contains a map of events (verbs or actions). This hierarchy contains

> fragments, groups, and clusters not exhaustively connected to each other. Instead, different senses of events are connected to the first floor to point where they belong to in the world of nouns. For example, one of the senses of the event “testing” is “stress testing” which belongs to materials in the first floor. Since the same concept can be applied to human body, it will also be connected to the relevant nouns in the first floor. However, the sense of “testing knowledge” in school will point to different parts of the noun trees in the first floor.
>
> The three-dimensional structure described above can be quickly visualized using the ongoing “building” analogy. Additional floors include time, place, and other dimensional information that are connected to the floors below.
>
> Analyzing a text using CO for search and retrieval purposes requires that the query and text share common locations in the building. This simple approach is flexible and versatile in identifying meaning match with high accuracy. It is also easily decomposable for queries asking for location, time, identity, cause, and method due to separate floor structures. Disambiguation is also handled in CO by populating the ontology with sequences instead of single word definitions. Accordingly, the unit of computation in CO is not restricted to single words.

Studies in this chapter together with literature review from chapters 1 and 2 indicate a need for semantic search engine for different domains and industries. This means that each industry sector will have to design and develop a semantic search engine according to its own database terminology, which defines the relation among that industry specific terms. As a result, similar terms in each industry sector will have different meaning in other sectors. For example, security in oil and gas has a different meaning from security in medical industry. Such a need for development of different terms based on their application has resulted in creating different ontologies. Based on this context, development of ontologies and search engines has become very important.

In this project, we aim at designing a semantic search engine similar in some aspects to *hakia* semantic search engine but with a different domain in ontology and different features. The user is looking for safety and security terms in a specific domain, where the terms are categorized in four different categories. Establishment of prototype design and implementation in semantic search engine will be part of this project. Next chapter will explain the design and architecture as well as implementation of the proposed semantic search engine, which named Semoogle.

Chapter 4

Design and Implementation of Semoogle

Approach

Based on results obtained from literature review discussed in previous chapters, there is a need for designing a semantic search engine in the area of safety and security. The first step to design this semantic search engine is to develop ontology based safety and security. The role of this ontology is defined based on communicating between terms within this domain. In simple words, each term in this domain has its specific meaning and it may have a different meaning outside this domain, which is not the interest of this study. Semantic search engines provides a definition to the user queries using ontology, and it results in producing broader results search based on user needs.

Base on this approach, we proposed a model involves architecture design, implementation, execution and testing the system. The aim of this chapter is to describe the design and implementation details of the suggested model. The first section gives an overview of the system. The second section will present the ontology used for this system, which is followed by detailed description of the approach.

4.1 Methodology

This section will present the methodology of the Semoogle search engine and will illustrate every stage of the design cycle. The figure 4.1 shows the overview of the whole system.

The first step of the process starts when the user enters the query in Semoogle user interface illustrated in figure 4.2. We assume that the user query is available in the ontology list of Semoogle. Therefore, the query will be expanded in the next step based on the proposed ontology. If user query is not available in the ontology list, the search process will be continued by using Google results as a normal search. The role of proposed ontology is to define the relations and the properties among the terms and entered

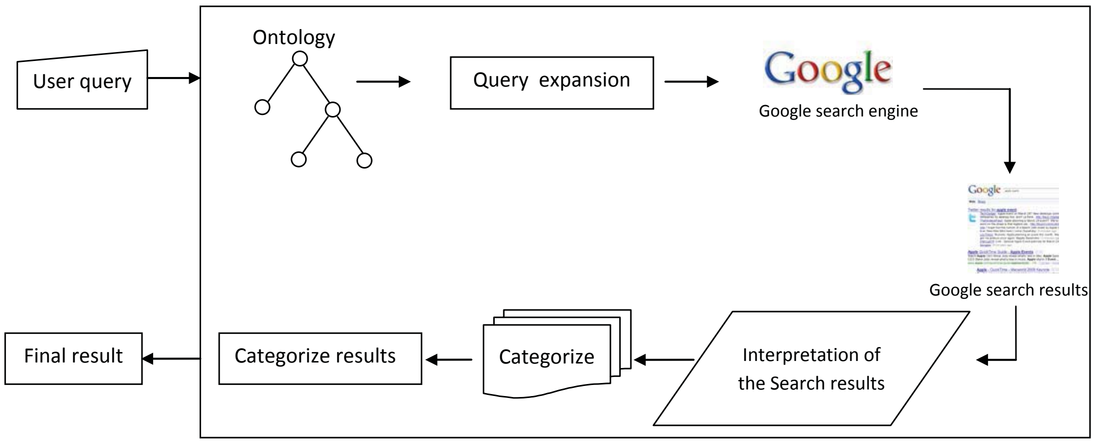


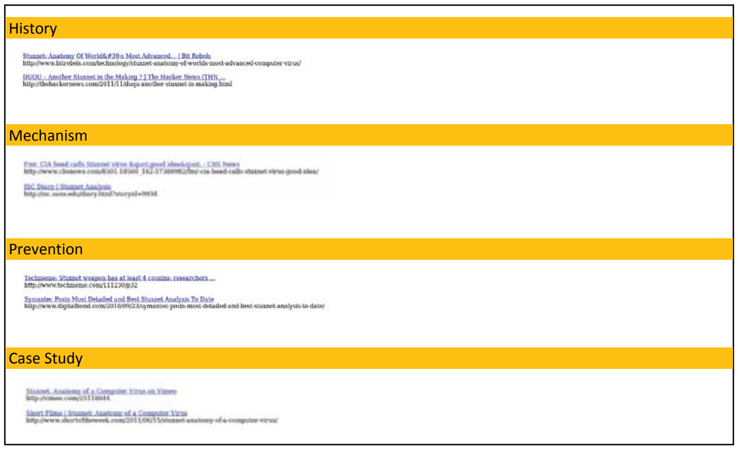
Figure 4.1: Suggested overview of the system

query. When the user enters a query, which is available in this ontology, the query will be expanded according to the related terms selected by the ontology. Query expansion is targeted to improve the user query in order to present a broader search results. In other words, when the query is expanded by the relevant terms in the ontology list, the results of the search will be highly relevant to the domain of study.

Figure 4.2: Semoogle search user interface

In the next step, the semantically enriched query will be transferred to the Google search engine. Google will return many *URLs*. The retrieved results search will be interpreted based on the title, and will be allocated to pre-defined categories. This mechanism will improve the search results presentation to the user. The categorization of the results is based on four categories and the title of the Google search should be in the range of these four categories in order to be allocated to the correct category. These four categories are “History”, “Mechanism”, “Prevention”, “Case study”. The categorized results will be published in the page shown in figure 4.3.

Since this application contains the semantic search based on Google search engine, we have chosen “**Semoogle**” name for our application. All the programing steps are implemented by python programing language and the Graphical User Interface (GUI) is based on Django Web programing language, which is user friendly and easy to understand and test.



History

Stuxnet: Anatomy Of World&#39;s Most Advanced... | Bit Rebels
http://www.bitrebels.com/technology/stuxnet-anatomy-of-worlds-most-advanced-computer-virus/

DUQU - Another Stuxnet in the Making ? | The Hacker News (THN ...
http://thehackernews.com/2011/11/duqu-another-stuxnet-in-making.html

Mechanism

Fmr. CIA head calls Stuxnet virus &quot;good idea&quot; - CBS News
http://www.cbsnews.com/8301-18560_162-57388982/fmr-cia-head-calls-stuxnet-virus-good-idea/

ISC Diary | Stuxnet Analysis
http://isc.sans.edu/diary.html?storyid=9934

Prevention

Techmeme: Stuxnet weapon has at least 4 cousins: researchers ...
http://www.techmeme.com/111230/p32

Symantec Posts Most Detailed and Best Stuxnet Analysis To Date
http://www.digitalbond.com/2010/09/23/symantec-posts-most-detailed-and-best-stuxnet-analysis-to-date/

Case Study

Stuxnet: Anatomy of a Computer Virus on Vimeo
http://vimeo.com/25118844

Short Films | Stuxnet: Anatomy of a Computer Virus
http://www.shortoftheweek.com/2011/06/15/stuxnet-anatomy-of-a-computer-virus/

Figure 4.3: Semoogle result schema

4.1.1 Computing Term-Weight Based on Term-Frequency (TF)

The use of ontologies to overcome the limitations of keyword-based search has been put forward as one of the motivations of the Semantic Web since its emergence in the late 90's. One way to show the semantic aspect of search engine is to acquire a user query and map it to the formal ontology, expand the query against a knowledge based(KB) ontology, and return tuples of ontology values that satisfy the query, which has been implemented in our search engine prototype. This process will be continued in our approach by transferring the ontology values to the Google search engine followed by return of the relevant documents by Google. In order to match the most relevant URLs to the appropriate category out of four categories(History, Mechanism, Prevention, Case study), it is suggested to use a combination of two assumptions for computing term-weight in the matched terms, which includes the following weighting steps:

1. Horizontal Weight(Linear)
2. Vertical Weight(Heuristic)

Figure 4.4 shows the overview of main idea behind the weighting terms. This method proposes a decision-making modelling concept, which covers one of the most important findings in term-weight computing and term-frequency. This model/idea is the outcome of a group discussion and brainstorming result. In the first phase of this brainstorming, we make a decision about horizontal weight, which is illustrated in figure 4.5 –every terms of title which are existed in the category-list have to get exact rank number equal to 1– and this idea is expand consequently to the vertical weight by specifying fixed weight to each category and ranking existing terms in the title which are available in the category list. The more details of ranking model is explained below.

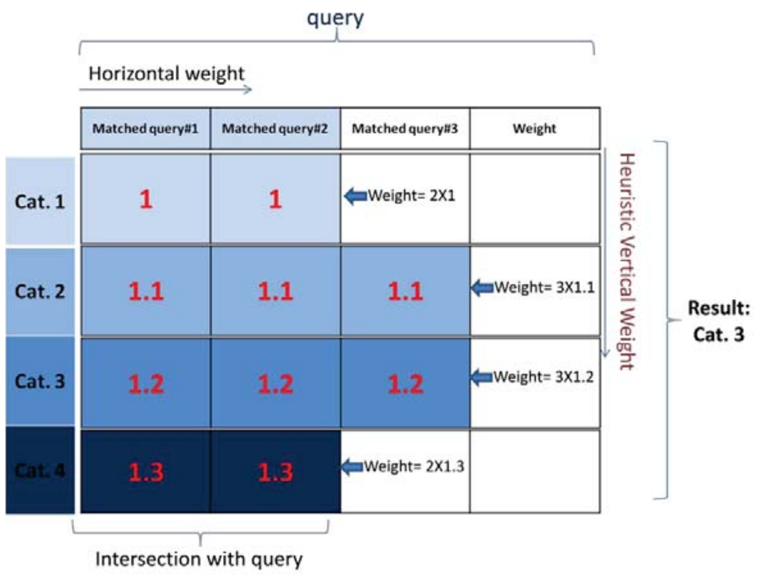
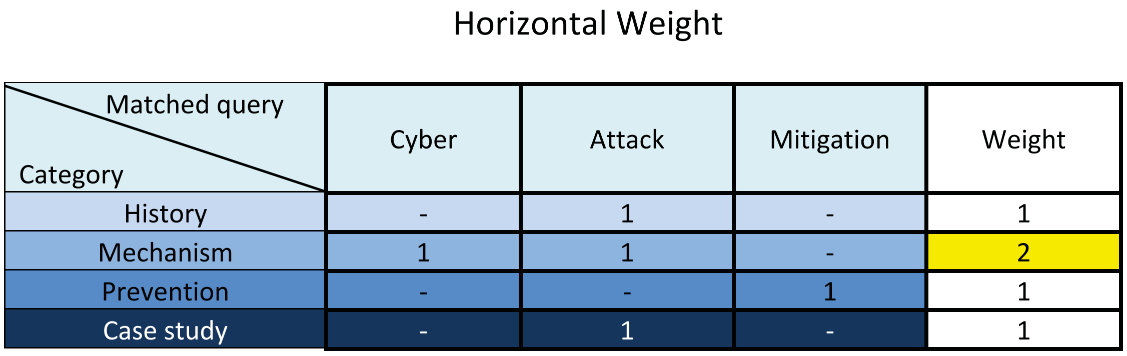


Figure 4.4: Suggested Algorithm in order to Category Ranking

4.1.1.1 Horizontal Weight(Linear)

It is the fact that for a certain title of Google results, it is required to specify the title to one of the existing categories in Semoogle. If the number of the terms, which are matched by one category is more than the other ones, that title will be specified to the category with the most number of the frequent keywords. This is the idea behind the "horizontal weight" based on counting the most frequent keywords, which are in the title. Figure 4.5 shows a sample of horizontal weight as an example. In figure 4.5, three samples of matched query is shown, which are "Cyber", "Attack" and "Mitigation". These terms are founded in the title of URLs in result search. Therefore, it is time to ranking those terms based on their relevancy to each category. Since both "Cyber" and "Attack" terms are relevant to the mechanism category, they will get the higher priority than others and the given title will be transfered to the mechanism category.



Horizontal Weight

Matched query / Category	Cyber	Attack	Mitigation	Weight
History	-	1	-	1
Mechanism	1	1	-	2
Prevention	-	-	1	1
Case study	-	1	-	1

Figure 4.5: Sample case of Category Ranking Algorithm–Horizontal Weight

4.1.1.2 Vertical Weight(Heuristic)

Once the URLs are sorted based on horizontal weight, we found URLs redundancy in different categories. In order to avoid redundancy of URLs in different categories, we decide to apply vertical weight. This idea is outcome of group brainstorming and used to the system. Vertical weight applies only to those of URLs with the equal number of ranking. According to this idea, each exist term in the category list given exact weight. For example *history* is given exact rank 1.0, *mechanism* is given exact rank 1.1, *prevention* is specified with exact rank 1.2 and *case study* is given exact rank 1.3. *Case study* category has the higher priority than the other categories in this application. This is due to the higher importance of case studies events in safety and security field. These fixed ranks are arbitrary and it is flexible to change the priority to other categorize by changing the exact ranks which are specified to each category.

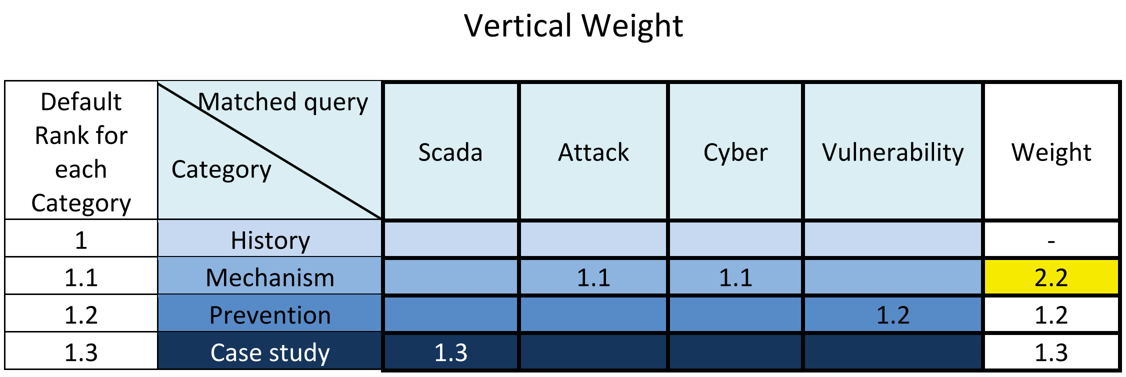
$$(cat1 < cat2 < cat3 < cat4)$$

By specifying the exact weight to each category, we stopped URLs redundancy. This method named vertical weight in this model. Figure 4.6 shown the fixed weight for each category.

Then number of common keywords is multiplied by the weight factor. Figure 4.6 shown an example of vertical weight given for samples of data sets.

This method is a proper solution for avoiding URLs redundancy in the situations where there are two titles with the same number of frequent keywords.

The proposed methodology for ranking between the relevant links is sufficiently flexible to be modified based on the importance and priority of the categorizes. For example, we assigned the maximum rate to the final category, which is "Case study". It can be



Vertical Weight

Default Rank for each Category	Matched query / Category	Scada	Attack	Cyber	Vulnerability	Weight
1	History					-
1.1	Mechanism		1.1	1.1		2.2
1.2	Prevention				1.2	1.2
1.3	Case study	1.3				1.3

Figure 4.6: Sample case of Category Ranking Algorithm–Vertical Weight

assigned to the other categories according to their importance.

The other advantages of this methodology is that it can be used in other areas of study as well, which means by modifying the ontology list to the relevant areas of the study, the same application can be applied to the query under search. In this thesis, we are using the terms in the area of safety and security domain.

4.2 Algorithm and Functions of Semoogle

This section covers the methodology in algorithm format including functionality of the system. Based on second section of Chapter 3 presenting the whole process of semantic search engine, the functional process of Semoogle search engine is described in this section. We establish the Semoogle search engine based on the following steps:

1. The user enters the query and in the initial step, Semoogle breaks down the query into single words.

2. The extracted single words are compared to the list encompassing designated ontology and relevant concepts (ontology list).

   (a) If the user query is not existed in proposed ontology list, the search process will be continued by following step 4 as a normal Google search.

   (b) If the user query is existed in proposed ontology list, the query is expanded based on the available terms on the list.

3. The expanded query will be searched using Google search engine.

4. The results of the search will be extracted from Google search result in order to be processed in the next step.

5. The results will be processed to compare the Google results search with the category-list, which is designed in such a way to cluster the search results in four categories, i.e. “History”, “Mechanism”, “Prevention” and “Case Study”.

6. Subsequently, the categorized search results will be shown in Semoogle output user interface page.

4.3 Suggested Ontology

Ontologies are used to capture knowledge about the domain of interest. An ontology describes the concepts in the domain and also the relationships that hold between those concepts. Different ontology languages provide different facilities [29]. An ontology should be used to solve the semantic issues and share knowledge with and among computers. An ontology supports sharing information and knowledge, defining the relationships between different resources, understanding of the domain and representation of conceptualization using several languages such as RDF, OWL, etc [30]. The most recent development in standard ontology languages is OWL from the World Wide Web Consortium (W3C)[1], which has been described in chapter 2.

The premise for the ontology implemented for this prototype is inspired from INSPIRE ontology, which is used to increase the security and protection in critical infrastructures. This ontology is designed by Michal Choras [3].

The INSPIRE Project focuses mainly on telecommunication, energy and transportation sectors [3]. The main classes of our ontology are according to the original INSPIRE ontology, and we substitute the subclasses based upon our requirements for safety and security. The main classes of this ontology are: “Assets”, “Safeguards”, “Threats”, “Source-of-attack” and “Vulnerabilities” [3]. We establish our ontology based on these main classes and the sub-classes of these main classes, which are according to user’s requirements in safety and security domain. Figure 4.7 shows the main classes of INSPIRE ontology, which are the basis to design our ontology as well.

Figure 4.7 represents abstraction of an ontology that has been created in OWL-DL language using the Protégé application. According to these main classes in INSPIRE project and the main requirements of safety and security domain, we determine relevant sub-classes for each main class. Figures 4.8, 4.9, 4.10, 4.11, 4.12 and 4.13 represent our preferences as sub-classes for suggested ontology.

In these figures, the essential terms in safety and security domain are presented. Except Physical-assets class, which is based on safety terms, the other main classes are in the area of security domain. All these classes need to be improved by the ontologist, who

[1]http://www.w3.org/TR/owl-guide/

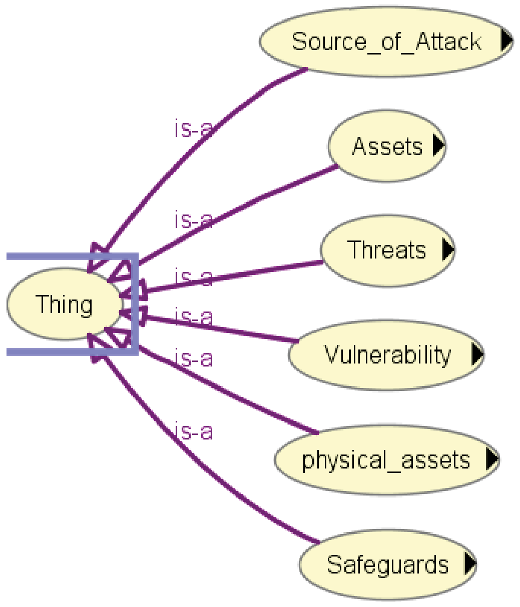


Figure 4.7: Main classes of INSPIRE and Our Ontology [3]

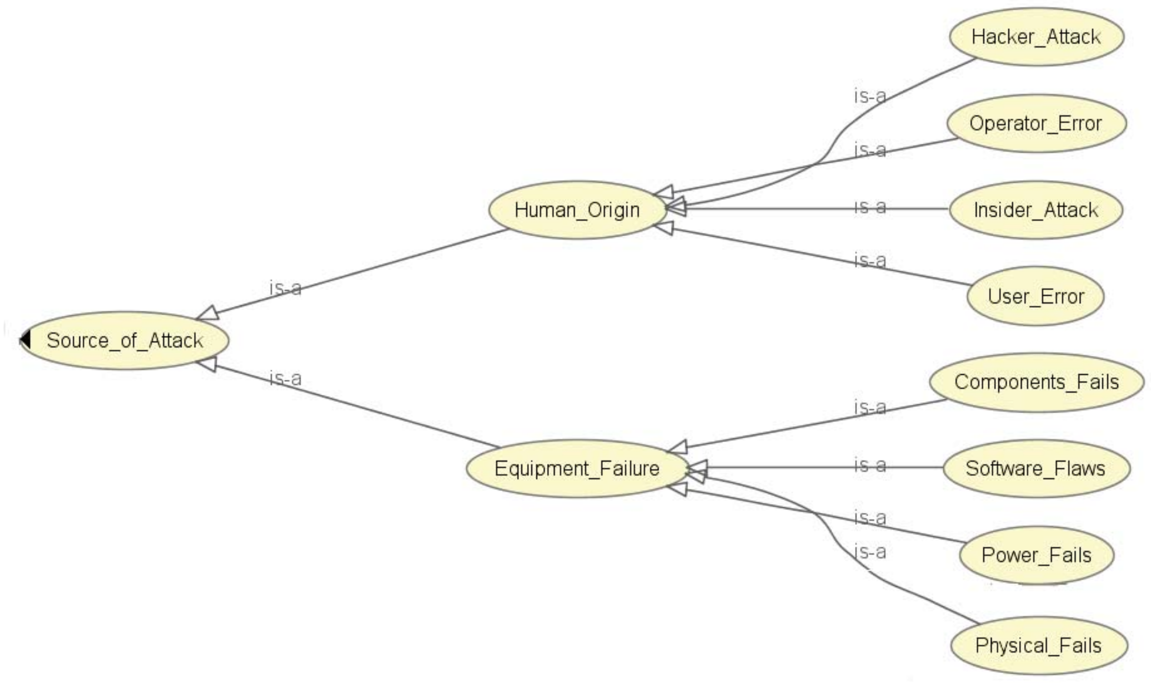


Figure 4.8: Sub-classes of Security Domain with Source-of-Attack Main Class

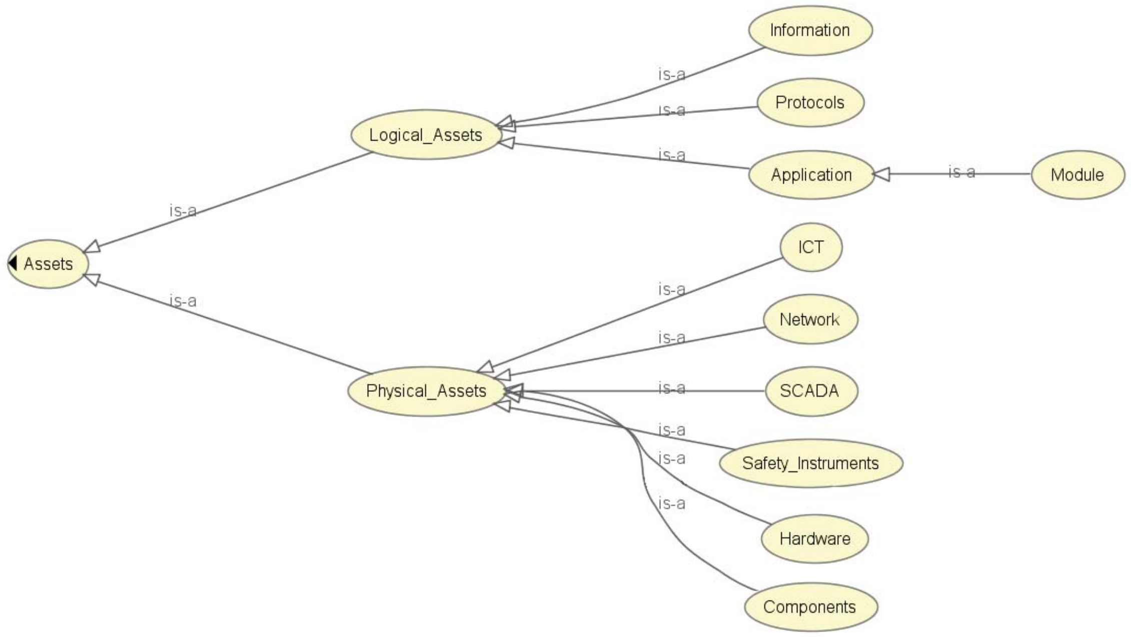


Figure 4.9: Sub-classes of Security Domain with Assets Main Class

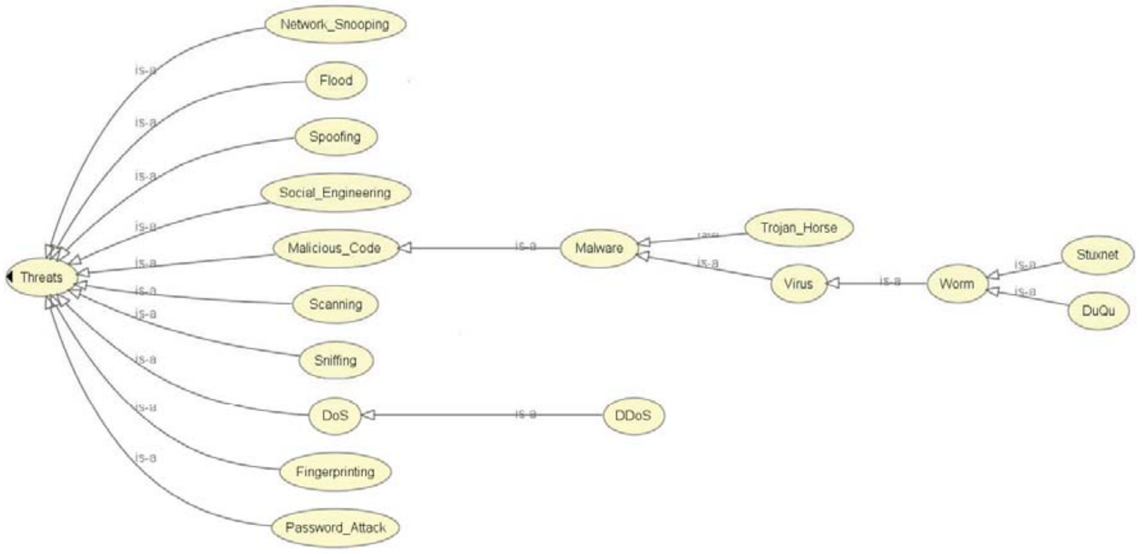


Figure 4.10: Sub-classes of Security Domain with Threats Main Class

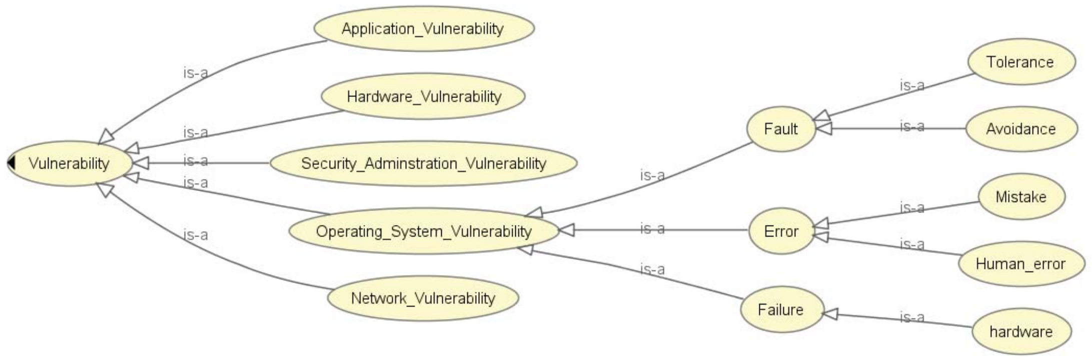


Figure 4.11: Sub-classes of Security Domain with Vulnerability Main Class

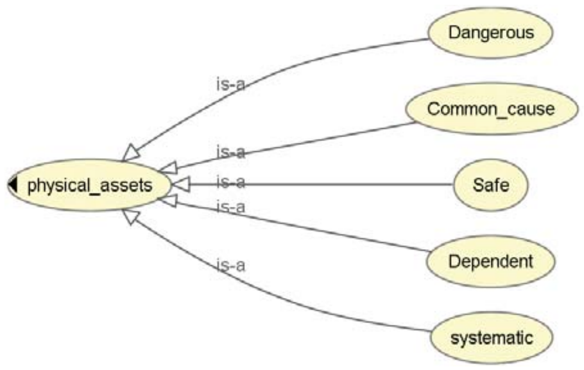


Figure 4.12: Sub-classes of Safety Domain with Physical-assets Main Class

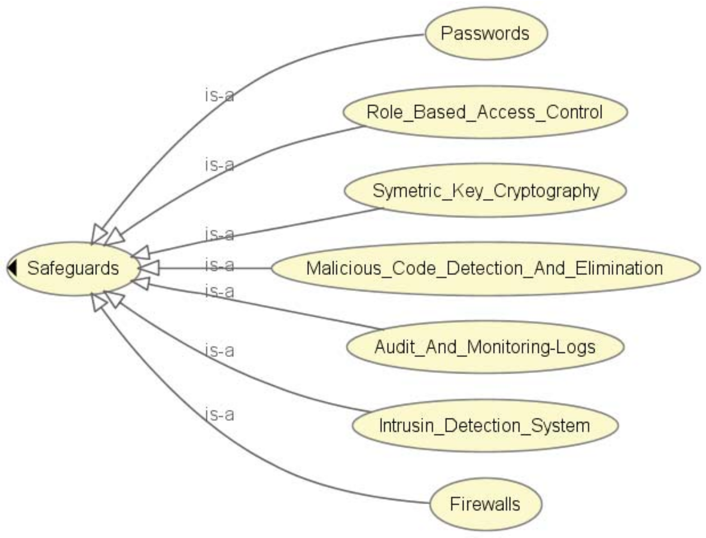

Safeguards
Passwords
Role_Based_Access_Control
Symetric_Key_Cryptography
Malicious_Code_Detection_And_Elimination
Audit_And_Monitoring-Logs
Intrusin_Detection_System
Firewalls
is-a

Figure 4.13: Sub-classes of Security Domain with Safeguards Main Class

is familiar with different aspects of security and safety and know the essential terms in applied field, i.e. oil and gas. The XML file related to these ontologies can be found in Appendix A.

4.4 Implementation and Design

This section will give a description of the implemented prototype. First, the APIs and additional software that has been used for the implementation is described. Next, a textual description of the prototype will be given.

4.4.1 Frameworks

This section will present an overview of the APIs and additional software used for implementing of this search engine prototype.

Python[2] is an interpreted, interactive, object-oriented, extensible programming language, which provides an extraordinary combination of clarity and versatility, and is free available and comprehensively ported. Python is a powerful dynamic programming language that is used in a wide variety of application domains. Python is often compared to Tcl, Perl, Ruby, Scheme or Java. Some of its key distinguishing features include[3]:

- Very clear, readable syntax
- Strong introspection capabilities
- Intuitive object orientation
- Natural expression of procedural code
- Full modularity, supporting hierarchical packages
- Exception-based error handling
- Very high level dynamic data types
- Extensive standard libraries and third party modules for virtually every task
- Extensions and modules easily written in C, C++ (or Java for Jython, or .NET languages for IronPython)
- Embeddable within applications as a scripting interface

[2]http://www.python.org/
[3]http://www.python.org/

Python is available for all major operating systems such as: Windows, Linux/Unix, OS/2, Mac, Amiga, among others. We use Python 2.7 in operating system UBUNTU 11.10, in order to back-end design and implementation of Semoogle application.

Django[4] is an open-source software and a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Django focuses on automating as much as possible and makes it easier to build better Web apps more quickly and with less code. We use Django Web programming framework in order to design the front-end Web page of this project and to gather the result in the result page.

GoogleAPI[5] is an easy way to import one or more APIs, and specify additional setting. The aim of using this API for this prototype is to have access to Google search engine and gather the URLs and title of the URLs from Google search engine for document collection. For this aim, PyGoogle is employed.

PyGoogle[6] is an easy-to-use wrapper for Google's Web API. We use this module to have access to Google's Web APIs through SOAP[7], and perform tasks such as search Google and get the results programmatically.

Protégé[8] is a free and open-source platform, which provides a suite of tools to construct domain models and knowledge-based applications with ontologies. At its core, Protégé implements a rich set of knowledge-modeling structures and actions, which support the creation, visualization, and manipulation of ontologies in various representation formats. It can be customized to provide domain-friendly support for creating knowledge-based models and entering data, and can be extended by way of a plug-in architecture and a Java-based API for building knowledge-based tools and applications [29]. We use Protégé platform to design the ontology proposed in this prototype. The results of this platform can be in diagram format as illustrated in Section 4.3, or can be a XML file to present the relationships between terms by OWL/XML, RDF/XML and Turtle language.

The following section describes the implementation strategy of the prototype.

4.4.2 Implementation and Design Strategy

This section describes the details of implemented prototype using combination of both textual, mock-up and pseudo code.

First of all, for implementing the prototype, we decided to provide mock-up to show

[4]https://www.djangoproject.com/
[5]https://developers.google.com/loader/
[6]http://www.google.com/apis/
[7]http://www.w3.org/TR/2000/NOTE-SOAP-20000508/
[8]http://protege.stanford.edu/

an overview of the implementation steps. Mock-up is a visual software showing a scale or is illustrated full-size model of a design, and is used for demonstration, design evaluation, promotion, and other purposes. We found it useful to clearly define all the tasks and details of the implementation steps. In figure 4.14, the design of a mock-up for the user query process in Semoogle interface. In this stage, the Semoogle is awaiting for user query for the next step that transfers the user query to the ontology for expanding the user's query. Figure 4.15 shows the next step of transferring the user query to the ontology and searching the user's query among the existing terms in the ontology list. For example, if user is looking for “Stuxnet”, the ontology list should contain “Stuxnet” term and other relevant terms related to the “Stuxnet” (such as worm, prevention, history, information, malware,...) to expand the query.

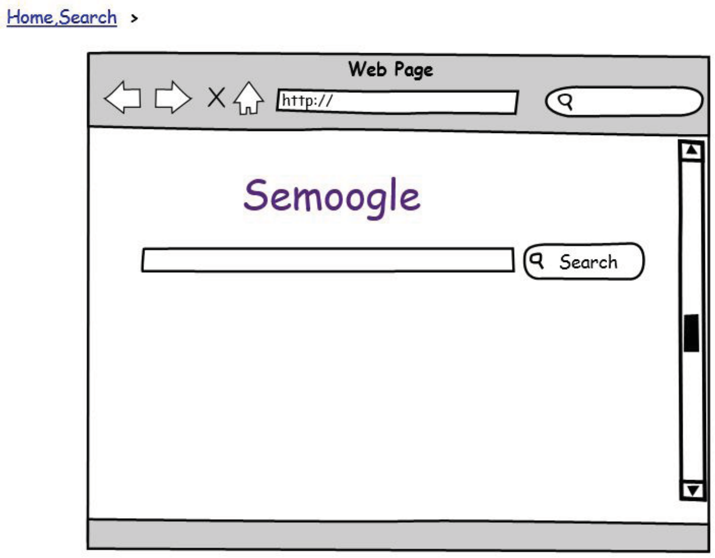


Figure 4.14: Mock-up of enter user query in Semoogle interface

The implementation section is then followed by describing the process after searching the query among the ontology list and expanding the query. The expanded query is ready to transfer to Google search engine to search and retrieve the relevant URLs. Figure 4.16 shows this step, which is a normal search using Google search engine. The results search of the Google will be retrieved containing *Title* and *URL* of each link. The titles of URLs are used to categorize the relevant results based on four categories. For categorizing the search results, a category-list has to be designed and specified in order to categorize the results based on these categories. In category-list, the terms relevant to each category are provided. For example, if the URL's title contains the name of the countries, this title is in accompanied with its description and its URL should be accommodated to the *History* category. Figure 4.17 shows the process of matching the relevant keywords in the title with relevant category. According to this figure, several terms relevant to the

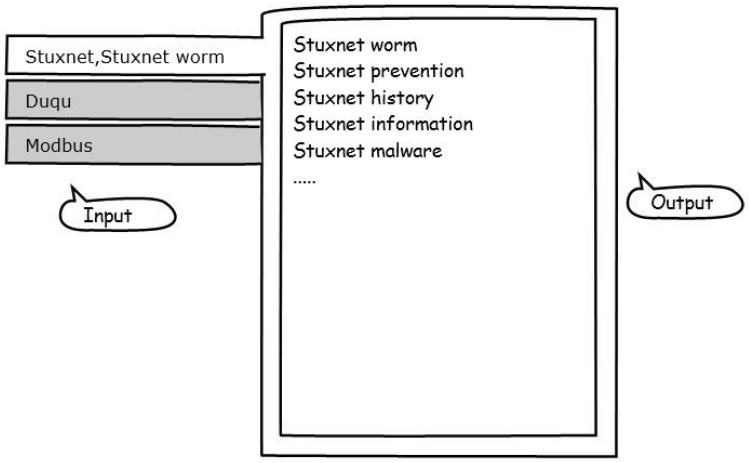


Figure 4.15: Mock-up of matching user query with existing terms in the ontology list

"Stuxnet" in *History* category could be found such as: history, fascinating history, nuclear news, new, etc. Assigning these terms to each category is depended on the domain of the ontology. We are using the safety and security terms, so the terms in each category is relevant to this domain of concept. Figure 4.18 shows the mock-up of final results, which is presented to user after categorizing the Google results search based on user needs into four categories (History, Mechanism, Prevention, Case study).

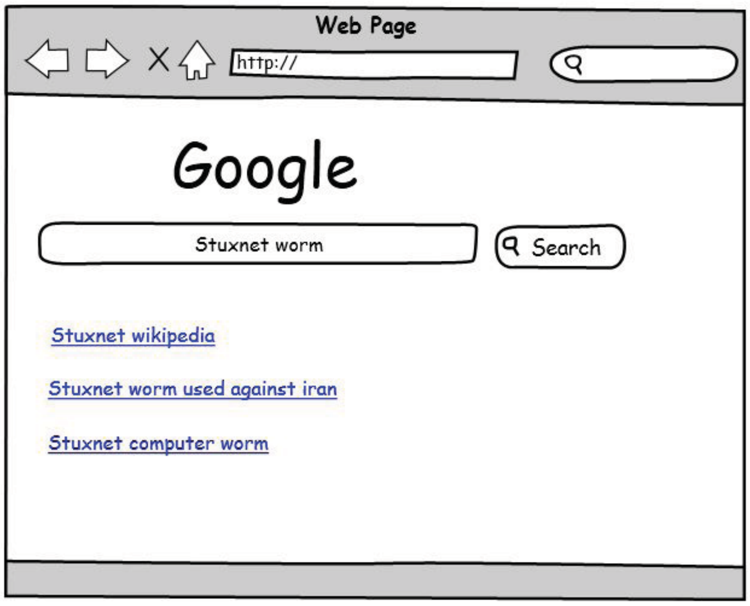


Figure 4.16: Mock-up of searching expanded query using Google search engine

Pseudo-code for the whole strategy is given in figure 4.19. The procedure starts with entering the query *qi*, which is available in the ontology list *Q*. The next line of pseudo

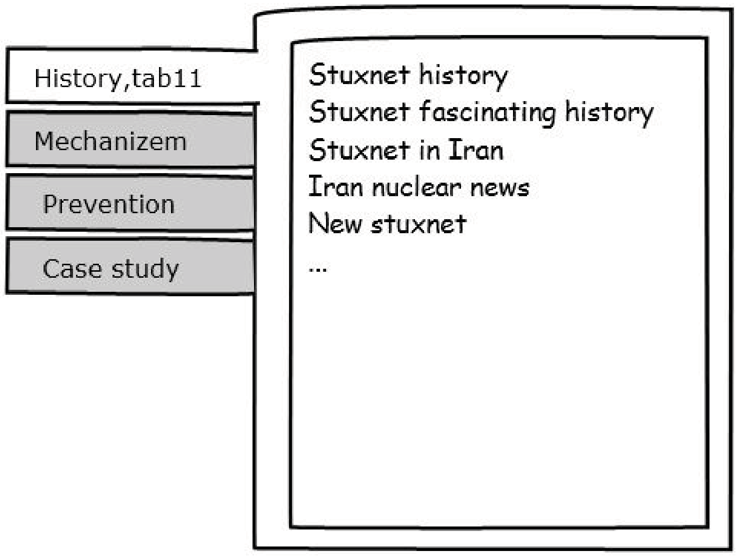


Figure 4.17: Mock-up of categorizing the Google search results based on four caregory

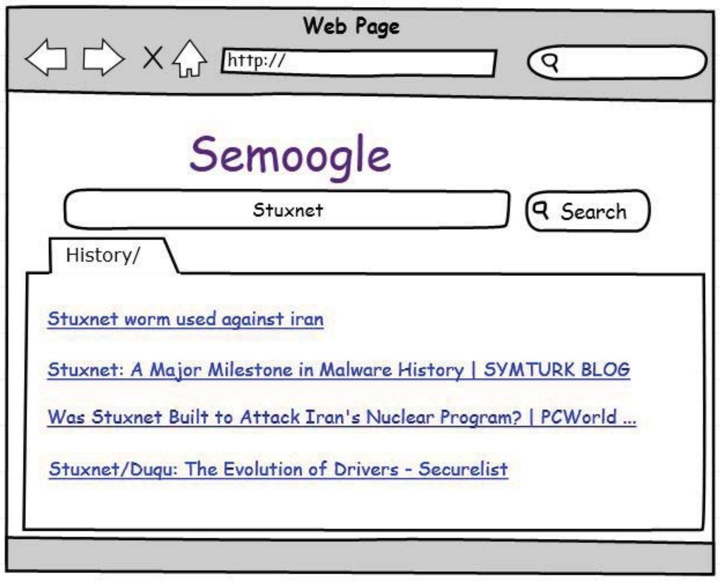

Web Page
http://
Semoogle
Stuxnet
Search
History/
Stuxnet worm used against iran
Stuxnet: A Major Milestone in Malware History | SYMTURK BLOG
Was Stuxnet Built to Attack Iran's Nuclear Program? | PCWorld ...
Stuxnet/Duqu: The Evolution of Drivers - Securelist

Figure 4.18: Mock-up of final results for user

code shows the retrieved process of relevant terms related to the query qi from ontology list Q. After this step, the expanded terms should be transferred to the Google search engine(line 3 in figure 4.19). By transferring the expanded terms to the Google search engine, the traditional search of terms will be started and provided the relevant search based on the expanded terms. In this stage of the process, because of the number of relevant terms related to the query, more relevant links in the result will be obtained than the exact query search using Google search engine. This part of the strategy will provide the semantic search by providing more relevant terms instead of searching the exact query directly in Google search engine. In next line of the pseudo-code, the results of the Google search will be provided and is ready to categorize based on user needs. In the next step, repetitive results or those URLs, which have the same number of match terms for two of categories are considered. The solution for this step is weighting terms, which attempt to assign weight to the latter categories. By this solution, category of *History* will have the lower priority than the category of *Mechanism*. Also, the category of *Mechanism* has the lower priority than the *Prevention* category. *Case study* category will have the higher priority in this process. Prioritizing of the categories is completely dependent on the user needs, which means that it is flexible to rearrange the priority of categorization based on other requirements of the user for the time being.

```
For (each $q_i$ in $Q$) {
    Retrieve all the existing relevant terms in ontology list related to the $q_i$;
    Expand the $q_i$ based on the relevant terms related to $q_i$ in ontology list;
    Transfer the expand $q_i$ to Google search engine;
    Return the results of Google search based on expand $q_i$;
    For (each Title $T_i$ in each Link $L_i$) {
        Find the relevant keyword $K_i$, which is matched to the category $C_i$;
        Assign the Title $T_i$ to category $C_i$;
        Remove the repetitive links;
        If Title $T_i$ has equal keywords to be assigned to category $C_i$ and $C_j$ {
            Assign the Title $T_i$ to category $C_j$;
            }
        }
    Show the categorized results for user;
    }
```

Figure 4.19: Pseudo code for the whole process of implementation strategy

Chapter 5

Evaluation

Once the design and implementation of the prototype is done, it is time to evaluate the Semoogle application. The evaluation section plays a crucial role to make a progress building better search engine. The primary aspects in evaluation of search engine is *effectiveness* and *efficiency*. The scope of this evaluation is concerned with the aspects of the implemented prototype. These aspects concern about the extent of relevancy between the retrieved results and user query in the search domain. This evaluation is significantly dependent on the ontology defining the relevant terms for expanding the user query. In this sence, the performance of the semantic search is dependent on the pre-defined ontology.

The focus of this chapter is on the evaluation of data, metrics and strategy of evaluation. Also, the observation of testers is documented. Based on the results obtained, the performance of the application is evaluated.

In order to evaluate the impact of using ontology to define the semantic search engine, we assumed the important metric as a target of evaluation. This metric includes *the impact of ontology on the user content (Coverage)*. For the impact of ontology on the user content, we are first interested in the quality of URLs accessed by the users in Semoogle results search. To this end, we compare the URLs in the Google with Semoogle results search, and measure the fraction of the URLs that are categorized by the Semoogle and diversity of these URLs compared with Google. The goal of this metric approach is to determine whether the URLs discovered via Semoogle are indeed useful and coverage the different aspects of the user's needs.

5.1 Evaluation of Data

In order to implement the evaluation, the result search of pre-defined queries in the safety and security domain is compared to the reference results. In this study, the retrieved results by Google search engine are selected as the reference case.

In order to carry out fair comparison, the Semoogle title search result is compare with the same number of retrieved URLs' title in Google. In Semoogle, each query is extended by adding the related terms and search is carried out for a bunch of related query based on pre-defined ontology. On the other hand, in Google search only one query is searched and it seems that for the exact word the number of search result from Google is more than Semoogle, while the returned URLs from Semoogle is semantically more relevant to the domain of study.

The domain of the evaluation data is safety and security terms. The ontology we are using in the search engine prototype evaluation is the INSPIRE ontology, which is modified based on the requirements in this domain of study. Notwithstanding, many of the concepts in INSPIRE ontology replaced during the construction of the new ontology according to INSPIRE ontology, the main classes of the new ontology is the same as INSPIRE ontology. Due to the lack of time, the new ontology is implemented in preliminary steps and developed for testing the Semoogle application purpose.

In evaluation phase, four sample queries is selected and the results are compared with Google. The sample queries are as following:

1. Virus
2. Worm virus
3. SCADA
4. Stuxnet virus

The first two sample queries include more general terms, which can be included in several scientific domains from computer sciences to medical sciences and agriculture. The last two samples are more specific for the scope of this thesis however still "SCADA" can be employed for industrial control system, and monitor or control of critical infrastructures.

In the evaluation, the search engine effectiveness metrics are quantified. This quantification is carried out in such a way that to what extent the search engine returns the results that can cover the important concepts relevant to the user query in the domain. All the steps of evaluation are described in the following paragraphs.

In figure 5.1, the result search of Semoogle is compared with Google for "Virus" as the user's query. As shown, Google returned more URLs for the exact query however, for the other relevant terms Semoogle exhibits more accurate response. Figure 5.1 compares the returned results of 92 URLs both in Google and Semoogle.

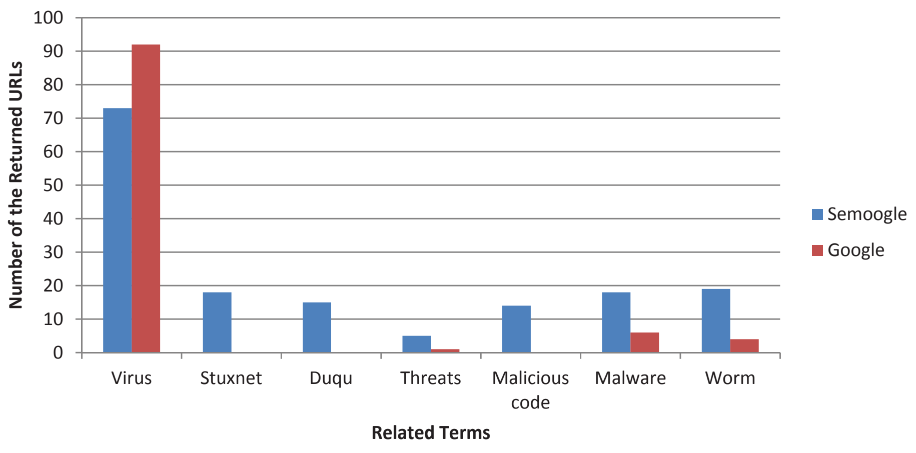


Figure 5.1: Compare the relevancy of returned URLs for “Virus” query in Semoogle and Google in the domain of study

As above mentioned, “Virus” is a general query over different domains and includes variety of concepts in each. For instance in computer science, it includes malware and attacks that threats the safety of computer systems, whereas in other domains such as medical science this is completely different concept. In this sense, Google returns more URLs with respect to the wider spectrum of domains, while many of them are out of the scope of safety and security. Therefore, the results of Semoogle is relatively lower than Google for the exact word of “Virus” (Red bar). On the other hand, Semoogle exhibits better performance in the other relevant terms as shown in blue bars. All the terms in the figure have semantic connection under pre-defined ontology in Semoogle search engine leading more insightful results by Semoogle.

In figure 5.2, the term “Worm Virus” is selected as a sample query to examine and evaluate the Semoogle search engine’s performance compared to Google out of 124 URLs. As showed in red bar for the exact term, more URLs have been retrieved by Google. It means for finding the exact term “worm virus” in the title of returned URLs, Google indisputably returns more results than the Semoogle. However, since an ad hoc search has been performed by Google in different domains, they are not necessarily related to our domain of study. The scope of search in Semoogle is narrowed down in order to capture the effect of semantic search in the selected domain. Hence, Google returned more results for the exact terms, “Worm” and “Virus” shown in red bar. However, for the other relevant terms related to the user query in safety and security domain, Semoogle returned more results, e.g. “Stuxnet”, “Duqu”, “Security”, “Threats” and “Malware”.

These two samples of the common terms have variety of concept in different domain of studies. The results demonstrate that the Google search engine returns more URLs for the exact user query, however the Semoogle returns more relevant results in comparison with the Google due to the existing relationship between terms and user query in

Semoogle search engine.

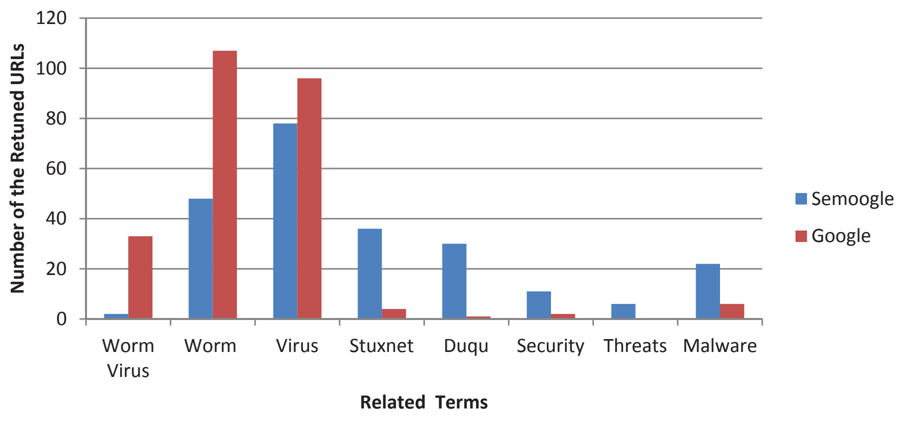


Figure 5.2: Compare the relevancy of returned URLs for "Worm Virus" query in Semoogle and Google in the domain of study

We carry out the evaluation on two other sample terms in order to consider the performance of Semoogle for the specific terms in safety and security domain compared to Google search engine. In this association, "SCADA" is selected as a sample user query. It is applied not only in safety and security domain, but also to the other fields, e.g. industrial control and monitor or control of critical infrastructures. The next query is "Stuxnet Virus", which is a term directly related to security and safety field. IN the following paragraph the results of the comparison carried out between Semoogle and Google is presented.

Figure 5.3 shows the number of retrieved results in Semoogle and Google search engine for "SCADA" term. The terms such as "Asset", "Security", "Threat", "Risk" and "Vulnerability" are selected as the related terms to "SCADA" in safety and security domain based on the pre-defined ontology in Semoogle. The red bar shows that the Google returns more URLs than Semoogle result search for the exact term of "SCADA". This result is the same as two queries in above. For the other related terms to "SCADA" in safety and security domain, it turns out that Google exhibits significantly poor performance. The retrieved URLs out of 32 is almost zero in Google search result for those terms related to security and safety domain. These results demonstrate that Google returns more URLs for the exact terms and Semoogle returns more accurate results related to the "SCADA" term in our domain of study (the blue bars in the figure 5.3).

The last chosen sample is "Stuxnet Virus", which is specific term in the safety and security domain. The aim of evaluation is not only focuses on the general terms, e.g. "Virus", and find the related URls to domain of study, but also to carry out this specification for the terms, which are belong to the domain of study, e.g. "Stuxnet Virus".

The results of "Stuxnet Virus" are depicted in figure 5.4), which are out of 148 retrieved

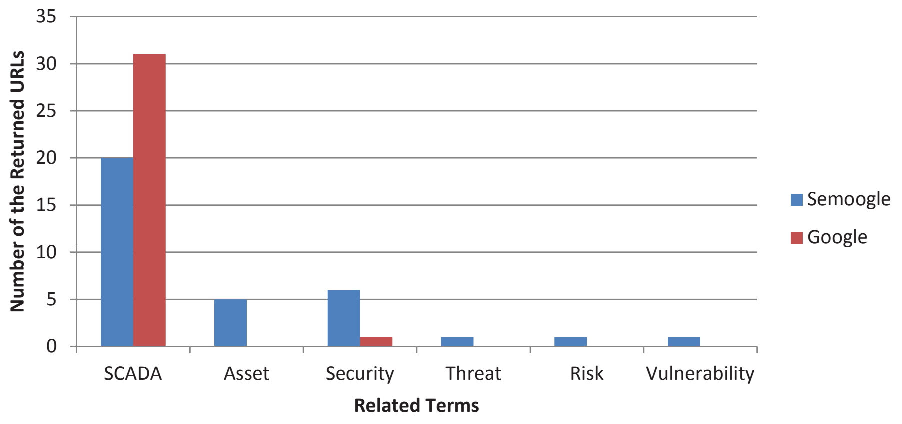


Figure 5.3: Compare the relevancy of returned URLs for “SCADA” query in Semoogle and Google in the domain of study

URLs. The figure shows that for the exact term “Stuxnet Virus”, Google search engine presents more significant results than Semoogle. Like the other considered sample above, the Semoogle returned more accurate results in further terms which shown the extent relevancy between “Stuxnet Virus” and other terms related to the “Stuxnet Virus” such as “Security”, “Threat”, “Malicious code”, “Malware” and “Worm”.

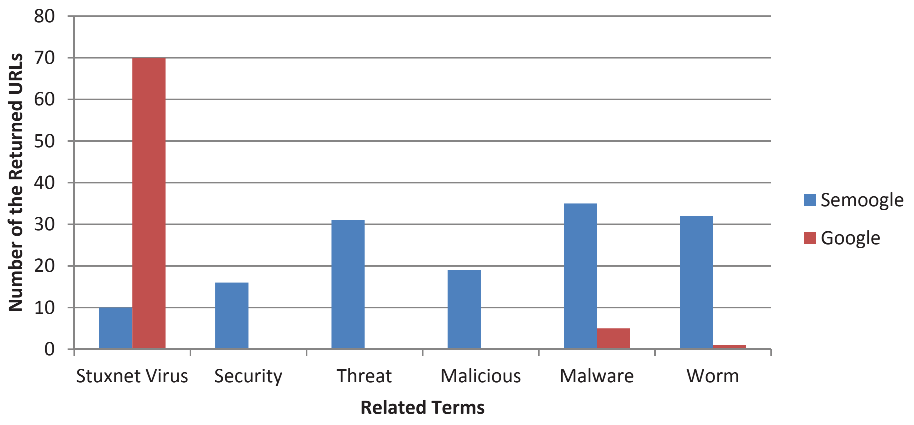


Figure 5.4: Compare the relevancy of returned URLs for “Stuxnet Virus” query in Semoogle and Google in the domain of study

The evaluation has shown that on average the search engine prototype increased the performance of search compared with purely keyword based search. This comparison is carried out both with respect to the general terms and more specific term in the domain of study. Implementation of semantic search based on the selected ontology perform superior to the keyword based search. Many of the less relevant URLs containing the instances of the search query are filtered out, and the output is much more oriented to the selected domain of study.

In stepwise approach, the relevancy of sample terms to the domain of study has been increased in four selected queries. The scope starts from very general terms and is narrowed down to more specific terms in safety and security domain. Moving from general to more specific terms, makes the result search to be more concentrated to the exact user query, while this effect is different in Semoogle. The result search are more spread over the relevant terms in case of specific queries.

The evaluation has demonstrated that our approach in semantic search is somewhat better than the traditional search in Google search engine, and covers and satisfies the user requirements. This prototype is applicable to categorize the results according to the category-list. In addition to the advantages of the semantic search, the categorization of the results will be very interesting and projects the results in better way for the users.

Due to the terms consist of several existing terms in ontology list in this approach, all the terms is mashed in the ontology and take their relevant terms and expanded based on their relationships. The complexity of the search in specific domain is significantly reduced and it is much easier to find the semantically relevant terms in the domain of study.

Chapter 6

Discussion

This chapter will give a description of our findings from evaluation and will discuss about the limitation and the possibilities for improvement of the prototype. This chapter will consider these discussion in three parts as advantages of Semoogle, its limitation and future work in order to develop this application.

6.1 Advantages of Semoogle

Compared to conventional search systems, Semoogle search engine which is a meaning-based or ontology-based search engine has a number of proven advantages in different aspects. We will further discuss these issues.

Semoogle search engine is precisely focused on its ontology and in fact it is ontology driven. Deploying ontology, which describes knowledge about the domain in terms of concepts or vocabularies within the domain and relationships between them. In Semoogle search engine has two advantages for this application. One of these advantages is due to semantic search and defines the relationship among terms. The other advantage is related to the user query which leads the user to use fewer queries in search process than Google. This benefit is based on additional existing terms related to the queries in the ontology.

The ontology deployed in this application is dynamic for developing the proposed ontology in safety and security domain. It is possible to extend the development of the ontology to other levels of safety and security in order to cover other relevant terms to this field of study. Development of this ontology will help the user to have more broader search results than current ontology. Also it is dynamic in order to substitute the ontology to other domains of the studies such as oil and gas pipeline, electrical domain and etc.

Since the results of Semoogle search are categorized in four categories, this application is more users friendly to project results to the user. Based on this characteristic of the Semoogle results search, the user can find the results in specified category leading to speed up the searching process. Therefore, this benefit is efficient in order to accelerate the accessibility of the required information for the user and time saving.

For searchers, well-known search engines generally mean more dependable results. Since the Semoogle application is using the Google search engine in order to searching the user queries as its backbone, we claimed Semoogle take full advantages of Google search engine which is well-known search engine through out the world at the present time. On the other hand, since the Semoogle can retrieve the results in more relevant aspects and categorize them in four categories, we can claim the solidarity of retrieved results in Semoogle search engine is more than Google search engine.

Compare to hakia search engine which is a popular semantic search engine, Semoogle can be developed for a variety of purposes such as oil and gas pipeline, electrical power grids, water distribution and wastewater collection systems, government operations, banking and finance, railway transportation systems and all fields of study by substituting the ontology to the relevant domain. But hakia claims that their powerful solutions is in aerospace, medical, and financial market areas.

6.2 Limitation of Semoogle

This search engine prototype has some limitations. In this section, we will consider the limitations of Semoogle. Since this chapter will show the road map for future work, require closer scrutiny because of its potential to improve and develop the prototype.

First, legitimately through Google APIs will determinate the user intent for the search up to one hundred queries per day for free. It is possible to have an account and buy more by paying for that. Although using Google APIs with one hundred queries per day will determinate the user, since this project is based on delighting a prototype, it will cover the query expected. In order to develop the prototype, it can be a solution to buy more queries from Google APIs. However, this application will be terminated after sending one hundred queries search request to Google API[1].

Second, since the user query is based on ontology we have used to improve the semantic search, the user has limitation to use the queries, which are not covered by ontology. Therefore, the results of search may be affected and don't cover the user satisfaction. Since this ontology needs to be improved and developed, this limitation can be solved in future by use more advanced ontology.

Third, the most existing search engine development tools do not support the development of search engines in languages other than English. The Semoogle application is capable to return only English results and can not support languages other than English.

[1] `https://developers.google.com/custom-search/v1/using_rest\#intro`

Forth, there are a couple of other limitations in this prototype due to lack of time to enhance and expand the ontology and also lack of information about the concepts and different aspects in security and safety domain. Hence, if the ontology list does not cover all the aspects of the fields of study, the results of the search faces limitation. This prototype needs to be improved to provide the complete list of terms and also to present a precise ontology. The terms play a significant role in the ontology and in search results. More connectivity and relationship between the terms in the ontology will present the precise results of the search. Moreover, the ontologist also play an important role to assign the relevant terms and find the relationships between the terms.

Note that system performance in terms of speed is not within the scope of this project, and will not be subject to evaluation. Evaluating the implementation using standard precision and recall measures is not viable for this project. This is mainly due to the fact that we do not have an overview of which documents in the document collection searched should be deemed as relevant for each of the queries. The approach chosen for the evaluation of the ranking of the hits is to specify several queries, and have the users evaluate the top 10 hits for each query and search strategy. Note that the ranking of search results is based on Google search engine ranking and modifying the Google search's ranking is beyond the scope of this prototype.

We found some small bugs in the implementation during the evaluation. We found out that the prototype was not capable to handle the unnecessary space before or after the query and got completely stuck. We attempted to address this issue by refining entered queries. Also, when user tried to using other keywords, which did not exist in the ontology, the application was not able to handle and transfer the query to the Google search engine. By our endeavor, the application is capable now to handle and transfer the keywords, which are not existing in the ontology, to Google search engine and return relevant results.

6.3 Future Work

Since the Semoogle search engine is prototype and needs improvement in many aspects, the suggestions for future work will help the developers for further modification and development of the Semoogle search engine in the future. In this section, we will consider the possible improvements for the future works.

The approach of this project is to design a prototype of a semantic search engine, and the main aspects of a search engine is addressed in this project. In this basis, the paradigm ontology needs to be improved and developed in the future work. As mentioned in the context of this report, this ontology is only a small instance, which needs the revision of ontologist in this field to improve and develop its aspects. Also, the proposed ontology can be substitute to different domains such as electrical, chemical, power plant domain and etc. This facility will help the developer to have this possibility in develop the search engine in many area of the studies.

Semoogle application puts high weight on “Case study” category in its ranking algorithm and therefore can be criticized for strengthening the other categories. While the weighting algorithm is prioritized for “Case study” alternatively, there is possibility to substitute it to other categories, which are more important from user side.

While most search engine applications allow users to select one factor to rank results, this application also can be improved in the area of ranking the search results according to the user needs, e.g. publication date. Consequently, the first search results can have high relevancy than the other search results.

Chapter 7

Conclusions

Since general search engines such as Google are basically text-based, i.e. match only the keywords entered by the users and return the list of results, the search engine prototype described in this prototype, investigates the development of the outperform Google, in terms that it not only matches keyword entered by the user, but also matches the other keywords having relationship with the given keywords in proposed ontologies.

According to the above premise, this study presents a prototype of a semantic search engine, named as Semoogle. It is based on ontology used to optimize the search process in order to improve the quality of the results search. The proposed ontology plays the main role in this semantic search engine. A domain of relevant document collection has been used in the process of constructing the semantic search engine, which is related to the area of safety and security. This ontology is assumed to be as paradigm ontology and can easily be adapted to the other domains of study in order to satisfy the user requirements in other fields of study.

This prototype categorizes the results search in four designated categories based on user requirements. These categories are alternatively gathered information for users within the domain scope of this study and can be adapted to the other groups in the other domains. The Semoogle search engine uses Google API to search terms and retrieves the results from Google search engine.

The heuristic method presented in this report for the ranking of results search is defined in two steps in order to avoid duplication of the retrieved results in different categories. These two steps are named as horizontal weight and vertical weight. The former is performed to assign the results search to the relevant category, and the later is performed to prioritize the categories.

In evaluation chapter, two steps are chosen in order to test the application. The first step is applied to test general query. The results of Semoogle search engine are compared with Google. The second test is performed to study the more specific term in safety and security domain in both Semoogle and Google search engine.

Comparing the results of Semoogle with Google are summarized to the following concluding remarks:

- The Semoogle search engine provides more relevant results according to the user query.
- The proposed ontology plays important role to optimize the returned results in Semoogle search engine.
- The returned results of Semoogle search engine are robust and more relevant to the field of study than Google.
- Moving from general terms to specific terms makes the results search to be more concentrated to the exact query in Google search where as Semoogle retrieves more broader and relevant results search in the domain of study.
- More consistency between the terms in the ontology will provide more relevant results as the outcome of Semoogle search engine.

The categorization of the results search will help the users to find their desired information quickly and faster than Google results search.

Bibliography

[1] "The search engine architecture, ppt, csci 572: Information retrieval and search engines summer 2010," *USV Viterbi, School of Engineering, University of Southern California*, 2010.

[2] Y. Mao and A. Yin, "Ontology modeling and development for traditional chinese medicine," in *Biomedical Engineering and Informatics, 2009. BMEI '09. 2nd International Conference on*, oct. 2009, pp. 1 –5.

[3] M. Choras, A. Flizikowski, R. Kozik, and W. Holubowicz, "Decision aid tool and ontology-based reasoning for critical infrastructure vulnerabilities and threats analysis," pp. 98–110, 2010.

[4] S. Brin and L. Page, "The anatomy of a large-scale hypertextual web search engine," *Comput. Netw. ISDN Syst.*, vol. 30, pp. 107–117, Apr. 1998.

[5] V. Sugumaran and J. Gulla, *Applied Semantic Web Technologies.* Taylor and Francis, 2011.

[6] A. Hogan, A. Harth, J. Umrich, S. Kinsella, A. Polleres, and S. Decker, "Searching and browsing linked data with swse: the semantic web search engine," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 9, no. 4, 2012. [Online]. Available: http://www.websemanticsjournal.org/index.php/ps/article/view/240

[7] H. Zhang, B. Pang, K. Xie, and H. Wu, "Computational intelligence and security," Y. Wang, Y.-M. Cheung, and H. Liu, Eds. Berlin, Heidelberg: Springer-Verlag, 2007, ch. An Efficient Algorithm for Clustering Search Engine Results, pp. 661–671. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-74377-4_69

[8] Hakia. [Online]. Available: http://www.hakia.com(Accessedon09February2009)

[9] N. Aghajani, "A systematic literature review on software security and scada security taxonomy and ontology," 2011.

[10] T. Berners-Lee, J. Hendler, and O. Lassila, "The Semantic Web (Berners-Lee et. al 2001)," *Scientific American*, May 2001.

[11] G. Denker, L. Kagal, T. Finin, and M. Paolucci, “Security for daml web services: Annotation and matchmaking,” in *In Proceedings of the 2nd International Semantic Web Conference, Sanibel Island.* Springer, 2003, pp. 335–350.

[12] C. D. Manning, P. Raghavan, and H. Schtze, *Introduction to Information Retrieval.* New York, NY, USA: Cambridge University Press, 2008.

[13] K. N. Andersen, E. Francesconi, Å. Grönlund, and T. M. van Engers, Eds., *Electronic Government and the Information Systems Perspective - Second International Conference, EGOVIS 2011, Toulouse, France, August 29 - September 2, 2011. Proceedings*, ser. Lecture Notes in Computer Science, vol. 6866. Springer, 2011.

[14] P. Hitzler, M. Krötzsch, and S. Rudolph, *Foundations of Semantic Web Technologies.* Chapman & Hall/CRC, 2009.

[15] D. L.McGuinness, “Ontologies come of age,” in *NIST Interoperability Week.* Stanford University, 2003.

[16] R. V. Guha, R. McCool, and E. Miller, “Semantic search,” in *The Twelfth International World Wide Web Conference*, 2003, pp. 700–709.

[17] N. F. Noy and D. L. mcguinness, “Ontology Development 101: A Guide to Creating Your First Ontology,” 2001. [Online]. Available: http://www.ksl.stanford.edu/people/dlm/papers/ontology101/ontology101-noy-mcguinness.html

[18] B. Smith and C. Welty, “Fois introduction: Ontology—towards a new synthesis,” in *FOIS ’01: Proceedings of the international conference on Formal Ontology in Information Systems.* ACM Press, 2001. [Online]. Available: http://dx.doi.org/10.1145/505168.505201

[19] J. Kassim and M. Rahmany, “Introduction to semantic search engine,” in *Electrical Engineering and Informatics, 2009. ICEEI ’09. International Conference on*, vol. 02, aug. 2009, pp. 380 –386.

[20] Elizabeth Liddy,Professor, School of Information Studies. Syracuse University , “How a search engine works,” *Searcher: The Magazine for Database Professionals*, vol. 9, pp. 38–45, May 2001.

[21] R. A. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval.* Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1999.

[22] “Our history in depth.” [Online]. Available: http://www.google.com/about/company/history/

[23] C. Calero, C. Muñoz, M. Moraga, and M. Piattini, *Handbook of Research on Web Information Systems Quality*, 2008.

[24] Y. Wang and M. Kitsuregawa, “Use link-based clustering to improve web search results,” in *Web Information Systems Engineering, 2001. Proceedings of the Second International Conference on*, vol. 1, dec. 2001, pp. 115–124 vol.1.

[25] H.-J. Zeng, Q.-C. He, Z. Chen, W.-Y. Ma, and J. Ma. SIGIR 2004, 2004, ch. Learning to Cluster Web Search Results, pp. 210–217.

[26] P.-H. Wang, J.-Y. Wang, and H.-M. Lee, “Query find: Search ranking based on users’ feedback and expert’s agreement,” *IEEE*, vol. 4, pp. 48–54, New York 2004.

[27] A. Navot, L. Shpigelman, N. Tishby, and E. Vaadia, ““nearest neighbor based feature selection for regression and its application to neural activity.”,” in *Advances in Neural Information Processing Systems 18*. MIT Press, 2005, pp. –1–1.

[28] W. B. Frakes and R. A. Baeza-Yates, Eds., *Information Retrieval: Data Structures & Algorithms*. Prentice-Hall, 1992.

[29] M. Horridge, H. Knublauch, A. Recto, R. Stevens, and C. Wroe, “A practical guide to building owl ontologies using the protege-owl plugin and co-ode tools,” vol. 27, pp. 0–117, 2004.

[30] M. Szlezak, “Markup language in realization of design tasks,” 2005. [Online]. Available: http://www.ecolleg.org/trms/ontology.html

Appendix A

OWL Ontology in XML

```
<?xml version="1.0"?>


<!DOCTYPE rdf:RDF [
    <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
    <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
    <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
    <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
    <!ENTITY Ontology1335173950609 "
    http://www.semanticweb.org/ontologies/2012/3/Ontology1335173950609.owl#" >
]>


<rdf:RDF xmlns="http://www.semanticweb.org/ontologies/2012/3/Ontology1335173950609.owl#"
     xml:base="http://www.semanticweb.org/ontologies/2012/3/Ontology1335173950609.owl"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
     xmlns:Ontology1335173950609=
     "http://www.semanticweb.org/ontologies/2012/3/Ontology1335173950609.owl#">
    <owl:Ontology rdf:about=
    "http://www.semanticweb.org/ontologies/2012/3/Ontology1335173950609.owl">
        <rdfs:comment>A security ontology that discribe the various terms based on
        relationship between the terms.</rdfs:comment>
    </owl:Ontology>



    <!--
    ///////////////////////////////////////////////////////////////////////////////////////
    //
    // Annotation properties
    //
    ///////////////////////////////////////////////////////////////////////////////////////
     -->




    <!--
    ///////////////////////////////////////////////////////////////////////////////////////
    //
    // Object Properties
    //
    ///////////////////////////////////////////////////////////////////////////////////////
     -->




    <!-- http://www.semanticweb.org/ontologies/2012/3/Ontology1335173950609.owl#AffectedBy -->

    <owl:ObjectProperty rdf:about="&Ontology1335173950609;AffectedBy">
        <rdfs:range>
            <owl:Restriction>
                <owl:onProperty rdf:resource="&Ontology1335173950609;AffectedBy"/>
            <owl:someValuesFrom rdf:resource="&Ontology1335173950609;Threats"/>
        </owl:Restriction>
    </rdfs:range>
</owl:ObjectProperty>



<!-- http://www.semanticweb.org/ontologies/2012/3/Ontology1335173950609.owl#isRelateing
-->

<owl:ObjectProperty rdf:about="&Ontology1335173950609;isRelateing"/>



<!--
///////////////////////////////////////////////////////////////////////////////////////
//
// Classes
//
///////////////////////////////////////////////////////////////////////////////////////
 -->




<!-- http://www.semanticweb.org/ontologies/2012/3/Ontology1335173950609.owl#Application
-->

<owl:Class rdf:about="&Ontology1335173950609;Application">
    <rdfs:subClassOf rdf:resource="&Ontology1335173950609;Logical_Assets"/>
</owl:Class>



<!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1335173950609.owl#Application_Vulnera
bility -->

<owl:Class rdf:about="&Ontology1335173950609;Application_Vulnerability">
    <rdfs:subClassOf rdf:resource="&Ontology1335173950609;Vulnerability"/>
</owl:Class>



<!-- http://www.semanticweb.org/ontologies/2012/3/Ontology1335173950609.owl#Assets -->

<owl:Class rdf:about="&Ontology1335173950609;Assets">
    <owl:disjointWith rdf:resource="&Ontology1335173950609;Safeguards"/>
    <owl:disjointWith rdf:resource="&Ontology1335173950609;Source_of_Attack"/>
    <owl:disjointWith rdf:resource="&Ontology1335173950609;Threats"/>
    <owl:disjointWith rdf:resource="&Ontology1335173950609;Vulnerability"/>
</owl:Class>



<!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1335173950609.owl#Audit_And_Monitorin
```

```
g-Logs -->

<owl:Class rdf:about="&Ontology1335173950609;Audit_And_Monitoring-Logs">
    <rdfs:subClassOf rdf:resource="&Ontology1335173950609;Safeguards"/>
</owl:Class>



<!-- http://www.semanticweb.org/ontologies/2012/3/Ontology1335173950609.owl#Avoidance -->

<owl:Class rdf:about="&Ontology1335173950609;Avoidance">
    <rdfs:subClassOf rdf:resource="&Ontology1335173950609;Fault"/>
</owl:Class>



<!-- http://www.semanticweb.org/ontologies/2012/3/Ontology1335173950609.owl#Common_cause
-->

<owl:Class rdf:about="&Ontology1335173950609;Common_cause">
    <rdfs:subClassOf rdf:resource="&Ontology1335173950609;physical_assets"/>
</owl:Class>



<!-- http://www.semanticweb.org/ontologies/2012/3/Ontology1335173950609.owl#Components -->

<owl:Class rdf:about="&Ontology1335173950609;Components">
    <rdfs:subClassOf rdf:resource="&Ontology1335173950609;Physical_Assets"/>
</owl:Class>



<!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1335173950609.owl#Components_Fails
-->

<owl:Class rdf:about="&Ontology1335173950609;Components_Fails">
    <rdfs:subClassOf rdf:resource="&Ontology1335173950609;Equipment_Failure"/>
</owl:Class>



<!-- http://www.semanticweb.org/ontologies/2012/3/Ontology1335173950609.owl#DDoS -->

<owl:Class rdf:about="&Ontology1335173950609;DDoS">
    <rdfs:subClassOf rdf:resource="&Ontology1335173950609;DoS"/>
</owl:Class>



<!-- http://www.semanticweb.org/ontologies/2012/3/Ontology1335173950609.owl#Dangerous -->

<owl:Class rdf:about="&Ontology1335173950609;Dangerous">
    <rdfs:subClassOf rdf:resource="&Ontology1335173950609;physical_assets"/>
</owl:Class>
```

```
<!-- http://www.semanticweb.org/ontologies/2012/3/Ontology1335173950609.owl#Dependent -->

<owl:Class rdf:about="&Ontology1335173950609;Dependent">
    <rdfs:subClassOf rdf:resource="&Ontology1335173950609;physical_assets"/>
</owl:Class>



<!-- http://www.semanticweb.org/ontologies/2012/3/Ontology1335173950609.owl#DoS -->

<owl:Class rdf:about="&Ontology1335173950609;DoS">
    <rdfs:subClassOf rdf:resource="&Ontology1335173950609;Threats"/>
</owl:Class>



<!-- http://www.semanticweb.org/ontologies/2012/3/Ontology1335173950609.owl#DuQu -->

<owl:Class rdf:about="&Ontology1335173950609;DuQu">
    <rdfs:subClassOf rdf:resource="&Ontology1335173950609;Worm"/>
</owl:Class>



<!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1335173950609.owl#Equipment_Failure
-->

<owl:Class rdf:about="&Ontology1335173950609;Equipment_Failure">
    <rdfs:subClassOf rdf:resource="&Ontology1335173950609;Source_of_Attack"/>
</owl:Class>



<!-- http://www.semanticweb.org/ontologies/2012/3/Ontology1335173950609.owl#Error -->

<owl:Class rdf:about="&Ontology1335173950609;Error">
    <rdfs:subClassOf rdf:resource="&Ontology1335173950609;Operating_System_Vulnerability"
    />
</owl:Class>



<!-- http://www.semanticweb.org/ontologies/2012/3/Ontology1335173950609.owl#Failure -->

<owl:Class rdf:about="&Ontology1335173950609;Failure">
    <rdfs:subClassOf rdf:resource="&Ontology1335173950609;Operating_System_Vulnerability"
    />
</owl:Class>



<!-- http://www.semanticweb.org/ontologies/2012/3/Ontology1335173950609.owl#Fault -->

<owl:Class rdf:about="&Ontology1335173950609;Fault">
```

```
        <rdfs:subClassOf rdf:resource="&Ontology1335173950609;Operating_System_Vulnerability"
        />
    </owl:Class>



    <!--
    http://www.semanticweb.org/ontologies/2012/3/Ontology1335173950609.owl#Fingerprinting -->

    <owl:Class rdf:about="&Ontology1335173950609;Fingerprinting">
        <rdfs:subClassOf rdf:resource="&Ontology1335173950609;Threats"/>
    </owl:Class>



    <!-- http://www.semanticweb.org/ontologies/2012/3/Ontology1335173950609.owl#Firewalls -->

    <owl:Class rdf:about="&Ontology1335173950609;Firewalls">
        <rdfs:subClassOf rdf:resource="&Ontology1335173950609;Safeguards"/>
    </owl:Class>



    <!-- http://www.semanticweb.org/ontologies/2012/3/Ontology1335173950609.owl#Flood -->

    <owl:Class rdf:about="&Ontology1335173950609;Flood">
        <rdfs:subClassOf rdf:resource="&Ontology1335173950609;Threats"/>
    </owl:Class>



    <!--
    http://www.semanticweb.org/ontologies/2012/3/Ontology1335173950609.owl#Hacker_Attack -->

    <owl:Class rdf:about="&Ontology1335173950609;Hacker_Attack">
        <rdfs:subClassOf rdf:resource="&Ontology1335173950609;Human_Origin"/>
    </owl:Class>



    <!-- http://www.semanticweb.org/ontologies/2012/3/Ontology1335173950609.owl#Hardware -->

    <owl:Class rdf:about="&Ontology1335173950609;Hardware">
        <rdfs:subClassOf rdf:resource="&Ontology1335173950609;Physical_Assets"/>
    </owl:Class>



    <!--
    http://www.semanticweb.org/ontologies/2012/3/Ontology1335173950609.owl#Hardware_Vulnerabil
    ity -->

    <owl:Class rdf:about="&Ontology1335173950609;Hardware_Vulnerability">
        <rdfs:subClassOf rdf:resource="&Ontology1335173950609;Vulnerability"/>
    </owl:Class>
```

```
<!-- http://www.semanticweb.org/ontologies/2012/3/Ontology1335173950609.owl#Human_Origin
-->

<owl:Class rdf:about="&Ontology1335173950609;Human_Origin">
    <rdfs:subClassOf rdf:resource="&Ontology1335173950609;Source_of_Attack"/>
</owl:Class>



<!-- http://www.semanticweb.org/ontologies/2012/3/Ontology1335173950609.owl#Human_error
-->

<owl:Class rdf:about="&Ontology1335173950609;Human_error">
    <rdfs:subClassOf rdf:resource="&Ontology1335173950609;Error"/>
</owl:Class>



<!-- http://www.semanticweb.org/ontologies/2012/3/Ontology1335173950609.owl#ICT -->

<owl:Class rdf:about="&Ontology1335173950609;ICT">
    <rdfs:subClassOf rdf:resource="&Ontology1335173950609;Physical_Assets"/>
</owl:Class>



<!-- http://www.semanticweb.org/ontologies/2012/3/Ontology1335173950609.owl#Information
-->

<owl:Class rdf:about="&Ontology1335173950609;Information">
    <rdfs:subClassOf rdf:resource="&Ontology1335173950609;Logical_Assets"/>
</owl:Class>



<!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1335173950609.owl#Insider_Attack -->

<owl:Class rdf:about="&Ontology1335173950609;Insider_Attack">
    <rdfs:subClassOf rdf:resource="&Ontology1335173950609;Human_Origin"/>
</owl:Class>



<!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1335173950609.owl#Intrusin_Detection_
System -->

<owl:Class rdf:about="&Ontology1335173950609;Intrusin_Detection_System">
    <rdfs:subClassOf rdf:resource="&Ontology1335173950609;Safeguards"/>
</owl:Class>



<!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1335173950609.owl#Logical_Assets -->
```

```
<owl:Class rdf:about="&Ontology1335173950609;Logical_Assets">
    <rdfs:subClassOf rdf:resource="&Ontology1335173950609;Assets"/>
</owl:Class>



<!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1335173950609.owl#Malicious_Code -->

<owl:Class rdf:about="&Ontology1335173950609;Malicious_Code">
    <rdfs:subClassOf rdf:resource="&Ontology1335173950609;Threats"/>
</owl:Class>



<!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1335173950609.owl#Malicious_Code_Dete
ction_And_Elimination -->

<owl:Class rdf:about="&Ontology1335173950609;Malicious_Code_Detection_And_Elimination">
    <rdfs:subClassOf rdf:resource="&Ontology1335173950609;Safeguards"/>
</owl:Class>



<!-- http://www.semanticweb.org/ontologies/2012/3/Ontology1335173950609.owl#Malware -->

<owl:Class rdf:about="&Ontology1335173950609;Malware">
    <rdfs:subClassOf rdf:resource="&Ontology1335173950609;Malicious_Code"/>
</owl:Class>



<!-- http://www.semanticweb.org/ontologies/2012/3/Ontology1335173950609.owl#Mistake -->

<owl:Class rdf:about="&Ontology1335173950609;Mistake">
    <rdfs:subClassOf rdf:resource="&Ontology1335173950609;Error"/>
</owl:Class>



<!-- http://www.semanticweb.org/ontologies/2012/3/Ontology1335173950609.owl#Module -->

<owl:Class rdf:about="&Ontology1335173950609;Module">
    <rdfs:subClassOf rdf:resource="&Ontology1335173950609;Application"/>
</owl:Class>



<!-- http://www.semanticweb.org/ontologies/2012/3/Ontology1335173950609.owl#Network -->

<owl:Class rdf:about="&Ontology1335173950609;Network">
    <rdfs:subClassOf rdf:resource="&Ontology1335173950609;Physical_Assets"/>
</owl:Class>
```

```
<!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1335173950609.owl#Network_Snooping
-->

<owl:Class rdf:about="&Ontology1335173950609;Network_Snooping">
    <rdfs:subClassOf rdf:resource="&Ontology1335173950609;Threats"/>
</owl:Class>



<!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1335173950609.owl#Network_Vulnerabili
ty -->

<owl:Class rdf:about="&Ontology1335173950609;Network_Vulnerability">
    <rdfs:subClassOf rdf:resource="&Ontology1335173950609;Vulnerability"/>
</owl:Class>



<!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1335173950609.owl#Operating_System_Vu
lnerability -->

<owl:Class rdf:about="&Ontology1335173950609;Operating_System_Vulnerability">
    <rdfs:subClassOf rdf:resource="&Ontology1335173950609;Vulnerability"/>
</owl:Class>



<!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1335173950609.owl#Operator_Error -->

<owl:Class rdf:about="&Ontology1335173950609;Operator_Error">
    <rdfs:subClassOf rdf:resource="&Ontology1335173950609;Human_Origin"/>
</owl:Class>



<!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1335173950609.owl#Password_Attack -->

<owl:Class rdf:about="&Ontology1335173950609;Password_Attack">
    <rdfs:subClassOf rdf:resource="&Ontology1335173950609;Threats"/>
</owl:Class>



<!-- http://www.semanticweb.org/ontologies/2012/3/Ontology1335173950609.owl#Passwords -->

<owl:Class rdf:about="&Ontology1335173950609;Passwords">
    <rdfs:subClassOf rdf:resource="&Ontology1335173950609;Safeguards"/>
</owl:Class>
```

```
<!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1335173950609.owl#Physical_Assets -->

<owl:Class rdf:about="&Ontology1335173950609;Physical_Assets">
    <rdfs:subClassOf rdf:resource="&Ontology1335173950609;Assets"/>
</owl:Class>



<!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1335173950609.owl#Physical_Fails -->

<owl:Class rdf:about="&Ontology1335173950609;Physical_Fails">
    <rdfs:subClassOf rdf:resource="&Ontology1335173950609;Equipment_Failure"/>
</owl:Class>



<!-- http://www.semanticweb.org/ontologies/2012/3/Ontology1335173950609.owl#Power_Fails
-->

<owl:Class rdf:about="&Ontology1335173950609;Power_Fails">
    <rdfs:subClassOf rdf:resource="&Ontology1335173950609;Equipment_Failure"/>
</owl:Class>



<!-- http://www.semanticweb.org/ontologies/2012/3/Ontology1335173950609.owl#Protocols -->

<owl:Class rdf:about="&Ontology1335173950609;Protocols">
    <rdfs:subClassOf rdf:resource="&Ontology1335173950609;Logical_Assets"/>
</owl:Class>



<!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1335173950609.owl#Role_Based_Access_C
ontrol -->

<owl:Class rdf:about="&Ontology1335173950609;Role_Based_Access_Control">
    <rdfs:subClassOf rdf:resource="&Ontology1335173950609;Safeguards"/>
</owl:Class>



<!-- http://www.semanticweb.org/ontologies/2012/3/Ontology1335173950609.owl#SCADA -->

<owl:Class rdf:about="&Ontology1335173950609;SCADA">
    <rdfs:subClassOf rdf:resource="&Ontology1335173950609;Physical_Assets"/>
</owl:Class>



<!-- http://www.semanticweb.org/ontologies/2012/3/Ontology1335173950609.owl#Safe -->

<owl:Class rdf:about="&Ontology1335173950609;Safe">
    <rdfs:subClassOf rdf:resource="&Ontology1335173950609;physical_assets"/>
```

```
</owl:Class>



<!-- http://www.semanticweb.org/ontologies/2012/3/Ontology1335173950609.owl#Safeguards -->

<owl:Class rdf:about="&Ontology1335173950609;Safeguards">
    <owl:disjointWith rdf:resource="&Ontology1335173950609;Source_of_Attack"/>
    <owl:disjointWith rdf:resource="&Ontology1335173950609;Threats"/>
    <owl:disjointWith rdf:resource="&Ontology1335173950609;Vulnerability"/>
</owl:Class>



<!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1335173950609.owl#Safety_Instruments
-->

<owl:Class rdf:about="&Ontology1335173950609;Safety_Instruments">
    <rdfs:subClassOf rdf:resource="&Ontology1335173950609;Physical_Assets"/>
</owl:Class>



<!-- http://www.semanticweb.org/ontologies/2012/3/Ontology1335173950609.owl#Scanning -->

<owl:Class rdf:about="&Ontology1335173950609;Scanning">
    <rdfs:subClassOf rdf:resource="&Ontology1335173950609;Threats"/>
</owl:Class>



<!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1335173950609.owl#Security_Adminstrat
ion_Vulnerability -->

<owl:Class rdf:about="&Ontology1335173950609;Security_Adminstration_Vulnerability">
    <rdfs:subClassOf rdf:resource="&Ontology1335173950609;Vulnerability"/>
</owl:Class>



<!-- http://www.semanticweb.org/ontologies/2012/3/Ontology1335173950609.owl#Sniffing -->

<owl:Class rdf:about="&Ontology1335173950609;Sniffing">
    <rdfs:subClassOf rdf:resource="&Ontology1335173950609;Threats"/>
</owl:Class>



<!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1335173950609.owl#Social_Engineering
-->

<owl:Class rdf:about="&Ontology1335173950609;Social_Engineering">
    <rdfs:subClassOf rdf:resource="&Ontology1335173950609;Threats"/>
</owl:Class>
```

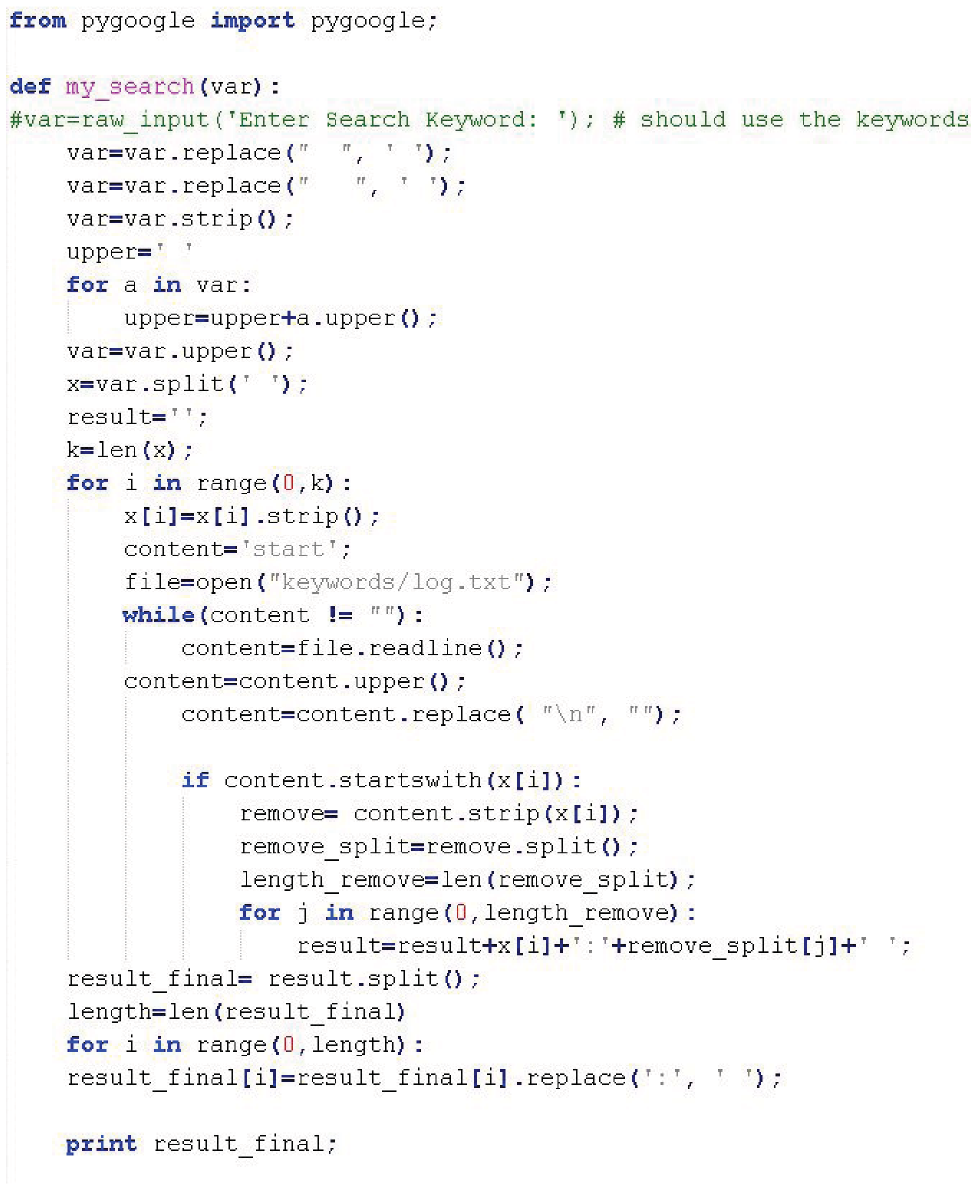
```
<!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1335173950609.owl#Software_Flaws -->

<owl:Class rdf:about="&Ontology1335173950609;Software_Flaws">
    <rdfs:subClassOf rdf:resource="&Ontology1335173950609;Equipment_Failure"/>
</owl:Class>



<!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1335173950609.owl#Source_of_Attack
-->

<owl:Class rdf:about="&Ontology1335173950609;Source_of_Attack">
    <owl:disjointWith rdf:resource="&Ontology1335173950609;Threats"/>
    <owl:disjointWith rdf:resource="&Ontology1335173950609;Vulnerability"/>
</owl:Class>



<!-- http://www.semanticweb.org/ontologies/2012/3/Ontology1335173950609.owl#Spoofing -->

<owl:Class rdf:about="&Ontology1335173950609;Spoofing">
    <rdfs:subClassOf rdf:resource="&Ontology1335173950609;Threats"/>
</owl:Class>



<!-- http://www.semanticweb.org/ontologies/2012/3/Ontology1335173950609.owl#Stuxnet -->

<owl:Class rdf:about="&Ontology1335173950609;Stuxnet">
    <rdfs:subClassOf rdf:resource="&Ontology1335173950609;Worm"/>
</owl:Class>



<!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1335173950609.owl#Symetric_Key_Crypto
graphy -->

<owl:Class rdf:about="&Ontology1335173950609;Symetric_Key_Cryptography">
    <rdfs:subClassOf rdf:resource="&Ontology1335173950609;Safeguards"/>
</owl:Class>



<!-- http://www.semanticweb.org/ontologies/2012/3/Ontology1335173950609.owl#Threats -->

<owl:Class rdf:about="&Ontology1335173950609;Threats">
    <owl:disjointWith rdf:resource="&Ontology1335173950609;Vulnerability"/>
</owl:Class>
```

```
<!-- http://www.semanticweb.org/ontologies/2012/3/Ontology1335173950609.owl#Tolerance -->

<owl:Class rdf:about="&Ontology1335173950609;Tolerance">
    <rdfs:subClassOf rdf:resource="&Ontology1335173950609;Fault"/>
</owl:Class>



<!-- http://www.semanticweb.org/ontologies/2012/3/Ontology1335173950609.owl#Trojan_Horse
-->

<owl:Class rdf:about="&Ontology1335173950609;Trojan_Horse">
    <rdfs:subClassOf rdf:resource="&Ontology1335173950609;Malware"/>
</owl:Class>



<!-- http://www.semanticweb.org/ontologies/2012/3/Ontology1335173950609.owl#User_Error -->

<owl:Class rdf:about="&Ontology1335173950609;User_Error">
    <rdfs:subClassOf rdf:resource="&Ontology1335173950609;Human_Origin"/>
</owl:Class>



<!-- http://www.semanticweb.org/ontologies/2012/3/Ontology1335173950609.owl#Virus -->

<owl:Class rdf:about="&Ontology1335173950609;Virus">
    <rdfs:subClassOf rdf:resource="&Ontology1335173950609;Malware"/>
</owl:Class>



<!--
http://www.semanticweb.org/ontologies/2012/3/Ontology1335173950609.owl#Vulnerability -->

<owl:Class rdf:about="&Ontology1335173950609;Vulnerability">
    <rdfs:subClassOf rdf:resource="&owl;Thing"/>
</owl:Class>



<!-- http://www.semanticweb.org/ontologies/2012/3/Ontology1335173950609.owl#Worm -->

<owl:Class rdf:about="&Ontology1335173950609;Worm">
    <rdfs:subClassOf rdf:resource="&Ontology1335173950609;Virus"/>
</owl:Class>



<!-- http://www.semanticweb.org/ontologies/2012/3/Ontology1335173950609.owl#hardware -->

<owl:Class rdf:about="&Ontology1335173950609;hardware">
    <rdfs:subClassOf rdf:resource="&Ontology1335173950609;Failure"/>
</owl:Class>
```

```
    <!--
    http://www.semanticweb.org/ontologies/2012/3/Ontology1335173950609.owl#physical_assets -->

    <owl:Class rdf:about="&Ontology1335173950609;physical_assets"/>



    <!-- http://www.semanticweb.org/ontologies/2012/3/Ontology1335173950609.owl#systematic -->

    <owl:Class rdf:about="&Ontology1335173950609;systematic">
        <rdfs:subClassOf rdf:resource="&Ontology1335173950609;physical_assets"/>
    </owl:Class>



    <!--
    ///////////////////////////////////////////////////////////////////////////////////////
    //
    // General axioms
    //
    ///////////////////////////////////////////////////////////////////////////////////////
     -->

    <rdf:Description>
        <rdf:type rdf:resource="&owl;AllDisjointClasses"/>
        <owl:members rdf:parseType="Collection">
            <rdf:Description rdf:about="&Ontology1335173950609;Audit_And_Monitoring-Logs"/>
            <rdf:Description rdf:about="&Ontology1335173950609;Firewalls"/>
            <rdf:Description rdf:about="&Ontology1335173950609;Intrusin_Detection_System"/>
            <rdf:Description rdf:about=
            "&Ontology1335173950609;Malicious_Code_Detection_And_Elimination"/>
            <rdf:Description rdf:about="&Ontology1335173950609;Passwords"/>
            <rdf:Description rdf:about="&Ontology1335173950609;Role_Based_Access_Control"/>
            <rdf:Description rdf:about="&Ontology1335173950609;Symetric_Key_Cryptography"/>
        </owl:members>
    </rdf:Description>
    <rdf:Description>
        <rdf:type rdf:resource="&owl;AllDisjointClasses"/>
        <owl:members rdf:parseType="Collection">
            <rdf:Description rdf:about="&Ontology1335173950609;Hacker_Attack"/>
            <rdf:Description rdf:about="&Ontology1335173950609;Insider_Attack"/>
            <rdf:Description rdf:about="&Ontology1335173950609;Operator_Error"/>
            <rdf:Description rdf:about="&Ontology1335173950609;User_Error"/>
        </owl:members>
    </rdf:Description>
    <rdf:Description>
        <rdf:type rdf:resource="&owl;AllDisjointClasses"/>
        <owl:members rdf:parseType="Collection">
            <rdf:Description rdf:about="&Ontology1335173950609;Components_Fails"/>
            <rdf:Description rdf:about="&Ontology1335173950609;Physical_Fails"/>
            <rdf:Description rdf:about="&Ontology1335173950609;Power_Fails"/>
            <rdf:Description rdf:about="&Ontology1335173950609;Software_Flaws"/>
        </owl:members>
    </rdf:Description>
</rdf:RDF>
```

Appendix B

Python Code Listing



```python
from pygoogle import pygoogle;

def my_search(var):
#var=raw_input('Enter Search Keyword: '); # should use the keywords
    var=var.replace("  ", ' ');
    var=var.replace("   ", ' ');
    var=var.strip();
    upper=' '
    for a in var:
        upper=upper+a.upper();
    var=var.upper();
    x=var.split(' ');
    result='';
    k=len(x);
    for i in range(0,k):
        x[i]=x[i].strip();
        content='start';
        file=open("keywords/log.txt");
        while(content != ""):
            content=file.readline();
        content=content.upper();
            content=content.replace( "\n", "");

            if content.startswith(x[i]):
                remove= content.strip(x[i]);
                remove_split=remove.split();
                length_remove=len(remove_split);
                for j in range(0,length_remove):
                    result=result+x[i]+':'+remove_split[j]+' ';
    result_final= result.split();
    length=len(result_final)
    for i in range(0,length):
    result_final[i]=result_final[i].replace(':', ' ');

    print result_final;
    cc=len(result_final);

    Url_search=[];
    Title_search=[];
    res_search={ };

# union of search result to aviod redudancy
    for i in range(0, cc):
    result_search=pygoogle(result_final[i]);
    result_search.pages=2;
    result_title_attr=result_search.search();
    res_search=dict(res_search.items()+result_title_attr.items());
    Title_search=res_search.keys();
    Url_search=res_search.values();

#print Title_search

    len_title=len(Title_search)
    len_url=len(Url_search)

    category=['History:','Case Study:', 'Mechanism:','Prevention:'];

    Weight = [[0 for x in xrange(len(category))] for x in xrange(len_title)]
    count=0;
    file_cat=open('keywords/output_category.txt');
    for j in xrange(len(category)):
        content_cat1=file_cat.readline();
        content_cat=content_cat1.replace( "\n", "");
    content_cat_split=content_cat.split();
    content_cat_split_up=[];
    for up in content_cat_split:
            content_cat_split_up.append(up.upper());
    for i in xrange(len_title):
        array_title1=Title_search[i];
        array_title2=array_title1.replace(",","  ");
        array_title3=array_title2.replace("#","  ");
        array_title=array_title3.replace("&","  ");
        array_title_split=array_title.split();
        array_title_split_up=[];
        for up_t in array_title_split:
        array_title_split_up.append(up_t.upper());
        inter_title_cat=list(set(array_title_split_up) & set(content_cat_split_up));
        if len(inter_title_cat)>0:
                Weight[i][j]=len(inter_title_cat)+0.1*j;

    Result={};
    for j in xrange(len(category)):
    Result[category[j]]=list();


    for i in xrange(len_title):
    Max_weight=max(xrange(len(Weight[i])), key=Weight[i].__getitem__);
    Result[category[Max_weight]].append((Title_search[i],Url_search[i]));

    print Weight
#print Result

    for j in xrange(len(category)):
    print category[j]
    print'*****************************************************'
    for k in xrange(len(Result[category[j]])):
        print Result[category[j]][k]
        print '---------------------'

    return Result
```