Ontology-Based Recommendation of Academic Papers

by

Esraa Hassan Jawad Al-Wakel

Bachelor of Computer Science, University of Babylon, 2001

A Thesis Submitted in Partial Fulfillment
of the Requirements for the Degree of

Master of Computer Science

in the Graduate Academic Unit of Computer Science

Supervisor(s): Ali A. Ghorbani, Ph.D., Faculty of Computer Science
Ebrahim Bagheri, Ph.D., Faculty of Computer Science

Examining Board: Weichang Du, Ph.D., Faculty of Computer Science (Chair)
Bruce Spencer, Ph.D., Faculty of Computer Science
Donglei Du, PhD., Faculty of Business Administration

This thesis is accepted by the
Dean of Graduate Studies

THE UNIVERSITY OF NEW BRUNSWICK

February, 2014

ABSTRACT

In an era when recommender systems aspire to reduce information overload, we analyze how recommender systems can be implemented to overcome current limitations. This thesis presents a novel framework for a semantic recommender system that not only copes with existing problems but also presents a strategy that computes customized recommendations using a variety of tools including semantic contents. To this end, we have identified the need for developing semantic recommender systems, which are able to extend existing systems and perform a semantic search in an effort to find the most suitable scientific papers in the field of Computer Science. For this purpose, we developed three different semantic recommender techniques rooted in annotation systems and its semantic matching components. The techniques, which are entitled REI, REII, and REIII, are based on GATE, Alchemy API, and a combination of both tools. These recommender techniques are capable of exploring an annotated database in an attempt to trace and rank the most relevant documents in a particular query. Precision and recall are subsequently measured and compared to a similar query conducted in Google Scholar indicating that this research is promising and can improve on current semantics-based recommender systems.

DEDICATION

This work is dedicated to the memory of my beloved uncle Kazem who encouraged me to complete my studies. I lovingly dedicate this thesis to my parents as well: All I have and will accomplish are only possible due to their love and sacrifices throughout my life.

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my supervisors Dr. Ebrahim Bagheri and Dr. Ali Ghorbani Without their patient guidance, insightful ideas, and invaluable reflections, this dissertation could not have been completed. Thank you for guiding me and empowering me to develop great knowledge in my field of expertise. I have been truly blessed to work with supervisors who cared so much about my work and who responded so eloquently to my questions and queries.

I would also like to thank my committee members, who served in the examining board of my thesis and helped bring this thesis to fruition.

My appreciation and sincere thanks also goes out to Iraq's Ministry of Higher Education and the Cultural Office in the Iraqi Embassy in Canada for their support.

I would also like to express my appreciation and warmest thanks to my lovely friends Faezeh, Soudeh, Asmaa, and all my friends who have helped me and encouraged me to complete this work.

I would also like to give my thank you to Dr. Boon and the staff at the UNB health clinic who were extremely supportive as I dealt with ongoing health issues.

Finally, I would like to thank my wonderful father and lovely sisters for their constant support and prayers. My mother's memories have also been a source of inspiration and strength in moments of despair.

Table of Contents

ABSTRACT ii
DEDICATION iii
ACKNOWLEDGEMENTS iv
Table of Contents v
List of Figures viii
1 Introduction 1
1.1 Motivations 3
1.2 Objectives 5
1.3 Contributions 6
1.4 Thesis Organization 8
2 Related Work 10
2.1 Recommender Systems 10
2.2 Ontology Based Information Extraction 15
2.3 Semantic Search Engines 18
2.4 Semantic Annotations 19
2.5 Recommended Systems and Mechanism for Automated Annotations 20
2.6 LOD-Based Annotation Systems 22
3 Semantic Based Recommender System 23
3.1 System Overview 23
3.2 Domain-Specific Ontology 26
3.3 Repository of Domain-Specific Documents 26
3.4 Domain-Specific Ontology-Based Annotation Component 27
3.4.1 GATE Components 28

3.4.2 The Annotation Strategy 33
3.5 LOD-Based Annotation Component 35
3.5.1 Alchemy 35
3.5.2 The Annotation Strategy 36
3.6 Semantic Similarity Measuring Component 36
3.7 Recommender Engine I 38
3.7.1 Algorithm 38
3.7.2 Explanation and Example 38
3.7.3 Discussion 42
3.8 Recommender Engine II 43
3.8.1 Algorithm 43
3.8.2 Explanation and Example 43
3.8.3 Discussion 47
3.9 Recommender Engine III 47
3.9.1 Algorithm 47
3.9.2 Explanation and Example 48
3.10 Differences Between the RE's Outcomes 52
3.10.1 Discussion and Example 52
3.10.2 Justifications 56
4 Implementation 58
4.1 The Repository of Domain Specific Documents 58
4.2 Domain-Specific Ontology 58
4.3 Domain-Specific Ontology-Based Annotation Component 60

4.3.1 Loading the CREOLE Plugin 61

4.3.2 Creating a new corpus 61

4.3.3 Loading our Ontology: 63

4.3.4 Creating and Running an Application (Corpus Pipeline) 64

4.3.5 Export Annotation to Repository 66

4.4 LOD-Based Annotation Component 66

4.5 Semantic Similarity Measuring Component 66

4.6 Recommender Engine I (REI) 67

4.7 Recommender Engine II (REII) 68

4.8 Recommender Engine III (REIII) 69

5 Evaluation 71

5.1 Introduction 71

5.2 Evaluation Results 72

5.3 Questions Description 75

5.4 Results 76

5.5 Precision and Recall 78

6 Conclusion and Future work 86

6.1 Conclusion 86

6.2 Future Work 88

Bibliography 90

Appendix 99

Curriculum Vitae 115

List of Figures

Figure 3-1 Components of our Ontology-Based Recommender System. 25

Figure 3-2 ANNIE English Tokeniser on a Sample Document. 29

Figure 3-3 ANNIE Sentence Splitter on a Sample of Document. 30

Figure 3-4 Building Ontology Resource Root (OntoRoot) Gazetteer from the Ontology. 32

Figure 3-5 Doc1 Contents. 54

Figure 3-6 Doc2 Contents. 54

Figure 4-1 The ACM Computing Classification System (CCS). 59

Figure 4-2 Domain-Specific Ontology Protégé. 60

Figure 4-3 The CREOLE Plugin Manager. 61

Figure 4-4 Creating Corpus. 62

Figure 4-5 Loading Ontology in GATE. 63

Figure 4-6 GATE Application. 64

Figure 4-7 Output Example. 65

Figure 4-8 SM Sample. 67

Figure 4-9 Sample REI Results. 68

Figure 4-10 Sample REII Results. 69

Figure 4-11 Sample REIII Results. 70

Figure 5-1 Sample of One Survey Question. 74

Figure 5-2 Rating Top Ten Papers by Four Search Engines for the Ontology Query. 76

Figure 5-3 Rating Top Ten Papers by Four Search Engines for Semantic Web Query. 77

Figure 5-4 Top Ten Papers by Four Search Engines for Web Service Query. 77

Figure 5-5 Precision of Four Search Engines for Ontology Query. 79

Figure 5-6 Precision of Four Search Engines for Semantic Web Query. 79

Figure 5-7 Precision of Four Search Engines for Annotation Query. 80

Figure 5-8 Precision of Four Search Engines for Social Network Query. 80

Figure 5-9 Precision of Four Search Engines for Web Service Query. 81

Figure 5-10 Recall of Four Search Engines for Ontology Query. 82

Figure 5-11 Recall of Four Search Engines for Semantic Web Query. 83

Figure 5-12 Recall of Four Search Engines for Annotation Query. 83

Figure 5-13 Recall of Four Search Engines for Social Network Query. 84

Figure 5-14 Recall of Four Search Engines for Web Service Query. 84

Chapter 1

1 Introduction

Information overload for the typical user surfing the Internet has led to an increasing demand for recommender systems. The central cause for information overload is the loss of significant data and the slow pace of work on the part of the user. Information overload will happen when users deal with more information than they are capable of coping with at any one point in time. This prohibits users from taking advantage of and processing information efficiently. Information overload prompts users to either make the wrong decisions or to delay making a decision altogether [1].

Recommender applications are powerful technologies that make personalized recommendations for users in terms of interesting choices available to them. They are a subclass of *information filtering* that search to estimate the 'score' or 'choice' that a person is likely to give various items. These items can range from audio, books, or films, to social components like people or groups. One of the top priorities in incorporating recommender systems is to offer personalized recommendations to users regarding various products and services. This outlook presents users with a fresh set of opportunities to find what they are interested in [29].

Recommender systems have become an integral and important part of e-commerce and on-line information systems. Aside from being essential navigational tools, recommender systems present powerful methods and techniques for e-commerce

applications. As an important component of e-commerce, recommender systems rely on the top overall sellers and the past behaviour of buyers in recommending items and presenting the necessary information to customers so that they can choose their items of interest for purchase. Amazon, eBay, CDNOW, and Moviefinder are considered as good examples of e-commerce websites that successfully apply recommender systems to guide users [29].

There are a broad range of applications that are using recommender systems, such as recommendation for academic papers [28], news [27], movies [24] [12], restaurants [12], CDs [23], music tracks [26], and web pages [25]. These applications provide reliable and automated suggestions to the users regarding the items that they might wish to choose, watch, study, purchase, or check. The capability of helping users locate potential products of interest to purchase in the wealth of online information is revolutionary.

Users can selectively attain what they need from a huge volume of information for any particular item. The sites that employ recommender systems are intent on presenting information on services and products that are most likely to be of interest for their audience. In presenting the recommendations, the recommender system uses facts from the registered user's account and exhibited behaviour patterns by other groups of people with similar interests and evaluates the data in order to reference features with the intention of offering recommendations [31].

1.1 Motivations

The earliest and most widely used recommendation approach is known as Collaborative Filtering (CF). CF creates a customer dataset from the user's past preferences, such as their website visits, item selections and purchases, and then recommends items for the target user based on the preferences of other users who are recognized to be similar to him or her [2], [7]. The CF method finds the rating of an item for a target user by analyzing the item-ratings table of other close users called *neighbors* who have similar and common interests as the target user. Such a method needs to have access to the ratings of all users and their used items.

CF approaches have some weaknesses. The most important problems are the following:

- Cold start [4] – since CF recommenders depend on the past performance of users, they cannot provide accurate recommendations for new users because they do not have enough background information about them. In addition, they cannot work well for recommending new items because there are not enough ratings from users for the new added items. Many researchers use hybrid recommendation approaches to overcome this problem [1].
- Sparsity – the number of items in any recommendation system is very large, so the rating number of popular items will be very small.

Content-based approaches recommend an item to a target user based on the common characteristics and features of the item related to the other items that are close to his or her preference and interest [6]. Content-based filtering commonly

focuses on recommending items that indicate textual information such as documents, papers, and news items because of the importance of several text-based applications and also because of the developments from the information retrieval and filtering communities. In the case of textual information, content-based recommend systems usually depict items by the weight of the most informative keywords in their contents. Content-based recommendation systems find information about the user's interests, needs, and preferences explicitly (e.g. by the ratings provided by the users) or implicitly (e.g. by analyzing what a user reads, clicks, and chooses) and save them in *user profiles*. These recommendation systems use *item profiles* for storing and managing information about items. The item profile contains a set of item features that come either automatically by using text classification algorithms or explicitly from the features of other similar items from other item profiles and user profiles. [4][6][8].

Content-based filtering overcomes the problem of CF, such as cold start and sparsity [8]. However, content-based filtering has weaknesses of its own, including limitation of keyword-based systems and overspecialization [6] [8]. The first limitation stems from the fact that content-based filtering depends on the similarities between items. When the system has insufficient semantic-based information about an item, it can fail to notice semantically similar items correctly, or it may retrieve many irrelevant items [8]. For example, a content-based recommendation system cannot recognize the terms "Programming Language," "C++," and "Visual Basic" as similar terms to "Java," if it interprets "Java" as an "island of Indonesia" instead of a

“programming language.” This problem can appear in the traditional content methods that represent a document as a vector of keywords [9]. Another problem is overspecialization, which comes from the fact that a user is limited to recommending items that have a high rating against the user’s profile, so the recommendation will be made of items that are similar to the items that have a good rating. In this case the recommender system may ignore different but still interesting items [1] [6] [8].

In order to overcome the aforementioned problems, we decided to develop an ontology-based recommendation. Using ontologies in content-based filtering for the representation of user profiles and item profiles can overcome the problems of this approach. Thus, items that are described in simple natural language terms will be mapped to an external knowledge base such as a domain ontology and the similarity between these representations will be calculated based on semantic analysis and measures [9] [8].

Using ontological inferences can result in a better user profiling; consequently, it can reduce the effect of cold-start problems and provide more accurate recommendations. Through the utilization of ontology, a recommendation system can access more information about items in item profiles and user profiles and hence provide more accurate recommendations [4]. In an effort to represent the context, ontology is used as a formal, explicit specification of a shared conceptualization [30].

1.2 Objectives

The main objective of this thesis is to procure an accurate recommendation system by extending the traditional content-based filtering methods by using semantic

web technologies to overcome the most popular problems found in the existing recommender system approaches. As discussed in the motivation section, content-based filtering has some weaknesses that include limitations of the keyword-based system and overspecialization. By extending the traditional content-based filtering methods, we are able to overcome these problems. Ontologies consist of concepts, instances of concepts and relationships between them. Textual items that have been annotated with ontological information contain more meaningful semantically rich information [13] and can be exploited by recommender systems for more accurate recommendations. Annotation can connect descriptions in the ontology with their concepts and relations in the text. The application area of our work that is used for testing our algorithm is comprised of academic papers in the field of Computer Science.

1.3 Contributions

In this thesis, we designed and implemented a semantic recommender system that allows an individual to perform a semantic search and discovery for scientific papers in the field of Computer Science. This system recommends and ranks documents based on the user's topics of interest. Our semantic recommender system uses ontology-based annotation components to find and annotate the main topics of a document. It also uses semantic-based matching techniques to measure the cumulative distance of the main topics of a document to any given query phrase and then ranks documents based on these distance measures. We implemented our system

in the domain of Computer Science academic papers and evaluated and compared its performance. Our contributions can be summarized as follows:

1. We developed an annotator system for annotating academic Computer Science papers. Our annotation system utilizes two highly efficient and reliable technologies namely GATE (General Architecture for Text Engineering) and Alchemy. Both of these annotation technologies are highly commended by professionals in the field and are capable of generating exceedingly precise and comprehensive annotations.
2. We developed a semantic search engine based on the annotation system and its corresponding semantic matching component. This system can search over an annotated dataset and find and rank the most relevant documents to a given query. We developed three different systems based on Gate, Alchemy, and their combination.
3. We implement our system in the domain of computer science academic papers by developing a Computer Science *subjects* ontology which contains information about subjects in the field of Computer Science. Our recommender system can recommend a paper in accordance to a user's various topics of interest.
4. We evaluated the performance of our search engine systems to retrieve the maximum number of relevant papers. We measured both precision and recall to compare our results with Google Scholar.

1.4 Thesis Organization

The current thesis is divided into six chapters. In the first chapter, we provide an overview and background of the proposed research. This includes exploring the circumstances surrounding the increasing demand for recommender systems. We provide a thorough description of recommender systems and how we use semantic web technologies to expand and improve on previous content-based filtering methods used in recommender systems. This section highlights the motivation behind the research, the aim and objectives of the work, and its contributions to the field of Computer Science.

The second chapter discusses a more nuanced understanding of recommender systems, along with other basic concepts including ontology-based search engines, ontology-based information retrieval, and semantic search engines. We explore how newfound interest in recommender systems is paving the way for recommender systems to be used as a means to reduce search ambiguities for users and obtain optimal results in the process. We point out that one of the aims of this research is to solve the limitations of previous recommender system models that miss the mark in producing results that are most relevant to particular user preferences.

Chapter three begins with a detailed discussion of the proposed approach in designing a semantic-based recommender system. We present a general overview of the system components and explore the various constituents found in the ontology-based recommender system designed for this particular study.

In chapter four, we describe the implementation of our approach for designing a semantic-based recommender system. The implementation of all system components will be described in meticulous details. In chapter five, we describe how we evaluate our systems based on precision and recall measures.

In the sixth chapter we conclude this study with a summary of our contributions and a discussion of possible future work. The concluding chapter reflects on the implications of this research as we summarize the outcomes of our work and outline directions for future research in this area.

Chapter 2

2 Related Work

The aim of this chapter is to provide an introductory framework for describing the general concept of recommender systems, ontology-based search engines, ontology-based information retrieval, and semantic search engines.

2.1 Recommender Systems

Recommender systems initially emerged as intelligent information filtering tools where users were offered recommendations and the system utilized the outcome to better meet the needs and demands of the target user [32]. The role played by recommender systems is reminiscent of a personalized decision guide. The growing interest in recommender systems has captured the imagination of experts and raised unlimited potential for research. The wealth of knowledge and personalized recommendations offered by recommender systems to users can assist them in dealing with information overload.

Recommender systems have advanced consistently over the years but still require considerable improvements in order to make their methods more effective and applicable to a wider audience and a broader range of applications. There are a myriad of ways that can assist recommender systems in order to make them state-of-the-art. Some of these include more enhanced ways of representing user behavior and the inclusion of a greater degree of contextual information among other things. In

order to accomplish all they are set out to do, recommender systems rely on user profiling and information filtering in order to provide users with concrete product or service recommendations that are in close proximity to the users' preferences and needs, in terms of features.

In time, the term recommender system has come to enjoy a broad meaning, depicting any system that generates individualized recommendations or has the potency to direct the user in a customized manner toward functional questions in an extensive space of conceivable alternatives. Such systems have a clear purpose in an environment where the measure of on-line information tremendously outperforms any singular competence to study it. Recommender systems are currently an essential part of various ecommerce destinations, as can be seen in the case of Amazon.com [33]. This section imparts the most popular approaches used in recommendation techniques proposed by researchers: Collaborative recommendations, Content-based recommendations, and Hybrid approaches [1] [3] [16].

Collaborative filtering is a technology utilized primarily to predict preferences for users particularly with regards to item selection or ratings. Collaborative filtering methods finds the rating of an item for a target user by analyzing the item-ratings table of other like-minded users called neighbours who have similar concerns and conceivably common interests with the target user. In order for collaborative filtering to be efficient, it needs to have access to the ratings of all users in an effort to make an impartial representation of the ratings as relevant to the target user. Collaborative recommender systems total appraisals or recommendations of articles, distinguishes

commonalities between users on the basis of their evaluations and generates new recommendations based on between-user examinations [34]. A common user profile in a collaborative system is comprised of a vector of items and their evaluations, continuously enlarged as the user connects with the system over a long period of time.

Collaborative filtering involves user-user similarity calculations through the utilization of user neighbourhoods to predict new products or services that are potentially of interest to the target user. In such a system, the relevance to the user is highlighted in the ongoing effort to improve the quality of recommendations. Predicting a user's inclination toward a product or service by linking the user's records with the recorded interests of a community of individuals is the central aim of collaborative filtering. Consequently, information that deals directly with the content of the item in question including words and descriptions are replaced with usage or preference patterns seen in other similar users who are of a similar mindset to the target user. Therefore, it would not be too far off to presume that collaborative filtering is based on the assumption that an effective method of coming to contact with interesting content for the target user is to find other individuals with a similar comparable interest in a particular content. That particular content, will of course need to possess characteristics that are of similar interest to the target user followed by the recommendation of titles that those like-minded users favour.

Content-based recommendation is an extension and continuation of information filtering research [35]. Content-based filtering predicts the user's preferred items

depending on similar features of those very items. Meanwhile, the hybrid approach combines the positive features of both collaborative and content-based approaches in order to predict items that the user is likely to prefer.

As already pointed out, the goal of content-based recommendation systems is to suggest items that are similar to those a particular user with a mindset similar to the target user has liked in the past. In order to accomplish this objective, content-based recommender systems match up characteristics and attributes of a user profile where the preferences and interests of the individual is stored with the features of particular contents or items that might be of interest to the user. At this point, the system confirms receiving information that explicitly describes the nature of a particular product or service. The next obvious step is to predict which items the target user is likely to find desirable based on the sample the recommender system has of the user's preferences. This form of content-based filtering does not depend on social information and instead is a reflection of the non-ratings information available.

This particular form of filtering system employs information received from a user profile that represents the content description of items the user had shown interest in previously. The function of such systems is to primarily compare extracted features from items that have yet to be rated by users or with content description in the profile of the user. It is through this analysis that items that are believed to be considerably similar to the user profile are recommended by the system to the target user. Content-based filtering systems that are more commonly utilized in ecommerce

applications generally derive content descriptions from textual features that have been extracted from product descriptions.

Despite their impressive contributions, content-based filtering systems have their disadvantages as well. The biggest downfall of these systems is their penchant to overspecialize the item selection, especially considering the fact that the profiles they derive their information from merely retain the user's previous rating of items. An additional point that tends to be ignored is that it may not be practical to assume that content-based filtering systems require that items be represented effectively using extracted textual features considering the heterogeneous nature of data found on the World Wide Web. Consequently, items that do not possess the precise characteristics found in the user profile may not get recommended despite their potential similarities, especially if synonyms are used when describing keyword terms.

The hybrid approach avoids the limitations of collaborative and content-based methods by combining them together in different ways in an effort to improve system performance [11]. The work in [22] enhances the performance of content-based and collaborative filtering methods by combining the separate vector rankings of these filtering methods in one list in order to get the best recommendation program of a television viewing system (PTV) [11] [20]. For a hybrid content-based collaborative filtering system to be effective, user profiles are used in accordance with content analysis. Subsequently, the profiles are compared in an effort to establish a baseline of similar users for collaborative recommendation. Accordingly, users are presented with items in two instances: the first one is when they score highly against their own

profile and the second is when they get high ratings from users that have a similar profile to the target user. It is the hybrid nature of this approach that circumvents the restrictions faced by content-based and collaborative systems while at the same time presenting users with new possibilities.

2.2 Ontology Based Information Extraction

The central purpose of Information Extraction (IE) is to locate specific kinds of structured information from the natural-language text. This mechanism was essentially produced for enterprises working on online platforms that use web services. Generally, these enterprises need terminology extraction mechanisms for the development of web crawlers and intelligent bots among other things [49]. According to Norvig and Russell, IE seeks to process natural language texts and obtain occurrences of the specific class of objects or events and occurrences of the relationships among them [36]. Riloff echoes these views by stating that information extraction is a form of natural-language processing in which certain kinds of data should be identified and removed from text [63]. An instance of an information extraction system at work is the extraction of data from various web pages regarding certain countries using a model that will specify what particular information is sought after taking into consideration the political, social, and financial indicators of the nations in question [63].

Ontology-Based Information Extraction (OBIE) has come out as a sub-field of information extraction. Here, ontology is utilized by the info extraction process and the result is usually offered via ontology [37]. It should be pointed out that ontology

is understood to be a formal and explicit specification of shared conceptualization. Broadly speaking, ontology is known for specific areas. Because data extraction is mainly focused on locating information for a specific site, formally and explicitly indicating the principles of the site via ontology could be useful to this method. For instance, a geo-political ontology that describes ideas like land, nation, and town may be used to steer the information extraction process. This is actually the basic idea behind ontology-based information extraction [37]. Thus, it can be concluded that IE is the act of detecting specific pieces of information from a given document or content. Using this method creates a space where useful structured data is obtained from unstructured text. To be more precise, information extraction is the process whereby a set of substrings are found for a set of specified slots within a document known as fillers.

Although OBIE has gained a favorable reputation in the past few years its actual origin could go as far back as the late 1990s, particularly in the work of Hwang in 1999, regarding creating ontology from text. Ontology-Based Information Extraction's surge in popularity can be seen in the increase in the number of publications regarding this topic. This, along with the fact that interest in information extraction is generally on the rise, is an indication that this field is experiencing substantial development [38]. Researching the implementation of information on different OBIE methods and showing the measurements that scientists pursued to assess the efficiency of OBIE systems may also prove to be useful in developing new OBIE systems [38].

Ontology locates two separate research areas in literature with virtually little or no features in common: the world of formal ontology specification and the world of ontologies for language-related AI (Artificial Intelligence) tasks [39]. The word ontology has come to be defined in a variety of ways including data, taxonomies, schema, formal ontologies, and inference. Ontologies may have been intended as an important and natural means of representing real world knowledge for the development of database designs [40]. In line with the findings of Neches et al. [41], ontology is a formal explicit description of concepts in a domain of discourse [60]. Overall, ontology is defined as an explicit specification of a conceptualization [42]. Gruninger and Lee [43] identify the uses of ontologies in the following manner:

- Communication
    - between implemented computational systems
    - between humans
    - between humans and implemented computational systems
- Computational inference
    - for internally representing plans and manipulating plans and planning information
    - for analyzing the internal structures, algorithms, inputs, and outputs of implemented systems in theoretical and conceptual terms
- Reuse (and organization) of knowledge
    - For structuring or organizing libraries or repositories of plans and planning and domain information [43].

2.3 Semantic Search Engines

Semantics are the most significant instrument used in finding any type of data on the Internet [44]. Semantic search seeks to enhance the reliability of web searching by taking into account the relevance and importance of context or meaning of the conditions so as to increase the probability of attaining more desirable web documents. Google and Bing search engines incorporate several semantic search instruments and make use of the science of meaning in language in order to create more appropriate results for users [45]. It is evident that the aim of semantic browsing is to provide data in a meaningful context as opposed to having to sort through lists of papers bound by key words [45]. Semantic search tools are indisputably striving to change our relationship with online searching. Semantic sources on the net will most likely depend on metadata to explain and collect files during information access. Meta-data is understood by the researchers to be “data about data" [46]. Semantic search engines depend on four approaches to search about their queries:

- Use contextual analysis
- Focus on reasoning
- Emphasize on natural language understanding
- Use Ontologies

Furthermore, some semantic search engines utilize more than one approach in order to get more accurate and relevant search results, Sudeepthi points to four approaches to semantic search that are utilized in some capacity by various search engines. These approaches are comprised of the use of meaning to improve the user’s

search experience, focus on reasoning, an emphasis on natural language understanding, and using ontology to represent knowledge about a domain and expand queries. The available semantic search engines tend to 'mix and match' these approaches in an effort to create the most efficient search experience for users [47]. In describing the architecture of the semantic search engine, Kassim and Rahmany maintain that such a search engine should be composed of various components including Ontology development, Ontology Crawler, Ontology Annotator, Web crawler, Performing semantic search, Query builder, and Query pre-processor[64]. The superiority of semantic search engines to traditional is undeniable. Perhaps the most outstanding difference between the two systems is that the former is incapable of dealing with domain knowledge and is thus unable to understand the meaning of the search request and the inherent links between the terms in a document; Semantic search engines understand the complexities of natural languages and recognize how best to employ domain knowledge in order to efficaciously impact the precision and recall of the search results [65].

2.4 Semantic Annotations

As the cornerstone of Semantic Web, Semantic Annotation (SA) is used in knowledge management and creating metadata. SA refers to the procedure of indexing and recovering service knowledge from records and consequently making annotation or metadata. These annotations or metadata are properly demarcated for a specific domain utilizing appropriate sentence structure and semantics. Accordingly,

the principal objective of SA is to make metadata that could be used by both humans and machines. As proposed by [48], the semantic web can help us achieve this objective as it proposes the utilization of space ontologies as semantic guides to annotate report content. Along these lines, semantic annotation is the procedure of mapping textual components discovered in texts with ontological notions.

In semantic web, the domain ontology is a fundamental asset for SA. Ontology is characterized as formal and unequivocally restricted to imparted conceptualization [62]. It represents knowledge in a structured form suitable for gathering knowledge and contemplating it. SA is comparably described as a methodology where semantic data is added to linguistic structures.

2.5 Recommended Systems and Mechanism for Automated Annotations

Considering that in our annotation components we will use GATE and Alchemy, both will be explained in greater detail in the following section:

I. **GATE** – The techniques depicted in this thesis are based on the GATE architecture. GATE encompasses features that are a considerable improvement on previously celebrated tools such as RapidMiner and NLTK [50]. One of GATES' main advantages which could indeed be interpreted as its greatest strength, is its ability to produce an exclusive KIM semantic base [50]:

- An information extraction feature is integrated within GATE, known as A Nearly New Information Extraction system or ANNIE. Its function is to handle tokenisers, splitters, taggers, gazetteers, etc.
- GATE is capable of handling multiple languages including English, French, Arabic, Chinese, etc [59].
- In order to increase efficiency and effectiveness, certain Plugins are also available.
- Several formats of input files, databases, and storage devices are supported by this tool.
- GATE uses JAPE transducers that are exclusively used to manipulate annotations [51].

Generally speaking, GATE is comprised of two principle functionalities:

1. GATE Developer is an open source desktop application composed in JAVA that gives a client interface for expert linguists and text specialists to unite a wide mixture of text examination apparatuses and apply them to a report or set of reports. Door Developer incorporates numerous NLP instruments as modules. Some have been created in-house, others have been composed explicitly for GATE, and there are those that have been ported from stand-alone open-source instruments.
2. GATE Teamware is an electronic administration stage for communitarian annotation. It conveys a multi-function client interface over the web for review and including text annotations.

II. **Alchemy** – Alchemy API is recognized as one of the most popular and well known natural language processing platforms. Researchers have worked to generate annotations using this tool. One of the strong suits of Alchemy API is the provision of cloud-based services along with the on premise services (on premise text analysis infrastructure). This feature reduces the complexity of integrating and processing large volumes of natural language generated though applications of data processing pipelines. It is also capable of analyzing and tagging data extracted automatically from previously defined data processing pipelines, services, etc.

2.6 LOD-Based Annotation Systems

Linked Open Data (LOD)-based annotation systems find the main annotation of any document based on linked data and semantic web. Wikipedia Miner, Tagme, Alchemy, and Textrazor are prime examples of LOD-based annotation components. It may be characterized as the dynamic semantic enrichment of unstructured and semi-structured documents and the linking of these to relevant domain ontologies/knowledge bases. Semantic annotation is understood from a text-mining perspective in the sense that it deals with annotating everything from ontology through metadata in texts. This task is carried out by referring to their Unique Resource Identifiers (URIs) in ontology [52]. LOD resources are not only used as target entity knowledge bases for semantic enrichment and resource linking but also serve as central sources for large-scale ontological knowledge [53].

Chapter 3:

3 Semantic Based Recommender System

In this chapter, we will describe the approach utilized for designing a semantic-based recommendation system. We will illustrate a general overview of the system components and then explain each component in greater detail.

3.1 System Overview

The components of our ontology-based recommendation system are illustrated in Figure 3.1. As seen in Figure 3.1, the ontology-based recommender system we have designed is comprised of the following components:

- Domain-specific ontology – An ontology used to represent the main subjects of the domain and the relationships between them in a formal way.
- Repository of domain-specific documents – This repository includes a set of domain-specific documents. The ontology-based recommender system can semantically search over these documents and provide recommendations accordingly.
- Domain-specific ontology-based annotation component – This component is responsible for annotating a domain-specific document based on domain-specific ontology. During the annotation phase, this component takes an ontology, finds the instances of the ontology concepts in the documents, and exports annotations into a specified repository.

- LOD-based annotation component – This component finds the main topics, keywords, and concepts of documents in the domain-specific based on open linked data knowledge bases such as Wikipedia.
- Semantic similarity measuring component – This component finds similarities between the annotations of documents in the repository of domain-specific documents and terms in a given query phase. It finds and calculates semantic similarity scores between words based on semantic similarity as opposed to syntactic similarity.
- Recommender engine I – This component uses the domain-specific ontology-based annotation component and the semantic similarity-measuring component and ranks documents based on similarity scores in order to provide users with highly specific and comprehensive documents from the repository.
- Recommender engine II – This component uses an LOD-based annotation component and a semantic similarity-measuring component and ranks documents based on the similarity of scores in order to provide users with recommendations of documents from the repository suited to their needs.
- Recommender engine III – This component works based on an integrated view of the annotations provided by a domain-specific ontology-based annotation component and an LOD-based annotation component. This engine ranks documents based on the scores returned by the semantic similarity component.

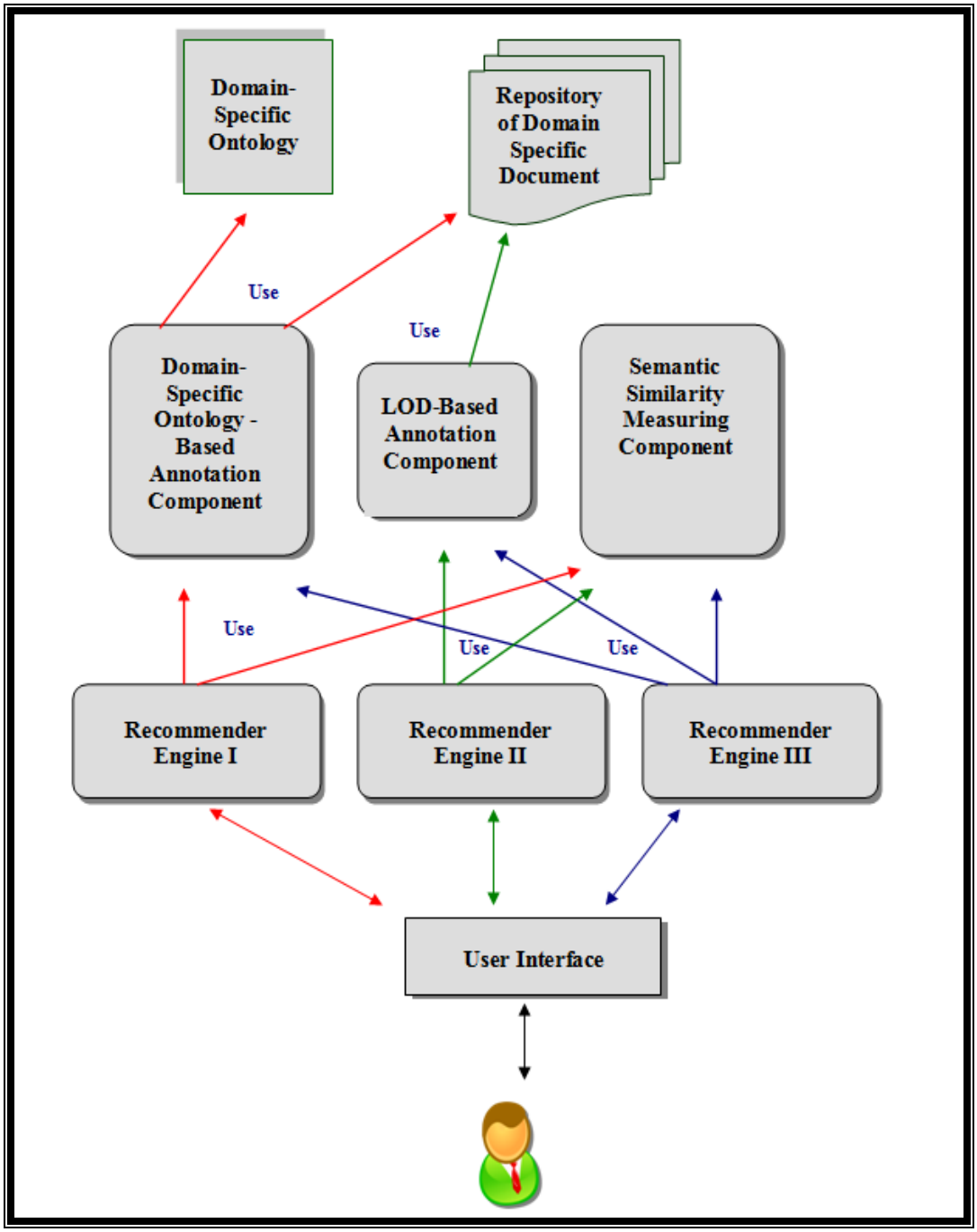

Figure 3-1 Components of our Ontology-Based Recommender System.

In the next section of this chapter, we will explain each component in greater detail and present an example in an effort to bring the recommendation systems into focus.

3.2 Domain-Specific Ontology

Using ontology for knowledge representation has its advantages, including the fact that explicit specification consistently makes knowledge sharing, knowledge understanding, as well as knowledge acquisition tasks considerably easier. Ontologies can represent objects, concepts, and other entities, but also the relationships that exist between them [54]. In order to ensure that a formal ontology is an appropriate specification of a domain, the conceptualization (concepts, their relations and constraints) of the domain must possess the appropriate features as well [55]. In our ontology-based recommender system, we use a domain-specific ontology that explicates the main concepts of the domain and their interrelationships. The ontology contains concepts, specific instances, and a set of parent-child relationships.

3.3 Repository of Domain-Specific Documents

This component contains a set of domain-specific documents. We will use this repository for all of the information retrieval tasks, along with annotating, searching, and providing semantic recommendations to users. As we will point out in Chapter 4, in our implementation of the system, this repository includes a set of academic

publications in the field of knowledge engineering, semantic web and search engines in Computer Science. Of course, this repository can be customized to any set of domain-based documents and texts in a different implementation of this system. Domain-specific ontology and the domain-specific repository of documents have a strong interrelationship: i.e. the more the main concepts in documents are covered by the ontology, the stronger semantic analysis our system can provide.

3.4 Domain-Specific Ontology-Based Annotation Component

This component annotates domain-specific documents in the repository base using domain-specific ontology. For the sake of annotation, we used General Architecture for Text Engineering (GATE). GATE contains functionality for various kinds of natural language processing tasks. One of the most prominent features of GATE is its ability to annotate documents. GATE provides facilities for domain experts to manually annotate documents with instances of a given ontology. In addition, it supports semi-automatic and automatic methods for annotation. The following features of GATE have been employed in our work:

- Language Resources (LRs) – This includes ontologies and documents that have been loaded in the system, in our case, all documents in the repository of domain-specific documents, and corpus.
- Processing Resources (PRs) – The tools used to create annotations or representing resources that are principally algorithmic, such as generators and parsers. Many PRs are combined in an application called *pipelines*.

- Applications – A term allocated to all processes that run on corpus or documents.
- Data stores – The place where corpus documents and annotations are saved.

3.4.1 GATE Components

Among the numerous plugins provided by GATE that focus on different resources, we used the following resources, plugins, and components:

- **ANNIE** – This is an application that performs information extraction (IE) on unstructured documents. It contains a set of PRs such as :
    - **ANNIE English Tokeniser** – This application splits the text into simple tokens and space tokens and provides a more sensible combination of words. Figure 3.2 shows the application of an ANNIE English Tokeniser on a sample document. As you can see in this figure, all tokens and space tokens (represented by pink and yellow colours in the figure) have been found and highlighted by this program resource.

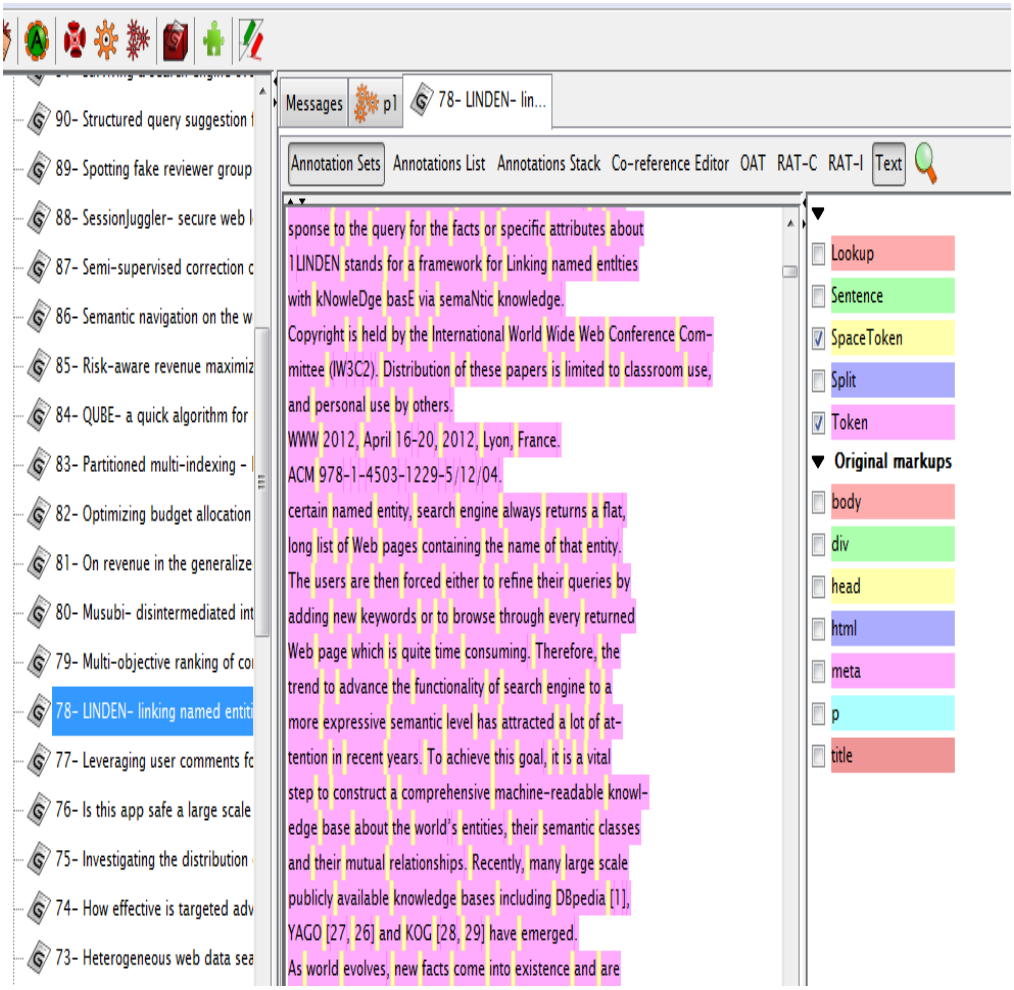

Figure 3-2 ANNIE English Tokeniser on a Sample Document.

- **ANNIE Sentence Splitter** – This PR splits a document into sentences. It finds sentences based on the tokens and sentence delimiters. . Figure 3.3 shows the application of an ANNIE Sentence Splitter on a sample document. As you can see in this figure, all sentences represented by the colour green in the

figure, have been found and highlighted by this program resource.

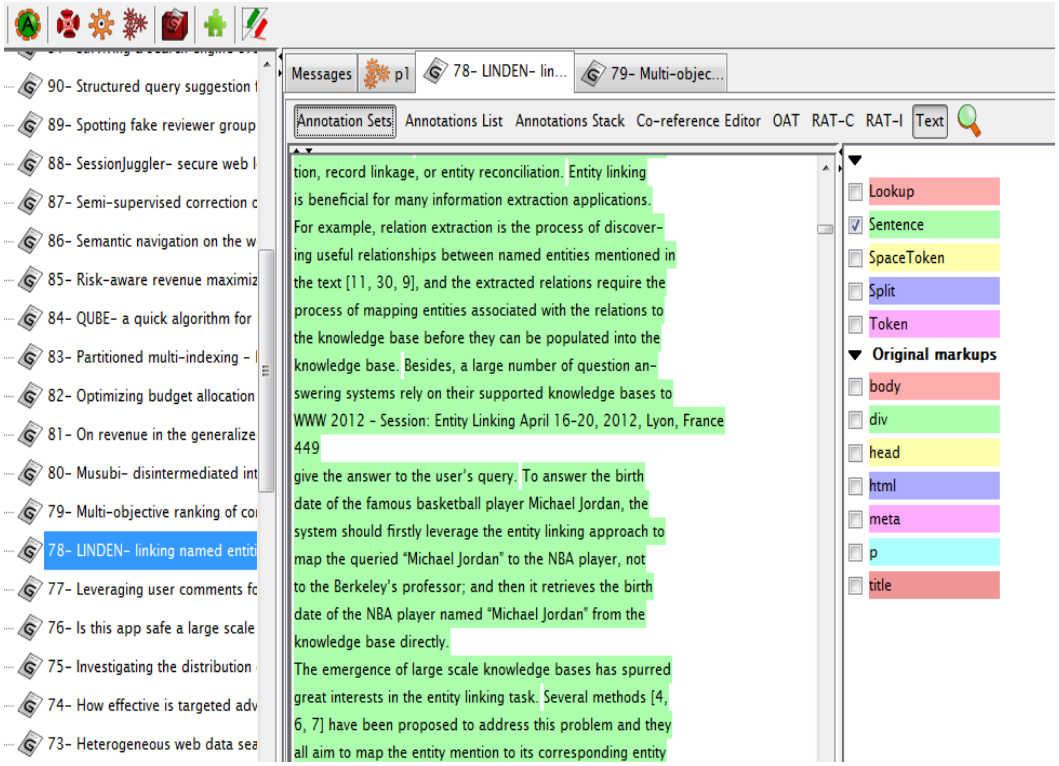

Figure 3-3 ANNIE Sentence Splitter on a Sample of Document.

- **ANNIE POS tagger** – **This** program resource is a Java implementation of *Brill's Transformation-based tagger*. It works with the Tokeniser and Sentence splitter to find and add part-of-speech tags as annotation tokens and symbols.

- **Flexible Gazetter** – A Gazetteer has a set of lists that contain the names of entities such as locations, organizations, and persons. These lists are used to

find the instances of these entities in documents. A Gazetteer processing resource processes a document and finds any mention of these lists and produces annotations in a type of “Look-up.” GATE has a set of predefined Gazetteer processing resources but also provides the flexible Gazetteer, a processing resource that allows users to specify their own lists of entities and search corpus on this basis. We used Flexible Gazetteer to specify the domain ontologies as the input list for document processing.

- **OntoRoot Gazetter** – The OntoRoot Gazetteer is a dynamically created Gazetteer and is required for ontology-based annotations. The OntoRoot Gazetteer is created by means of the OntoRoot Application (ORA) that processes the domain ontology and creates a set of entity lists from the ontology. Figure 3.4, which is taken from the GATE user guide, shows the main components of ORA. ORA works on a list of all concept names, individual names, and properties included in ontology. It first tokenizes every single linguistic term in the domain ontology and then allocates part-of-speech and lemma information to each token. Consequently, each individual token will possess an additional feature known as "root." The root of each token is comprised of the lemma as generated by the morphological analyzer. Subsequently, the lemma or set of lemmas are inserted into the dynamic gazetteer list.

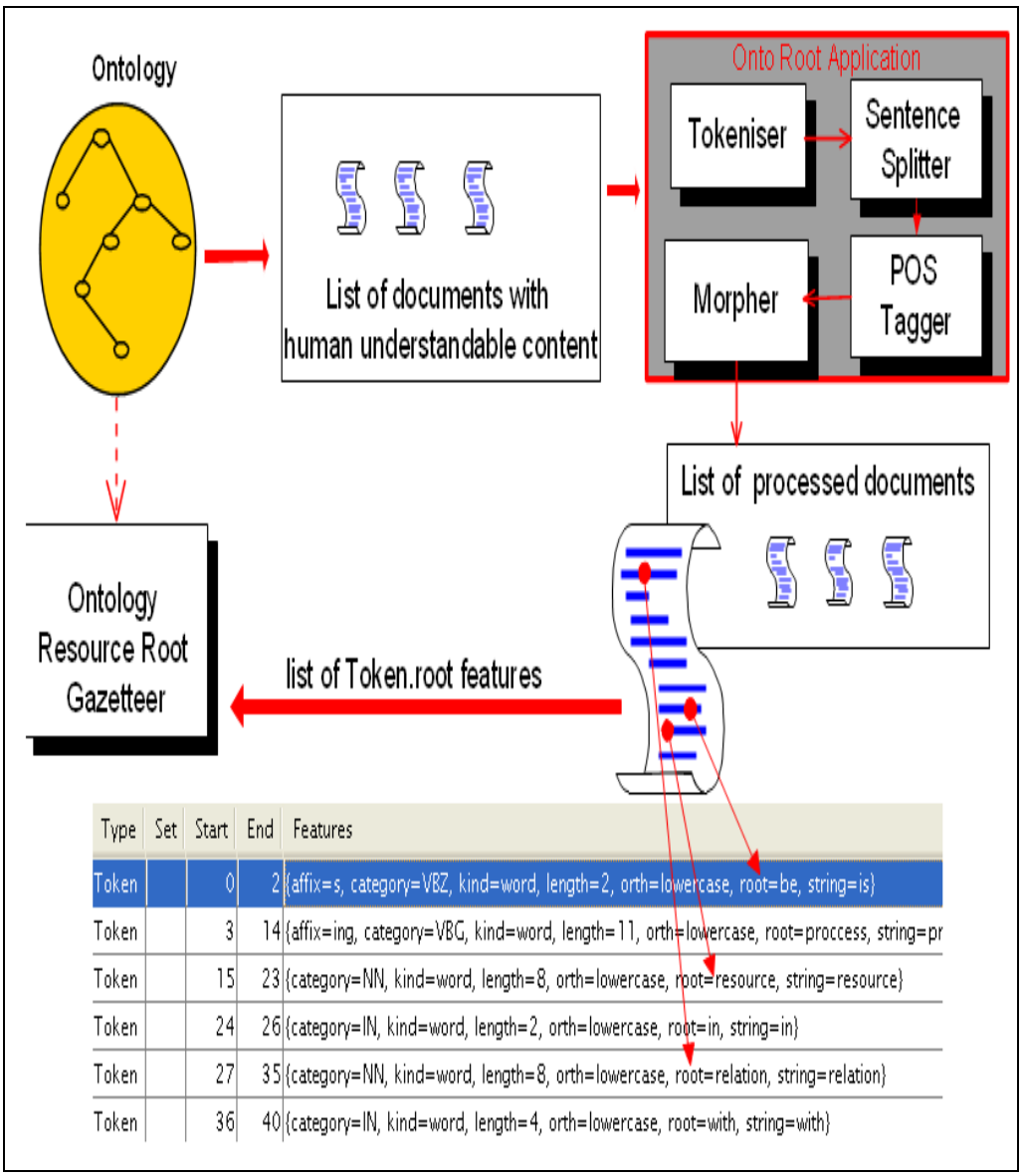

Figure 3-4 Building Ontology Resource Root (OntoRoot) Gazetteer from the Ontology.

- **GATE Morphological Analyser** – This processing resource identifies features like "lemma" and "affix" on Tokens. For instance, if the word under consideration is "singing," it can be used as a noun as well as a verb. In the

case where it is identified as a verb, its lemma will be "sing" and the affix is "ing."

In illustrating the various components of GATE, it is essential to understand that natural language understanding is needed to intelligently comprehend the information processed by computers. It should also be pointed out that the great majority of data analyzed by computers, i.e. texts, is semi-structured. One of the functions of ontologies is to assist in the process of understanding and analyzing these large volumes of text, it can identify the concepts and their relationships and assist to provide the main keywords in the documents.

3.4.2 The Annotation Strategy

The process of document analysis in our work is performed in two phases: 1) Document Analysis phase and 2) Ontological Concept Identification phase. In the following section, we will describe these two phases in greater detail.

3.4.2.1 Document Analysis Phase

In this phase, we first use the ANNIE English Tokeniser to split the text in our documents into simple tokens such as words, space tokens, numbers, punctuation, and to distinguish between words in lowercase or uppercase or mixed caps in an effort to maximize efficiency of text analysis. Second, we use the ANNIE Sentence splitter for segmenting the text into sentences. In the next phase, we use the ANNIE POS Tagger for recognizing each token as a part of speech tag, such as noun, verb, and adjective, among others. Then, we utilize the GATE Morphological analyser in

order to recognize the lemma and affix by taking each processed documents based on previously explained processing resources. The domain-specific documents that have been analyzed by these processes are used in the Ontological Concept Identification phase.

3.4.2.2 Ontological Concept Identification

In this section, we first build our gazetteer lists from the domain-specific ontology. For this purpose, we use Flexible Gazetteer and OntoRoot Gazetteer resources from GATE. As previously mentioned in Section 3.4.1, we use the Morphological analyzer to facilitate the process of extracting the list of concepts from the ontology. Afterwards, we perform a direct Gazetteer Lookup against a list of relevant terms found in the documents that have been analyzed in the Document Analysis Phase.

Since we have applied morphological analysis on both documents and ontology, for the purpose of Gazeteer lookup it is not necessary that the text in documents be in precisely the same form as what would appear in the Gazetteer list. In light of the fact that there are considerable morphological variations in the English language and a number of other languages, the necessary flexibility required may not be provided by a precise matching. The Exploiting Flexible Gazetteer and Morphological analysis on the processed ontologies and document enable us to consider lemmas of tokens and concepts of ontology in the process of matching instead of precise matching.

3.5 LOD-Based Annotation Component

LOD, an open structured way for publishing data, has emerged as a centrally important notion in the semantic web community. One of the most popular LOD companies that analyze documents and texts is AlchemyAPI. In the remaining part of this chapter, we will provide a detail description of what LOD components are capable of accomplishing.

3.5.1 Alchemy

AlchemyAPI employs different parsing methods in an effort to find the main concepts, keywords, and named entities in texts and links each concept to an entity in Wikipedia. AlchemyAPI engages in an in-depth analysis of documents and texts and offers various functions for text analysis. We used the following functions in order to annotate (find the main concepts of a document) our documents:

- Concept Tags – Alchemy has developed and provided techniques for finding the main concepts of documents in a way that is believed to be similar to how humans would recognize these concepts. The concept tagging API is capable of identifying the relatedness between concepts and the text in documents and finding concepts that are the main topics of a given text, even if these concepts are not explicitly mentioned in the text. For instance, in a text that mentions the words OWL and ontology on a number of occasions, the concept tagging API may find “semantic web” as a concept of the document even if it is not explicitly stated in the text.

- Field Terminology – This API employs complex statistics and natural language processing technologies to automatically classify a text. This classification is based on terminologies and content in a given text and their classification into main content categories or fields like computer-internet, business, semantic web, and networks among other terminologies.

3.5.2 The Annotation Strategy

The process of document analysis in our research is applied by using the concept tagging API and the field terminology API from Alchemy. In the first step, we used the concept tagging API to find all central relevant concepts in documents. This API finds the main concepts and also provides links to Wikipedia, DBpidia, and Yago entities with regards to each concept. The next step is to find more concepts related to the text using the field terminology API. This step will help us get more accurate information about the documents field.

3.6 Semantic Similarity Measuring Component

For the sake of our recommender system, we need to use a component to find the similarity of what is expressed in a query and what is mentioned in documents. One of the most prevalent means of finding the similarities between the two text segments is the employment of a simple lexical matching method and producing a similarity score based on the number of lexical units that occur in both input segments [57]. This method may cause a synonymy problem, where the terms used in queries are different from the words used in a document for describing the same or

similar meanings. For example, a document that talks about President Obama is related to the query "President of the United States," even though different words have been used.

There have been suggestions made to improve this simple method including stemming, stop-word removal, part-of-speech tagging, and longest subsequence matching, together with various incrementing and normalization factors [58]. Despite relative success, the application of such similarity methods has not always accomplished its ultimate aim of identifying the semantic similarity of texts. For instance, President of the United States example, none of these improvements can find a similarity between the two phrases "Barack Obama" and "President of the United States."

Contrary to the existing variations of syntactic similarity scores, the semantic similarity measuring component of our work finds the distance between the keywords or annotation of documents and any given query based on semantic analysis. In order to find the semantic similarity, we applied a method proposed in [61] to employ the knowledge expressed in Wikipedia in order to find the semantic distance between words. Wikipedia has turned into a frequently used source of semantic information along with WordNet, gazetteer and NER (Named Entity Recognizr) systems [56]. The method we have employed for semantic similarity finds the distance between any given two Wikipedia entities based on the number of common pages that have links to these entities.

3.7 Recommender Engine I

Recommender Engine I utilizes the domain-specific ontology-based annotation component and the semantic similarity measuring component and ranks documents based on similarity scores in order to accommodate the user.

3.7.1 Algorithm

The process used to recommend academic papers to the user is comprised of the following steps:

1) Prepare the repository of domain-specific documents that contains academic Computer Science papers.
2) Prepare a Computer Science Ontology, which contains information about subjects in the field of Computer Science.
3) Use GATE to annotate a domain-specific document based on the domain specific ontology.
4)) Extract and manage the extracted annotations for each document.
5) Find and calculate the semantic similarity between the keywords or annotations of the documents and any given query.
6) Rank documents based on the similarity score tailored to each individual user's interest.

3.7.2 Explanation and Example

This recommendation algorithm will be explained in greater depth via a sample in this section. As an example, assume we have a repository R1, including all

domain-specific documents and an ontology O1, which is a domain-specific ontology (Step 1 and Step 2 of the algorithm). The third step in the algorithm is to use GATE components for analyzing all documents in R1 and producing annotations. We use the components that have been previously described in Section 3.4.1 to analyze all documents in R1 and then export all annotations in the Annotation Repository, AR (Steps 3 and 4 in the algorithm). In step 4, we only export the annotations that are more frequent than a specific threshold in order to find the main topic of a document.

For the purpose of our example, assume that AR has the following records:

Doc 1: Social Networks, Networks, Associated Value, Book, and Facebook

Doc 2: Semantic, Annotation, and Logic

Doc3: Design, Publishing, Evaluation, and Engineering

Assume that we have a query, "Description Logic." Step 5 in this algorithm is to find the similarity between "Description Logic" and all annotations in all documents. What we have obtained from the semantic similarity component for this particular example is as follows:

Doc1	Similarity to "Description Logic"
Social Networks	0.294
Networks	0.254
Association rule	0.0
Facebook	0.0
Book	0.0

Doc2	Similarity to "Description Logic"
Semantic	0.811
Annotation	0.0
Logic	0.749

Doc3	Similarity to "Description Logic"
Design	0.286
Publishing	0.0
Evaluation	0.0
Engineering	0.227

The last step is to rank the documents based on these similarity measures. We used the following formula for calculating the final similarity measure between a document and a query term:

$$FSM(Doc_i, q) = \sum_{j=0}^{|ant^k(Doc_i)|} SEM - SIM(ant^k(Doc_i)[j], q)$$

Where $ant^k(Doc_i)$ is the list of all top-k annotations for Doci. In this formula, FSM is the Final Semantic Measure for a document (Doci) and a query term (q).). FSM is equal to the sum of the semantic similarity (SEM-SIM) for those annotations that have been found in the document more than a threshold k and the query term q.

For example, in our sample data store, top-k (Doc1, “Description Logic”) is 0.294, 0.254, 0.0; top-k (Doc2, “Description Logic”) is 0.811, 0.0. 0.749; and top-k (Doc3, “Description Logic” is 0.286, 0.0, 0.227, when we choose k as equal to 3. In this example, FSM (Doc1, “Description Logic”) is 0.548, FSM (Doc 2, “Description Logic”) is 1.560, and FSM (Doc 3, “Description Logic”) is 0.513.

It is evident in the provided example that Doc 1, Doc 2, and Doc 3 are ranked in the following order for the query term “Description Logic :”

1. Doc2.
2. Doc1.
3. Doc3.

3.7.3 Discussion

One of the main problems that the users of ordinary recommender systems face throughout their search task is being bombarded with a set of irrelevant topics for each query. The ontology-based recommender system, Recommender Engine I, can assist in alleviating this problem and improving the recommendation results by reducing the ratio of non-related suggestions for queries. We will further explain this notion through an example in this section. As an example, assume we want to generate recommendations for the Computer Science domain and have an ontology describing the main concepts of this domain. In addition, we have a repository R1 that has a set of documents including D1 and D2. D1 is a document about Java Island, an island in Indonesia, with a great deal of information about the islands geography, population, and tourism. On the other hand, D2 is a document about the Java Programming Language that includes different Computer Science concepts such as programming languages, the Java compiler, and the Object Oriented notion.

Consistent with recommender Engine I, both documents are analyzed using the GATE components and the Computer Science ontology, and their annotations are found and saved in a specific repository (Step 4 in Algorithm in Section 3.7.1). Now assume a query about Java Programming Language is received by Recommender Engine I. This recommender finds the semantic similarity between the Java Programming Language and the Java (The only concept found in D1 using GATE and the domain-specific ontology). This recommender Engine finds the similarity between the Java Programming language and Java, complier and Object Oriented

Programming for D2. Understandably, FSM for D2 is much greater than D1, which results in a document related to domain and the query is recommended to users.

3.8 Recommender Engine II

Recommender Engine II uses the LOD-based annotation component and the semantic similarity measuring component and ranks documents based on similarity scores to comply with the user's needs and interests.

3.8.1 Algorithm

The process of recommending academic papers to users is comprised of the following steps:

1) Prepare the repository of domain-specific documents that contains academic Computer Science papers.
2) Use AlchemyAPI to annotate our domain-specific documents.
3) Extract and manage the extracted annotations for each document.
4) Find and calculate the semantic similarity between the keywords or annotation of documents and any given query.
5) Rank documents based on the similarity score.

3.8.2 Explanation and Example

We will explain this recommendation algorithm through an example in this section. As an example, imagine we have a repository R1, including all domain-

specific documents (step 1). Use AlchemyAPI in the second step in the algorithm to analyze all documents in R1 and produce annotations. We use the component that we described in Section 3.5.1 to analyze all documents in R1 and then export all annotations in the Annotation Repository, AR (step 2 and step 3). For this example assume that AR has the following records:

Doc1: Data Structure, Artificial Intelligence, Statistical Classification, and Matrix

Doc2: Keyword Search, Search Engine, and Relational Database

Doc3: Semantic Web, Probability Theory, Machine Learning, and Artificial-Intelligence

Assume that we have a query, "Description Logic." Step 4 in this algorithm is to find the similarity between "Description Logic" and all annotations in all documents. What we have obtained from the semantic similarity component for this example is as follows:

Doc1	Similarity to "Description Logic"
Data Structures	0.474
Artificial Intelligence	0.533
Statistical Classification	0.466
Matrix	0.341

Doc 2	Similarity to "Description Logic "
Information Retrieval	0.486
Search Engine	0.435
Relational Database	0.549

Doc3	Similarity to "Description Logic"
Semantic Web	0.776
Probability Theory	0.537
Machine Learning	0.549
Artificial Intelligence	0.533

The last step is ranking documents based on their similarity measures SM score. The following formulas were used to calculate the final similarity measure between a document and a query term:

$$FSM\left(Doc_i, q\right) = \sum_{j=0}^{\left|key^k\left(Doc_i\right)\right|} SEM - SIM\left(key\left(Doc_i\right)\left[j\right], q\right)$$

Where $key^k\left(Doc_i\right)$ is the list of all keywords for Doci found by Alchemy API. In this formula, FSM is the Final Semantic Measure for a document (Doci) and a query term (q). FSM is equal to the sum values for the semantic similarity measure between all keywords recognized by Alchemy API of Doci and q.

For instance, in our sample data store, SM (Doc1, “Description Logic”) is 0.474, 0.533, 0.466; SM (Doc2, “Description Logic”) is 0.486, 0.435, 0.549; and SM (Doc3, “Description Logic”) is 0.776, 0.537, 0.549, and 0.533. In this example, FSM (Doc1, “Description Logic”) is 1.815, FSM (Doc2, “Description Logic”) is 1.470, and FSM (Doc3, “Description Logic”) is 2.396.

In this example we can find that Doc1, Doc2, and Doc3 are ranked in the following sequence as a result for the query term “Description Logic.”

1. Doc3.
2. Doc1.
3. Doc2.

3.8.3 Discussion

In this section we have demonstrated the advantages of the second approach using the same example that we used in section 3.7.3. Recommender Engine II produces annotations for both D1 and D2 using Alchemy APIs and exports them into a specific repository. In this case, D1 includes the main document concepts such as Tourism, Geography, Island, and Indonesia; D2 incorporates concepts such as Programming Languages, Compilers, and OO. It is apparent that D2 will have a larger final ranking score and will be recommended to users utilizing this recommender engine.

3.9 Recommender Engine III

Recommender Engine III is built on a combination of annotations provided by the domain-specific ontology-based annotation component and the LOD-based annotation component, along with the semantic similarity measuring component. Once these components are combined, the documents are ranked based on a final ranking score to express user interest.

3.9.1 Algorithm

The processes used to recommend academic papers to the user are comprised of the following steps:

1) Prepare the repository of domain-specific documents that contains academic Computer Science papers.

2) Prepare a Computer Science Ontology, which contains information about subjects in Computer Science.
3) Use GATE for annotating a domain-specific document based on our ontology.
4) Use AlchemyAPI to annotate our domain-specific documents.
5) Extract and manage the extracted annotations for each document.
6) Find and calculate the semantic similarity between the keywords or annotations of documents and any given query.
7) Rank documents based on a new similarity score to provide papers interest to the user.

3.9.2 Explanation and Example

In this section, we will explain this recommendation algorithm through an example. As an example, assume we have a repository R1, including all domain-specific documents, and ontology O1, a domain-specific ontology (Step 1 and Step 2 of the algorithm). The third step in the algorithm is comprised of using GATE components for analyzing all documents in R1 to produce annotations. We use all the components that have been previously described in Section 3.4.1 to analyze all documents in R1 and then export all annotations in the Annotation Repository, AR (Step 3 and 5 in the algorithm). Step 5 in the algorithm is to use AlchemyAPI for analyzing all documents in R1 and producing annotations. We use the component that was described in Section 3.5.1 to analyze all documents in R1 and then export all annotations in the Annotation Repository, AR (step 4 and step 5). For step 7, we use the semantic similarity scores between all annotations recognized by GATE and

alchemy API for the purpose of recommendation .In this example, assume that AR has the following records:

Doc1: Data Structures, Artificial Intelligence, Statistical Classification, Matrix, Social Networks, Networks, and Association Rules.

Doc2: Information Retrieval, Search Engine, Relational Database, Semantic, Annotation, and Logic.

Doc3: Semantic Web, Probability Theory, Machine Learning, and Artificial-Intelligence, Design, Publishing, Evaluation, and Engineering.

Assume that we have a query, "Description Logic." Step 5 in this algorithm is to find the similarity between "Description Logic" and all annotations in all documents. The following outlines what we have attained from the semantic similarity component for this example is as follows:

Doc1	Similarity to "Description Logic"
Data structures	0.474
Artificial Intelligence	0.533
Statistical classification	0.466
Matrix	0.341
Social Networks	0.291
Networks	0.254
Association rules	0.0

Doc2	Similarity to "Description Logic"
Artificial Intelligence	0.486
Search Engine	0.435
Relational database	0.549
Semantic	0.811
Annotation	0.0
Logic	0.749

Doc3	Similarity to "Description Logic"
Semantic Web	0.776
Probability theory	0.537
Machine learning	0.549
Artificial Intelligence	0.533
Design	0.286
Publishing	0.0
Evaluation	0.0
Engineering	0.227

The final step is to rank documents based on these similarity measures. We used the following formula to calculate the final similarity measure between a document and a query term:

$$FSM\left(Doc_i, q\right) = \sum_{j=0}^{\left|ant^k\left(Doc_i\right)\right|} SEM - SIM\left(ant^k\left(Doc_i\right)[j], q\right) + \sum_{j=0}^{\left|key^k\left(Doc_i\right)\right|} SEM - SIM\left(key\left(Doc_i\right)[j], q\right)$$

In this formula, FSM is the Final Semantic Measure for a document (Doci) and a query term (q). FSM is equal to the sum values for the semantic similarity measure between all annotations of Doci and q.

For example, in our sample data store, SM (Doc1, “Description Logic”) is 0.474, 0.533, 0.466, 0.341, 0.294, and 0.254. 0.0; SM (Doc2, “Description Logic”) is 0.486, 0.435, 0.549, 0.811, 0.0., and 0.749; and SM (Doc3, “Description Logic”) is 0.776, 0.537, 0.549, 0.533, 0.286, 0.0, and 0.227. In this example, FSM (Doc1, “Description Logic”) is 2.364, FSM (Doc2, “Description Logic”) is 3.031, and FSM (Doc3, “Description Logic”) is 2.909. In this example we have illustrated that Doc1, Doc2, and Doc3 are ranked as follows for the query term “Description Logic.”

1. Doc 2.
2. Doc3.
3. Doc1.

3.10 Differences Between the RE's Outcomes

3.10.1 Discussion and Example

There are some situations where GATE is not working well. These cases usually happen when the main keywords in R1 are not in our ontology, so the AR will not be enough to rank the documents correctly. By using Alchemy API, we aim at overcoming the GATE weakness in the cases when the ontology does not cover all keywords in the domain repository. Alchemy can recognize the main keywords and

themes of a document without employing a domain ontology. Nonetheless, this does not mean that using Alchemy is always the best option and that it does have some problem. This problem could happen when Alchemy produces a set of unrelated or wrong keywords or when it fails to find main domain key phrases in documents.

The situation of using GATE and LOD alone is sometimes not as effective as when using the combination of them. When we have combined the annotations provided by the domain-specific ontology-based annotation component and LOD-based annotation component, we will have a rich annotation repository especially when the keywords in our ontology are good and the keywords from Alchemy also are related with a given query, which can help to produce accurate and related documents with a given query. The performance of this approach is highly dependent on the performance of the annotations produced by each of GATE or LOD-based annotation systems.

As an example, consider we have R1 including Doc1 and Doc2. Doc2 is more related to the "description logic" term than Doc1. Figure 3.5 and 3.6 shows the contents of Doc1 and Doc2:

Simulate the Closed World Assumption in a concrete class are:

Amino Acid Kinetic --> Simulation Algorithm Ontology

BioAssay Ontology --> Lipid Ontology

BioTop --> NanoParticle Ontology

Bleeding History Phenotype --> Neomark Oral Cancer-Centred Ontology

Cancer Research and Management ACGT Master Ontology --> Ontology for Genetic Interval.

During encoding, to gather additional information about a pattern, we noted the OWL 2 description logic constructs each utilizes.

Figure 3-5 Doc1 Contents.

Description logic (DL) is a family of formal knowledge representation languages. It is more expressive than propositional logic but has more efficient decision problems than first-order predicate logic.

DL is used in artificial intelligence for formal reasoning on the concepts of an application domain (known as terminological knowledge).

Figure 3-6 Doc2 Contents.

The AR by GATE for both documents has the following records:

Doc1: Pattern, Description Logic, and Logic.

Doc2: Description logic, Logic, and Artificial Intelligence.

Assume that we have a query, "Description Logic." What we have obtained from the semantic similarity component for this AR is as follow:

Doc1: Pattern = 0.0, Description logic = 1, and Logic = 0.74

Doc2: Description logic = 1, Logic = 0.74, and Artificial Intelligence = 0.53

The FSM for both documents and query term are:

FSM (Doc1, "Description Logic") = 1.74

FSM (Doc2, "Description Logic") = 2.27

In this example we can find that Doc1 and Doc2 are ranked in the following sequence as a result for the query term “Description Logic:”

1- Doc2

2- Doc1

The AR by AlchmeyAPI for both documents has the following records:

Doc1: Web Ontology Language, Description Logic, Ammonia, Gene, Amino Acid, Semantic Web, and Acid.

Doc2: Scientific Method, Formal Language, Propositional Calculus, Cognition, Logic, First-Order Logic, Predicate Logic, and Reason.

Assume that we have a query, “Description Logic.” What we have obtained from the semantic similarity component for this AR is as follow:

Doc1: Web Ontology Language = 0.78, Description logic = 1, Ammonia = 0.16, Gene = 0.0, Amino Acid = 0.16, Semantic Web = 0.77, and Acid = 0.16

Doc2: Scientific Method = 0.37, Formal Language = 0.58, Predicate Logic = 0.61, Logic = 0.74, Propositional Calculus = 0.69, First-Order Logic = 0.69, Cognition = 0.43, and Reason = 0.58

The FSM for both documents and query term are:

FSM (Doc1, "Description Logic") = 3.03

FSM (Doc2, "Description Logic") = 4.69

In this example we can find that Doc1 and Doc2 are ranked in the following sequence as a result for the query term “Description Logic”

1- Doc2

2- Doc1

The AR by combination of GATE and LOD for both documents has the following records:

Doc1: Pattern, Description Logic, Logic, Web Ontology Language, Description Logic, Ammonia, Gene, Amino Acid, Semantic Web, and Acid.

Doc2: Description logic, Logic, Artificial Intelligence, Scientific Method, Formal Language, Propositional Calculus, Cognition, Logic, First-Order Logic, Reason, and Predicate Logic.

Assume that we have a query, “Description Logic,” and we found the SM for all AR in the same way that we did it in the last section. The FSM for both documents will be as follow

FSM (Doc1, "Description Logic") = 1.74+3.03 = 4.77

FSM (Doc2, "Description Logic") = 2.27+4.69 = 6.96

3.10.2 Justifications

As mentioned earlier, GATE and AlchemyAPI result can become in different cases. For our example in Section 3.10.1, we recognized that GATE results are good and related with a query term because our ontology covers all main concepts in the R.

However, AlchemyAPI found many unrelated concepts, which did not affect our result for these examples because their weights are small. AlchemyAPI in this example works well and gives good ranking. The combination of GATE and LOD is also acceptable because the results and the ranking of the documents for each one are the same so the results of their combination is acceptable.

Chapter 4:

4 Implementation

In this chapter, we describe the implementation of our approach for designing the semantic-based recommender system. We also describe the implementations of all system components in the subsequent sections.

4.1 The Repository of Domain Specific Documents

We have created a repository of domain-specific documents in order to be used in all of the information retrieval tasks. Our domain-specific documents were collected from two sources: 1) papers that have been published in the Semantic Web - ISWC 2011 - 10th International Semantic Web Conference, Bonn, Germany, and 2) papers that have been published in the 21st International Conference on World Wide Web, WWW (2013): Lyon, France**.** We have randomly chosen 100 Computer Science papers and saved them in the repository.

4.2 Domain-Specific Ontology

We have created an ontology to represent the main subjects of the Computer Science domain and the relationships between them. In order to create the ontology, we used the information provided by the ACM Computing Classification System (CCS). Figure 4.1 shows the main components in CCS:

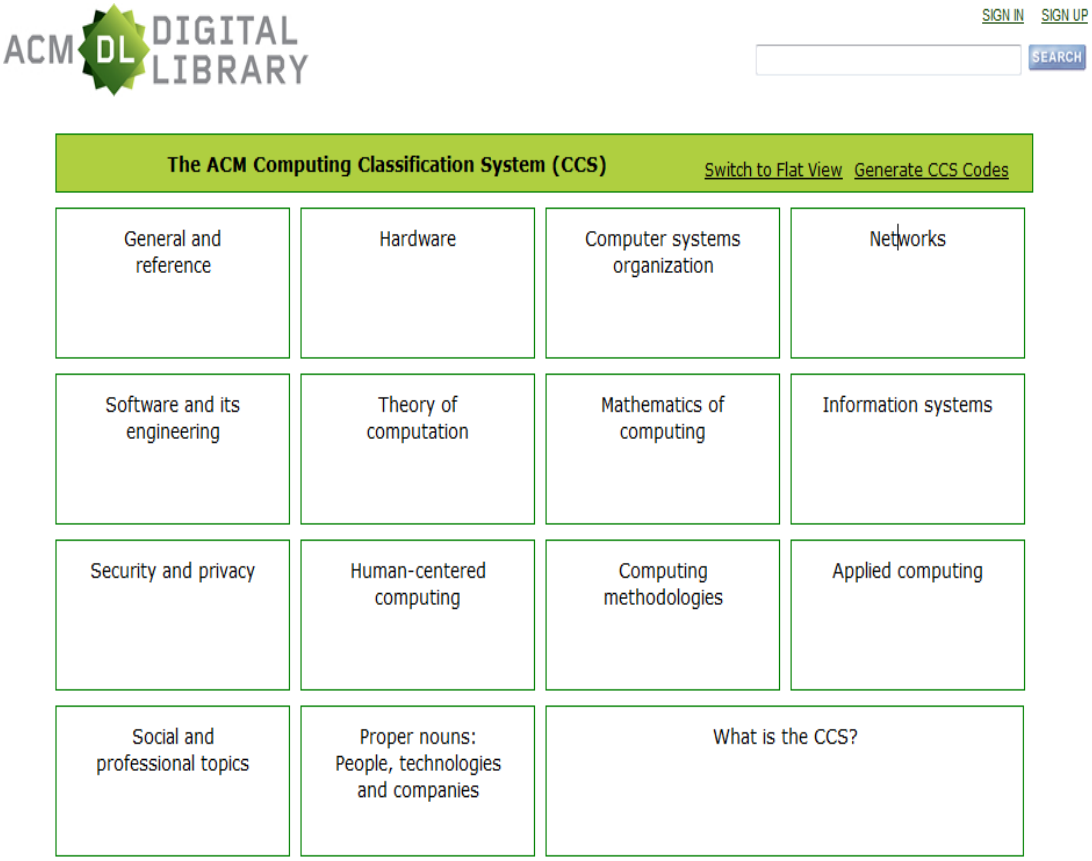

Figure 4-1 The ACM Computing Classification System (CCS).

We created the domain-specific ontology by using the Protégé Software. The reason for choosing the Protégé Software is that it is one of the best freely available open source software that can be used without any complexity for the editing and development of ontologies. Figure 4.2 depicts the main classes of our ontology in Protégé.

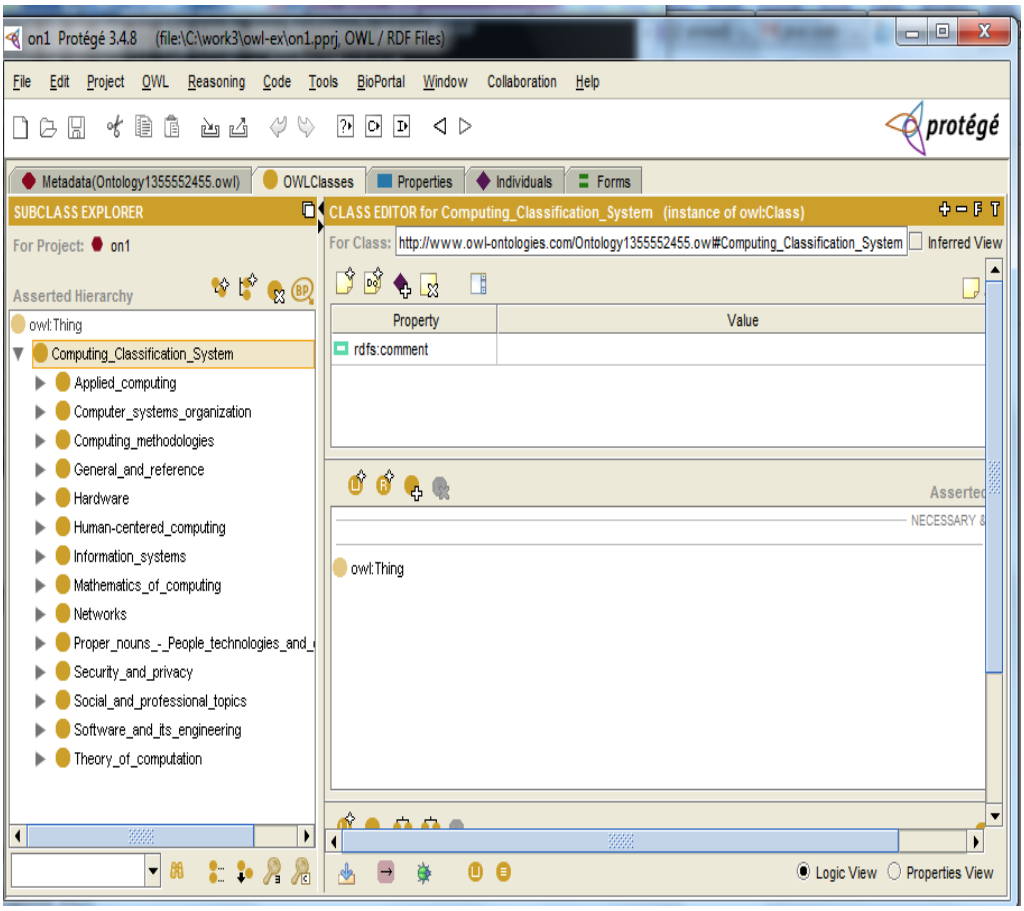

Figure 4-2 Domain-Specific Ontology Protégé.

4.3 Domain-Specific Ontology-Based Annotation Component

We have annotated our domain specific documents in the repository using the domain-specific ontology that we created using Protégé. We used all components of GATE described previously in Section 3.4.1. The following steps explain this process in greater detail:

4.3.1 Loading the CREOLE Plugin

In Gate, all processing resources (PRs) are in the CREOLE plugin. Thus, in order to load Gazetteer_Ontology_Based plugin, ANNIE, and Tools plugin, we have to first load the CREOLE plugin that contains these features. Figure 4.3 illustrates the loading of our PRs:

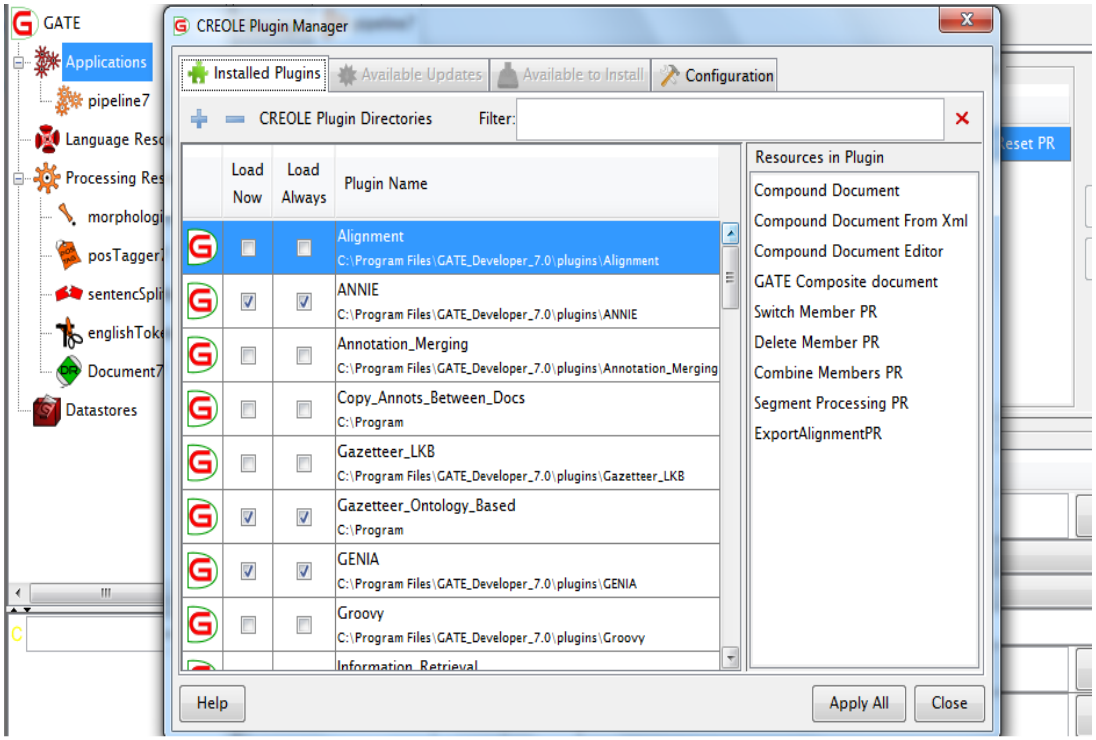

Figure 4-3 The CREOLE Plugin Manager.

4.3.2 Creating a new corpus

The next step is to create a new place called "corpus" and subsequently populate our domain-specific documents inside it. Figure 4.4 demonstrates this step more accurately:

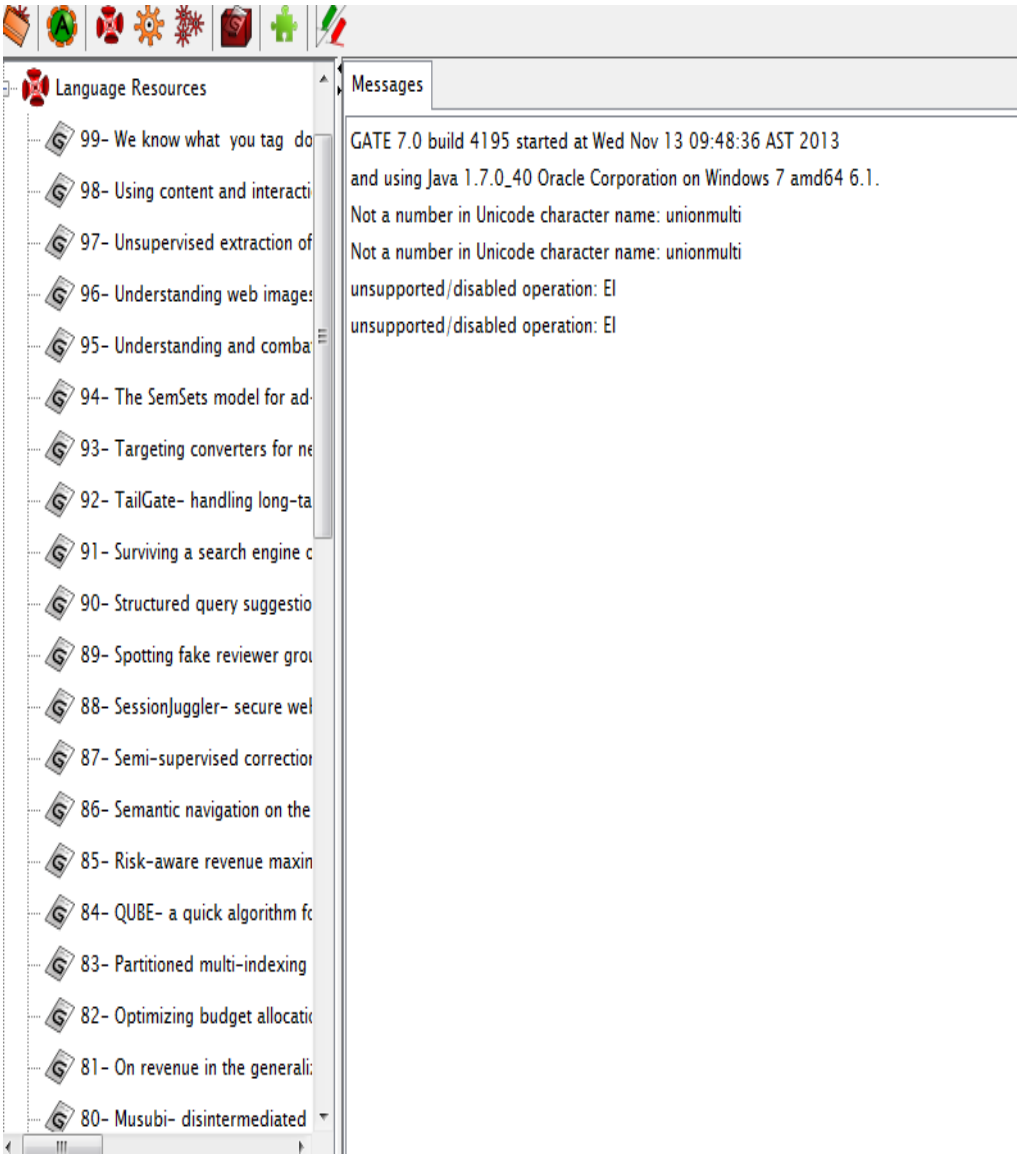

Figure 4-4 Creating Corpus.

4.3.3 Loading our Ontology:

Here we load the ontology test-ontology-instances.owl. Figure 4.5 shows our ontology in GATE.

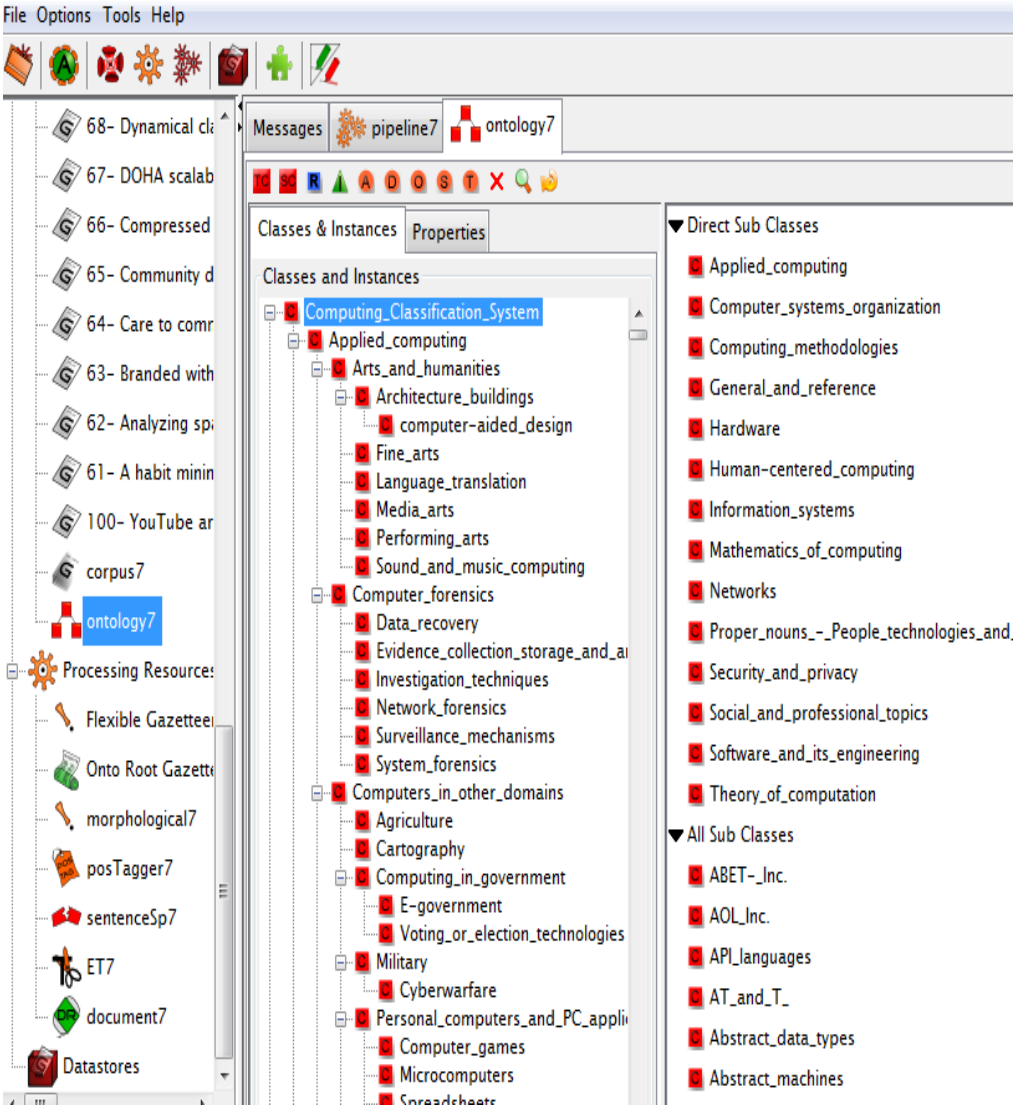

Figure 4-5 Loading Ontology in GATE.

4.3.4 Creating and Running an Application (Corpus Pipeline)

Once our ontology and all other resources have been loaded, as outlined in Section 3.4.1, we create an application to load the Gate tools (Tokeniser, Sentence Splitter, POS Tagger, Morphological Analyser, OntoRootGazetteer, and Flexible Gazetteer) over the corpus that we already created in the previous step. The input used in our pipeline Application consists of a series of documents that will be annotated depending on the domain ontology. Figure 4.6 displays a GATE application with PRs.

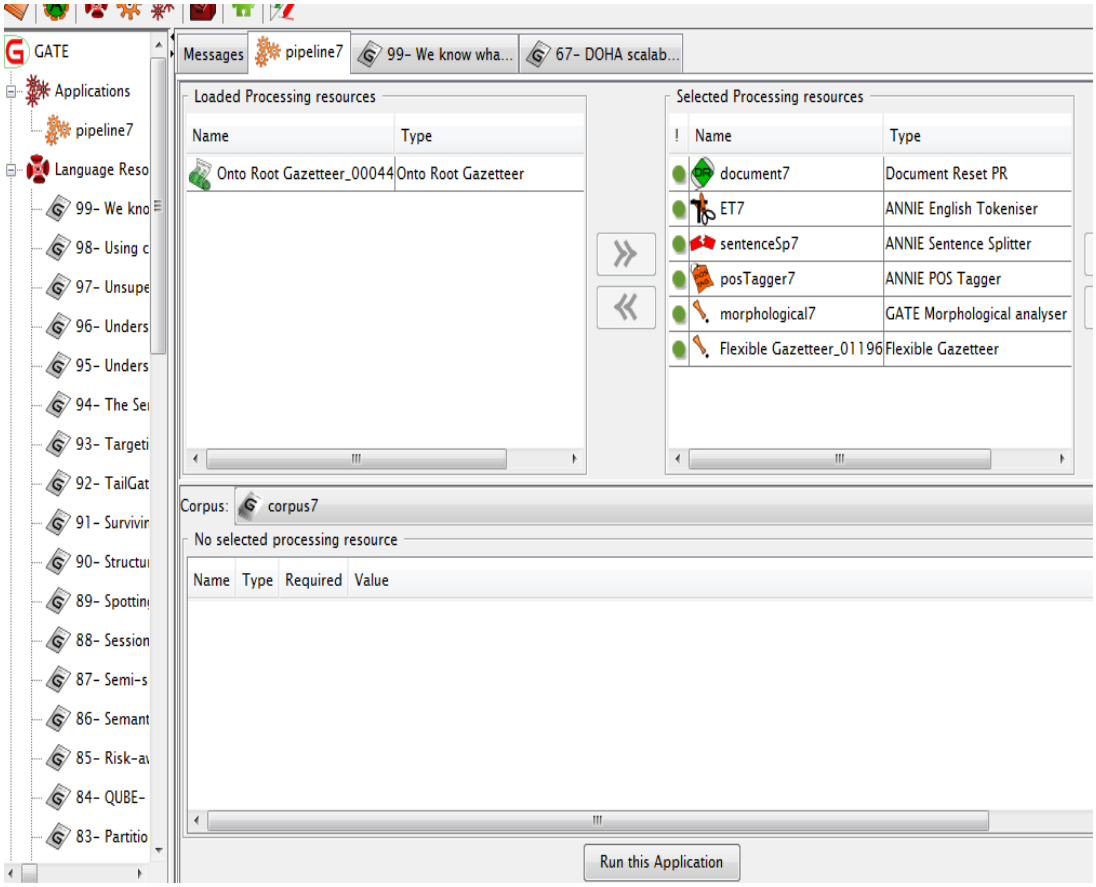

Figure 4-6 GATE Application.

The next step is to run the application in order to annotate the documents. Documents with annotations of type Lookup are considered as output. Every single one of these documents possesses the feature 'URI' which is used to identify the URI of the ontology resource. The documents also contain the feature 'type' for identifying the specific type of ontology resource, such as classes or individuals. Figure 4.7 shows a sample of annotating documents.

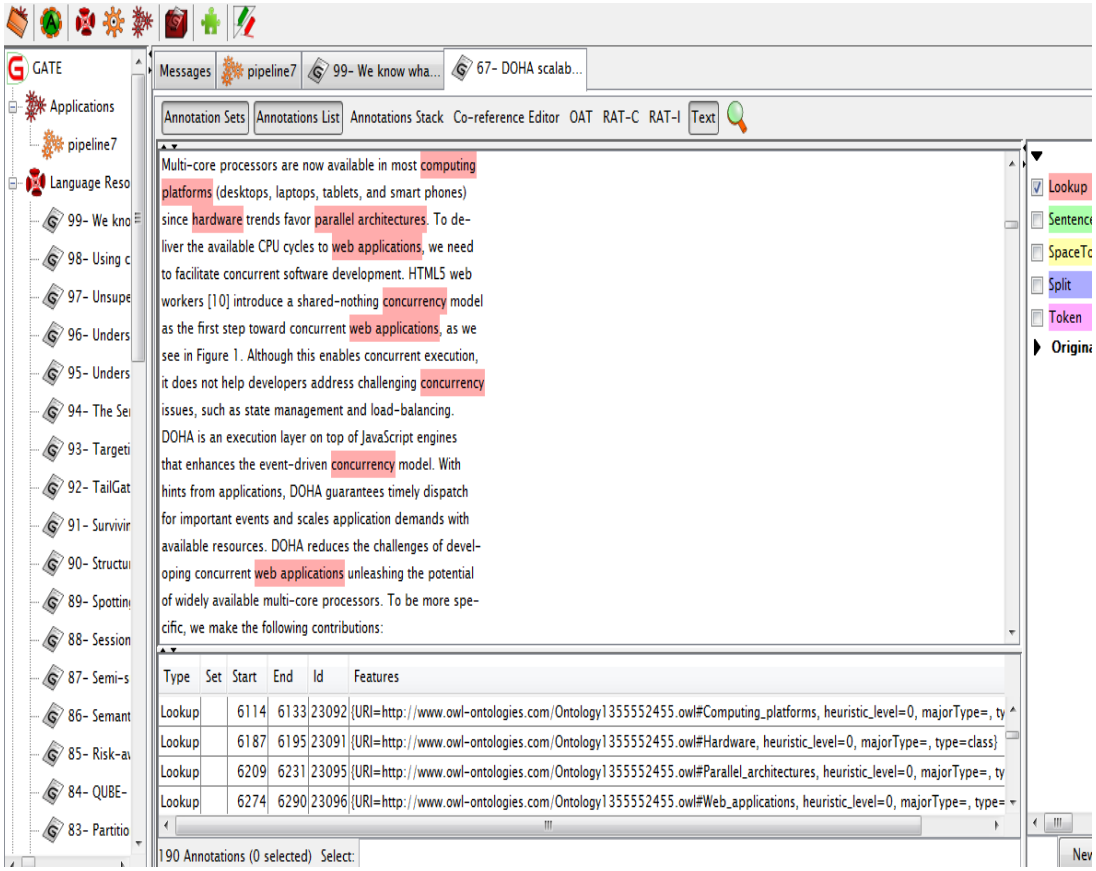

Figure 4-7 Output Example.

4.3.5 Export Annotation to Repository

Finally, all annotations will be saved in the text file in order to use it as an input data to the semantic similarity measuring component.

4.4 LOD-Based Annotation Component

AlchemyAPI is among the most popular natural language processing services used by different applications. It can extract semantic meta-data from content such as information on people, topics, facts, and authors. By utilizing Alchemy API, it is possible to have access to different content analysis services on internet-accessible web pages and posted HTML or text content. We used field terminology and concept tagging services and applied them to the documents in our repository.

4.5 Semantic Similarity Measuring Component

One of the most powerful tools used to analyze unstructured documents is TagMe!. We have used the RelatingAPI service from TagMe! to measure the semantic similarity between two topics. This measure produces a value between 0 and 1, which estimates the semantic relatedness between topics. The following figure shows a sample of the result of our java code used to measure semantic similarity between the concept semantic web and a set of different concepts.

SM(Semantic web, Ranking) = 0.0
SM(Semantic web, RDF) = 0.847
SM(Semantic web, World Wide Web) = 0.6
SM(Semantic web, Semantic web) = 1.0
SM(Semantic web, Resource Description Framwork) = 0.847
SM(Semantic web, Entity) = 0.466
SM(Semantic web, companies) = 0.0
SM(Semantic web, Information Retrival) = 0.639
SM(Semantic web, Search Engine) = 0.611
SM(Semantic web, Model) = 0.0

Figure 4-8 SM Sample.

4.6 Recommender Engine I (REI)

In REI, we have used the domain-specific ontology-based annotation component that we created by applying GATE to 100 academic papers. The semantic-similarity measuring component then ranks documents based on similarity scores. In order to find the main topics of a document, we have exported those annotations that are more frequent than threshold >=3 and found the FSM for the document and query by finding the summation of the top 5 largest values for SM. Figure 4.9 demonstrates results after running a sample for the "Ontology" query.

	A	B
1	A Distribution Semantics for Probabilistic Ontologies	5.8273999095
2	Reasoning under Uncertainty with Log-Linear Description Logics	5.5410999358
3	The SemSets model for ad-hoc semantic list search	5.3998999819
4	Ontology Design Pattern Language Expressivity Requirements	5.3410999477
5	Semantic Link Prediction through Probabilistic Description Logics	5.1331999004
6	Cross-Lingual Web API Classification and Annotation	4.6351999938
7	Learning Terminological Naive Bayesian Classifiers under Different Assumptions on Missing Knowledge	4.5253999531
8	SPARQL-DL Queries for Antipattern Detection	4.3254999518
9	CODI- Combinatorial Optimization for Data Integration results for OAEI 2011	4.3096999526
10	An Expert System on Linguistics to Support Natural Multilingual Collaborative Management of Interlingual Semantic Web Knowledge bases	4.2548999786
11	Care to comment recommendations for commenting on news stories	4.1330999881
12	Structured query suggestion for specialization and parallel movement- effect on search behaviors	4.0625999868
13	Semantic navigation on the web of data. specification of routes	3.9567999691
14	Classifying Users and Identifying User Interests in Folksonomies	3.7960999906
15	Surviving a search engine overload	3.7725000381
16	Dynamical classes of collective attention in twitter	3.6547999978
17	DOHA scalable real-time web applications through adaptive concurrent execution	3.4666999727
18	An Evidential Approach for Modeling and Reasoning on Uncertainty in Semantic Applications	3.3958999515
19	Musubi- disintermediated interactive social feeds for mobile devices	3.3056999967
20	Applications of Ontology Design Patterns in the Transformation of Multimedia Repositories	3.2547999844
21	KIM – a semantic platform for information extraction and retrieval	3.2430000007
22	HoxhaEtAl_COLD2011	3.2391999662
23	Linguistic Patterns for Information Extraction in OntoCmaps	3.2156999409
24	LINDEN- linking named entities with knowledge base via semantic knowledge	3.1841000114
25	Crowdsourcing tasks in Linked Data management	3.1450000107
26	Leveraging user comments for aesthetic aware image search reranking	3.1176999658
27	A Pattern For Interrelated Numerical Properties	3.0822999775
28	Incremental SPARQL Evaluation for Query Answering on Linked Data	3.0195999742
29	Understanding and combating link farming in the twitter social network	3.0115999877

Sheet1

Figure 4-9 Sample REI Results.

4.7 Recommender Engine II (REII)

We used LOD-based annotation-component and semantic similarity measuring component, and then ranked documents based on their similarity scores to provide the users with their desired documents. Figure 4.10 demonstrates the results of running a sample of "Ontology" query.

	A	B
1	An Evidential Approach for Modeling and Reasoning on Uncertainty in Semantic Applications	8.117499955
2	Finite Lattices Do Not Make Reasoning in ALCI Harder	7.3921000063
3	Reasoning under Uncertainty with Log-Linear Description Logics	7.2470000386
4	Heterogeneous web data search using relevance-based on the fly data integration	6.8901999593
5	Building A Fuzzy Knowledge Body for Integrating Domain Ontologies	6.6235000491
6	Interactively Mapping Data Sources into the Semantic Web	6.5999999642
7	An Expert System on Linguistics to Support Natural Multilingual Collaborative Management of Interlingual Semantic Web Knowledge bases	6.5961000025
8	The SemSets model for ad-hoc semantic list search	6.478399992
9	Identifying Information Needs by Modelling Collective Query Patterns	6.3921999931
10	Building Ontologies by using Re-engineering Patterns and R2RML Mappings	6.2822999954
11	Evolution of the COMA Match System	6.0667000711
12	LDOA Results for OAEI 2011	6.0587999821
13	Semantic navigation on the web of data. specification of routes	5.9686999917
14	Lily Results on SEALS Platform for OAEI 2011	5.9216000736
15	Development of an Ontological Model of Evidence for TRANSFoRm Utilizing Transition Project Data	5.8079000413
16	CODI- Combinatorial Optimization for Data Integration results for OAEI 2011	5.7882000208
17	SPARQL-DL Queries for Antipattern Detection	5.6078999341
18	Extracting core knowledge from Linked Data	5.5452000797
19	Incremental SPARQL Evaluation for Query Answering on Linked Data	5.521600008
20	Handling uncertainty in information extraction	5.4706000686
21	Defining and Executing Assessment Tests on Linked Data for Statistical Analysis	5.3805000484
22	Formalising Uncertainty- An Ontology of Reasoning	5.3805000484
23	Consuming Linked Data within a Large Educational Organization	5.2902000695
24	Factorizing YAGO scalable machine learning for linked data	5.141200006
25	Applications of Ontology Design Patterns in the Transformation of Multimedia Repositories	5.133400023
26	Pharmaceutical Validation of Medication Orders Using an OWL Ontology and Business Rules	5.1214999855
27	Leveraging user comments for aesthetic aware image search re-ranking	5.0311999619
28	Crowdsourcing tasks in Linked Data management	5.0000999868
29	A Semantic Model for Integrated Content Management	4.9843000472

Sheet1

Figure 4-10 Sample REII Results.

4.8 Recommender Engine III (REIII)

We used a combination of annotations provided by the domain-specific ontology-based annotation component and the LOD-based annotation component, along with the semantic similarity measuring component. The documents are ranked based on a final ranking score of the combination of REI and REII. Figure 4.11 shows the results of running a sample "ontology" query:

	A	B
1	Reasoning under Uncertainty with Log-Linear Description Logics	12.7880999744
2	The SemSets model for ad-hoc semantic list search	12.7488999888
3	An Evidential Approach for Modeling and Reasoning on Uncertainty in Semantic Applications	11.5133999065
4	An Expert System on Linguistics to Support Natural Multilingual Collaborative Management of Interlingual Semantic Web Knowledge bases	10.8509999812
5	A Distribution Semantics for Probabilistic Ontologies	10.6821999252
6	Finite Lattices Do Not Make Reasoning in ALCI Harder	10.2625999749
7	CODI- Combinatorial Optimization for Data Integration results for OAEI 2011	10.0978999734
8	SPARQL-DL Queries for Antipattern Detection	9.9333998859
9	Semantic navigation on the web of data. specification of routes	9.9254999608
10	Ontology Design Pattern Language Expressivity Requirements	9.674399972
11	Building A Fuzzy Knowledge Body for Integrating Domain Ontologies	9.5881000459
12	Building Ontologies by using Re-engineering Patterns and R2RML Mappings	9.2351999581
13	Interactively Mapping Data Sources into the Semantic Web	8.7175999582
14	Cross-Lingual Web API Classification and Annotation	8.4156000614
15	Incremental SPARQL Evaluation for Query Answering on Linked Data	8.3921999633
16	Applications of Ontology Design Patterns in the Transformation of Multimedia Repositories	8.3882000074
17	Evolution of the COMA Match System	8.3490000814
18	Learning Terminological Naive Bayesian Classifiers under Different Assumptions on Missing Knowledge	8.3253999352
19	Identifying Information Needs by Modelling Collective Query Patterns	8.211800009
20	Leveraging user comments for aesthetic aware image search reranking	8.1488999277
21	Crowdsourcing tasks in Linked Data management	8.1450999975
22	Factorizing YAGO scalable machine learning for linked data	8.0862999856
23	Heterogeneous web data search using relevance-based on the fly data integration	8.0704999715
24	Extracting core knowledge from Linked Data	8.0627000928
25	A Pattern For Interrelated Numerical Properties	8.0311999321
26	Semantic Link Prediction through Probabilistic Description Logics	7.8311999142
27	Defining and Executing Assessment Tests on Linked Data for Statistical Analysis	7.6943000257
28	HoxhaEtAl_COLD2011	7.6902999878
29	Modest Use of Ontology Design Patterns in a Repository of Biomedical Ontologies	7.6233999804

Sheet1

Figure 4-11 Sample REIII Results.

Chapter 5:

5 Evaluation

5.1 Introduction

In this chapter, we report on the evaluation of the performance and effectiveness of the REI, REII, and REIII techniques aimed at retrieving the maximum number of relevant papers for a given query. In an effort to calculate the efficiency of our approach, we employed both *precision* and *recall* measures in order to illustrate *specificity* and *coverage*. The main idea behind retrieval information system evaluation is interrelated to relevant or non-relevant documents regarding a particular query. There are various measures utilized for evaluating the performance of information retrieval systems. The two most important evaluation measures include precision and recall:

Precision is defined as the number of relevant documents retrieved by a search divided by the total number of documents retrieved by that same search.

$$\text{precision} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{retrieved documents}\}|}$$

Recall is defined as the number of relevant documents retrieved by a search divided by the total number of existing relevant documents.

$$\text{recall} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{relevant documents}\}|}$$

5.2 Evaluation Results

We used an online survey to evaluate the results we obtained thus far in the study. First we asked a series of questions in the form of a survey and requested that graduate computer science students and researchers fill out the survey. Seven questions were used in each query term. The query terms that we used included semantic web, social network, ontologies, annotation, and web services. In our questionnaires, we also asked the users to evaluate Google Scholar but the name of the search engines remained hidden at all times to ensure that evaluation conditions were fair and unbiased. In light of our decision to keep the names of our systems and Google Scholar hidden, we used System 1 instead of Google Scholar, System 2 in place of REII, System 3 rather than REI, and System 4 instead of REIII. The questions we used for each query are as follows ("query term" was replaced by the appropriate query term in each question):

Q1. The following is a list of the top ranked academic papers returned by system1 for the "query term" from the existing papers in our repository. Please rank the relatedness of each paper to the "query term."

Q2. The following are the first 10 academic papers that are returned by system2 for the "query term." Please rank the relatedness of each academic paper to the query term.

Q3. The following are the first 10 academic papers that are returned by system3 for the "query term." Please rank the relatedness of each academic paper to the query term.

Q4. The following are the first 10 academic papers that are returned by system4 for the "query term." Please rank the relatedness of each academic paper to the query term.

Q5. The following is the list of the top ranked papers that have been returned by system1 for the "query term," but is not returned by system2. Please select those papers that you think should be included in the first 10 papers and their rank.

Q6. The following is the list of the first top ranked papers that have been returned by system1 for the "query term," but is not returned by system3. Please select those papers that you think should be included in the first 10 papers and their rank.

Q7. The following is the list of the first top ranked papers that have been returned by system1 for the "query term," but is not returned by system4. Please select those papers that you think should be included in the first 10 papers and their rank.

Figure 5.1 demonstrates a sample of question 3 for system 3 on the survey for the query term “ontology:"

3* 3. The following is the the first 10 academic papers that is returned by system 3 for the query term " Ontology ". Please rank the relatedness of each academic paper to the query term.

	1 Not related	2	3	4	5	6	7	8	9	10 Strongly related
Paper no 1 A Distribution Semantics for Probabilistic Ontologies ftp://ceur-ws.org/pub/publications/CEUR-WS/Vol-778.zip#page=83	○	○	○	○	○	○	○	○	○	○
paper no 2 Reasoning under Uncertainty with Log-Linear Description Logics http://ceur-ws.org/Vol-778/proceedings.pdf#page=113	○	○	○	○	○	○	○	○	○	○
paper no 4 Ontology Design Pattern Language Expressivity Requirements http://ceur-ws.org/Vol-929/proceedings.pdf#page=30	○	○	○	○	○	○	○	○	○	○
paper no 5 Semantic Link Prediction through Probabilistic Description Logics ftp://ceur-ws.org/pub/publications/CEUR-WS/Vol-778.zip#page=95	○	○	○	○	○	○	○	○	○	○
paper no 6 Cross-Lingual Web API Classification and Annotation http://ceur-ws.org/Vol-775/775-complete.pdf#page=6	○	○	○	○	○	○	○	○	○	○
paper no 7 Learning Terminological Naive Bayesian Classifiers under Different Assumptions on Missing Knowledge http://ceur-ws.org/Vol-778/paper6.pdf	○	○	○	○	○	○	○	○	○	○
paper no 8 SPARQL-DL Queries for Antipattern Detection http://ontologydesignpatterns.org/wiki/images/5/56/WOP_paper_8.pdf	○	○	○	○	○	○	○	○	○	○
paper no 9 CODI- Combinatorial Optimization for Data Integration results for OAEI 2011 http://s3.amazonaws.com/academia.edu.documents/30594105/om2011_proceedings.pdf?AWSAccessKeyId=AKIAIR6FSIMDFXPEERSA&Expires=1382162748&Signature=r2ntlUY67XFKKly%2FGPfJEsEeFNs%3D&response-content-disposition=inline#page=144	○	○	○	○	○	○	○	○	○	○
paper no 10 An Expert System on Linguistics to Support Multilingual Management of Interlingual Semantic Web Knowledge bases http://www-sop.inria.fr/members/Maxime.Lefrancois/docs/LefrancoisGandon-TIA2011-ULiS.pdf	○	○	○	○	○	○	○	○	○	○

Figure 5-1 Sample of One Survey Question.

5.3 Questions Description

- The goal of question 1 is to find the precision of Google scholar. The aim is to rank how many of the papers that are returned by Google Scholar are correct and strongly related to the query term. For this purpose, we have used a multiple choice question format. The respondents are asked to give a rating for the question on a scale from 1 to 10 by selecting one option they believe to be closest to the correct answer. The scale is marked by the two end points "not related" and "strongly related." The users decided which papers are very good, good, or bad in relation to the query term. From the responses to this question, we are going to find the number of related papers from the first top 10 papers that are returned from Google Scholar.
- The objective of questions 2, 3, and 4 is generally similar to the first question. However, instead of finding the precision measure of Google Scholar, they will find the precision measure of Recommender EngineII (REII), Recommender EngineI (REI), and Recommender EngineIII (REIII), respectively.
- The goal of question 5 is to find all the papers that are returned by Google Scholar but not by recommender EngineII and see whether they are related to the query term or not. This question serves a double purpose, in the sense that we can make sure that respondents have noted the questions that are not appearing in REII and rated them again. At this juncture, we use this rating in

addition to the ratings obtained from questions 1-4 in order to calculate the recall.

- The aim of questions 6 and 7 are similar to question 5, but to find the recall measure for REI and REIII, respectively

5.4 Results

Participants samples are show in Figures 5.2, 5.3, and 5.4. Each figure shows a sample of ratings for the top ten papers for the query terms, "ontology," "semantic web," and "web service," respectively:

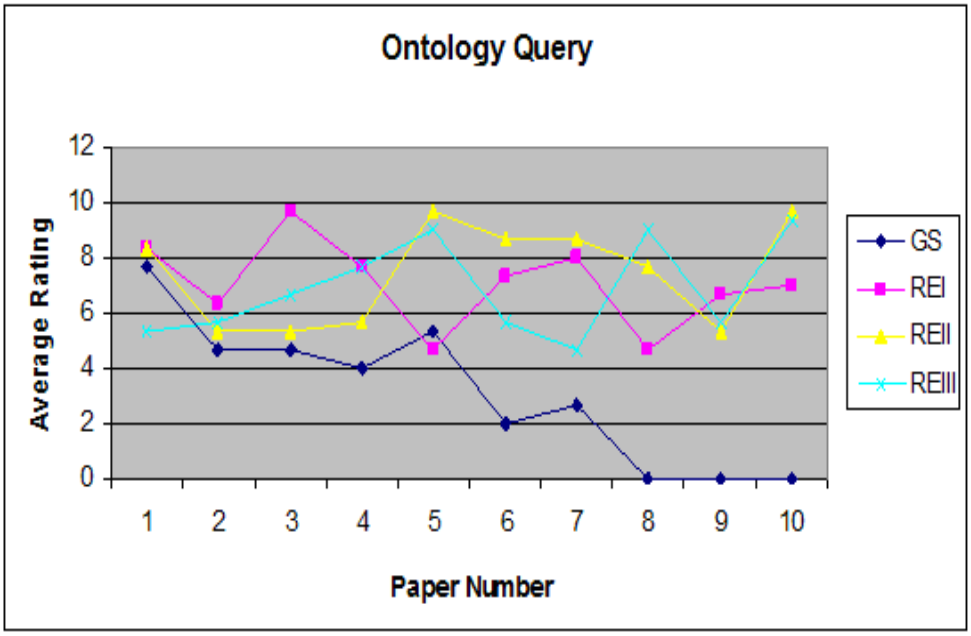

Figure 5-2 Rating Top Ten Papers by Four Search Engines for the Ontology Query.

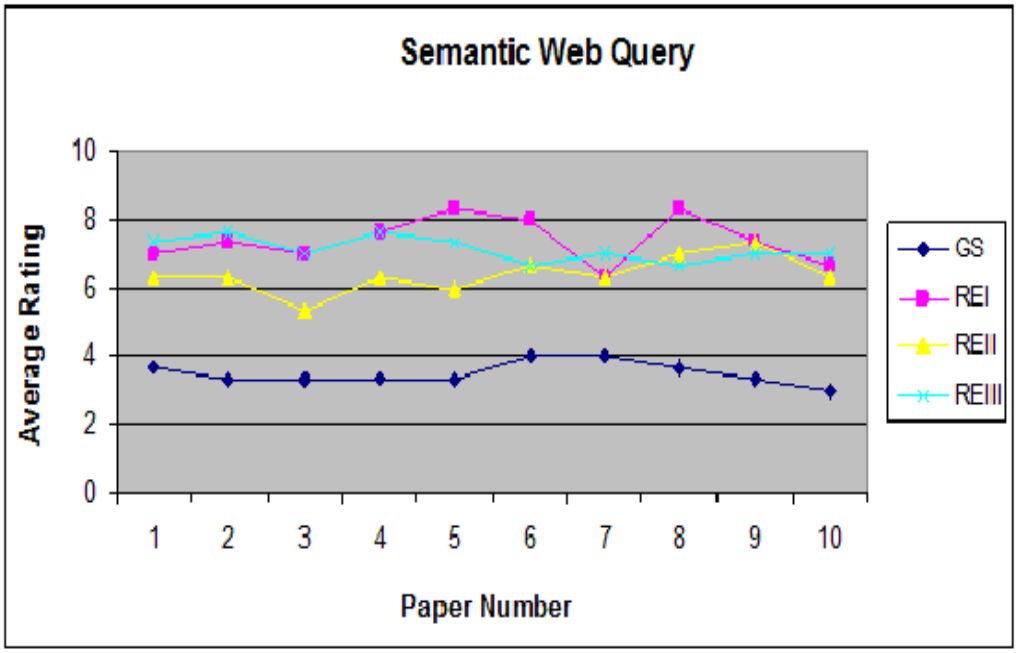

Figure 5-3 Rating Top Ten Papers by Four Search Engines for Semantic Web Query.

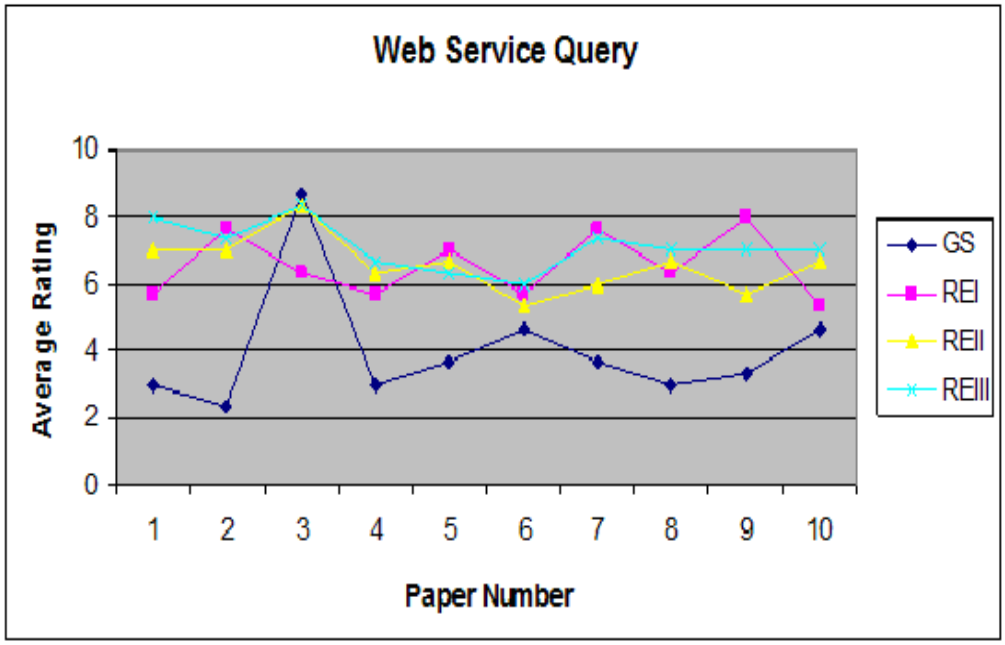

Figure 5-4 Top Ten Papers by Four Search Engines for Web Service Query.

5.5 Precision and Recall

In this section, we describe how precision and recall were calculated, followed by a review of the overall results that have been computed and used in precision and recall measures. Precision is the number of correct answers divided by the whole number of answers. In our repository, in some cases Google Scholar returned less than 10 papers, our search engines returned 10 papers. All papers returned with different rankings. The participants assigned different numbers (1-10) to each paper. The following is the formula used for computing precision (P):

$$P = \frac{\sum_{i=1}^{n} r_i}{10.n}$$

Where

r_i is the average rating provided for paper number *i*

n is the overall number of papers

Figures 5.5, 5.6, 5.7, 5.8 and 5.9 show the precision results of all techniques for the query terms " ontology," "semantic web", "annotation," "social network," and "web service," respectively:

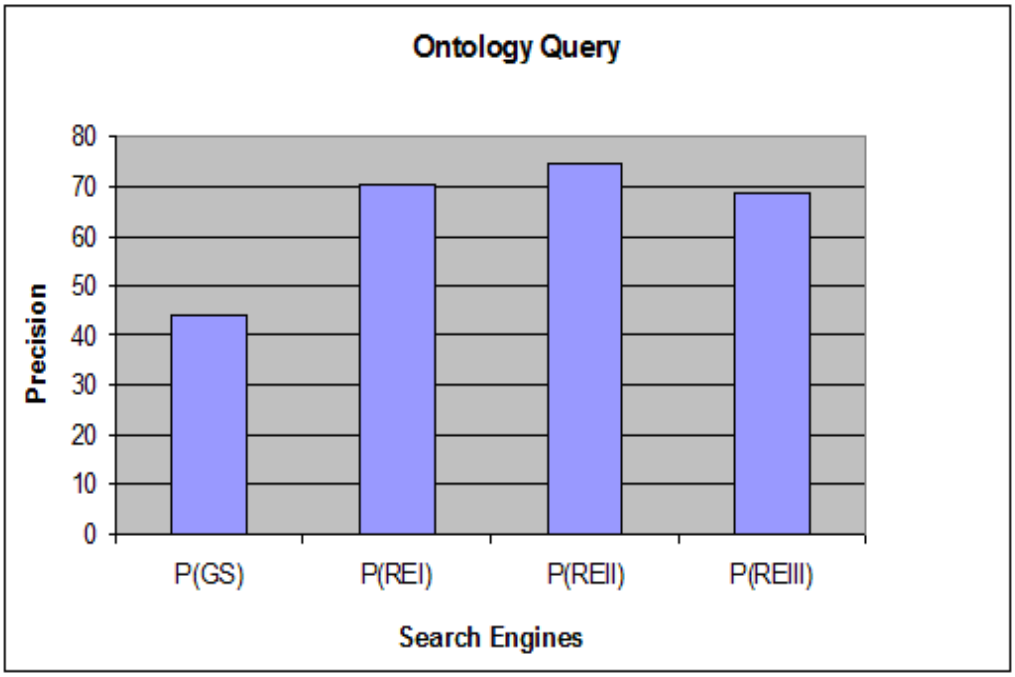

Figure 5-5 Precision of Four Search Engines for Ontology Query.

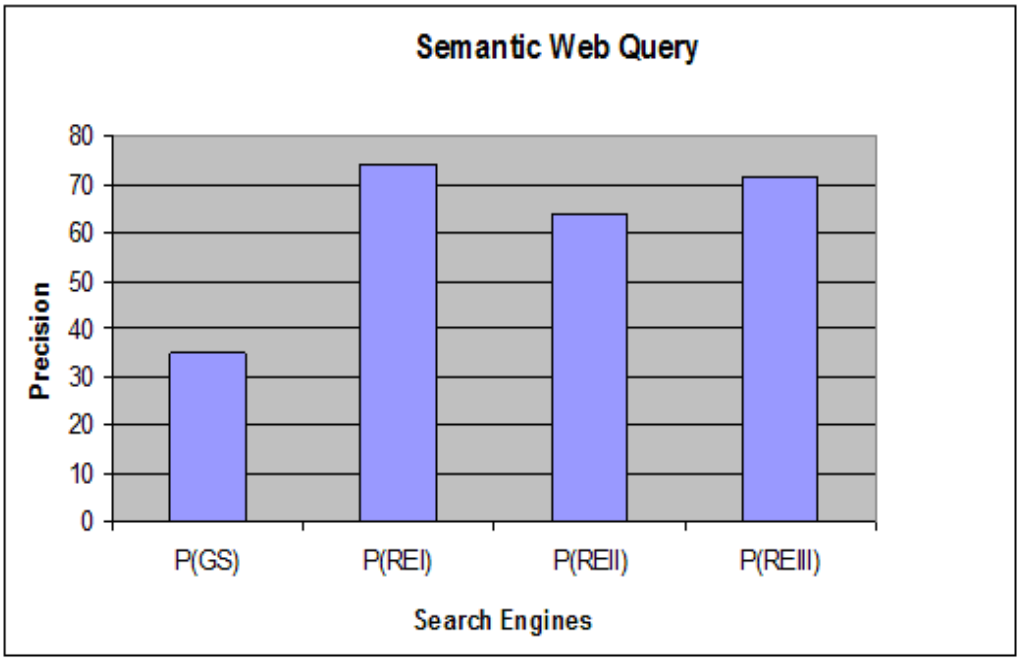

Figure 5-6 Precision of Four Search Engines for Semantic Web Query.

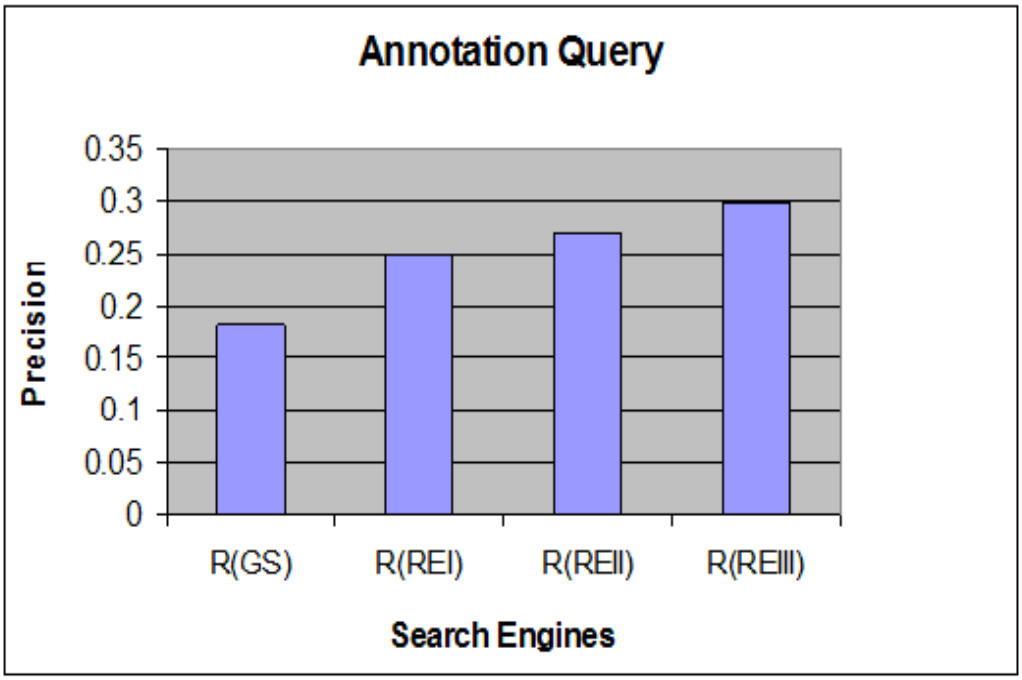

Figure 5-7 Precision of Four Search Engines for Annotation Query.

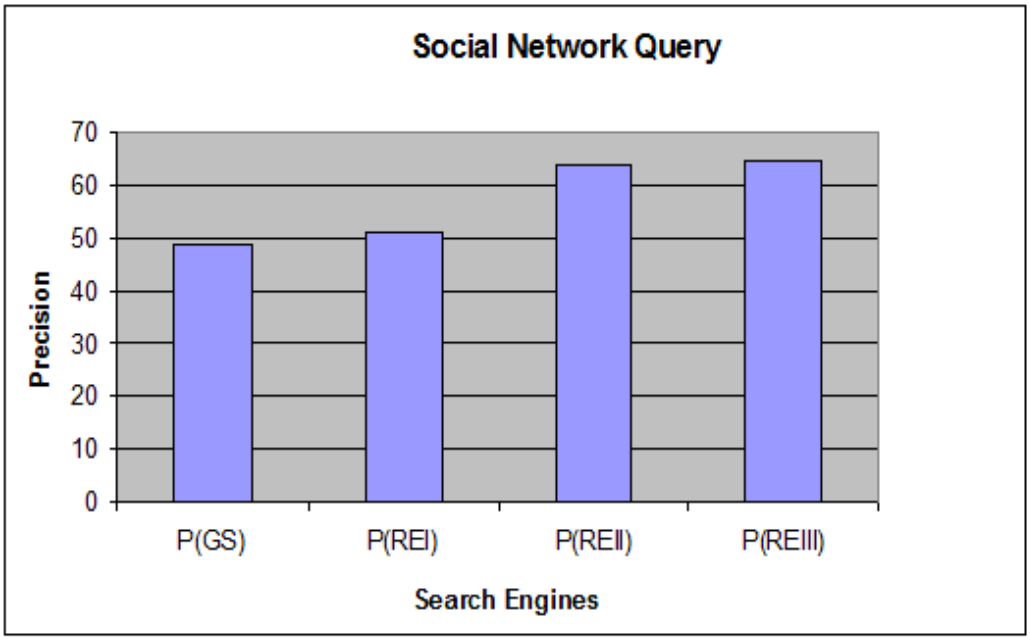

Figure 5-8 Precision of Four Search Engines for Social Network Query.

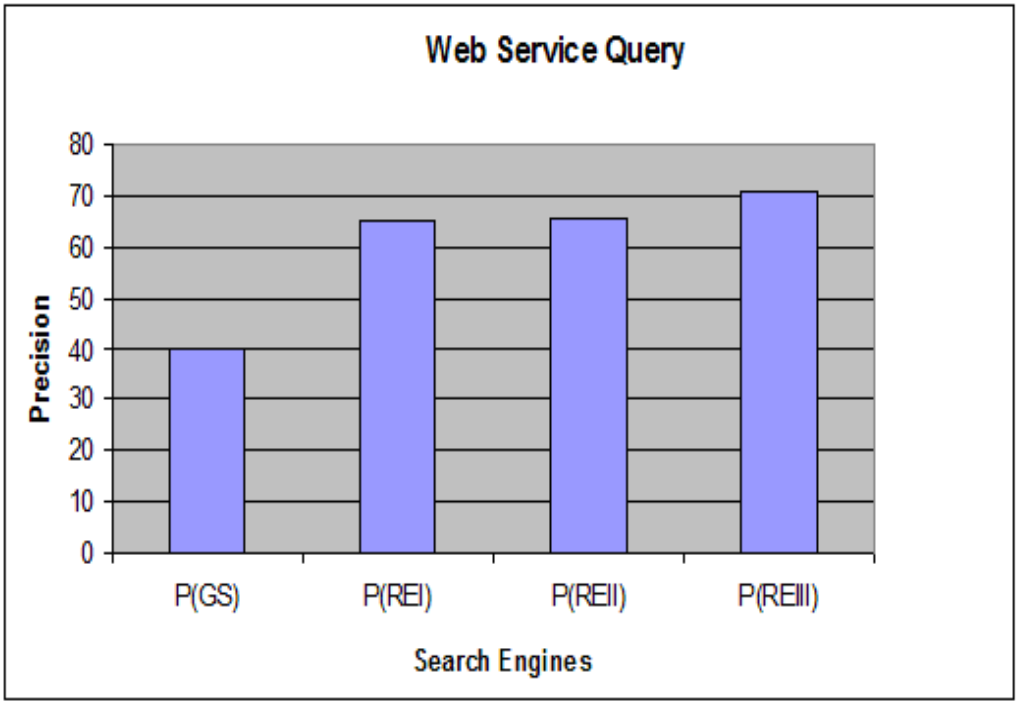

Figure 5-9 Precision of Four Search Engines for Web Service Query.

To calculate recall, we carried out a kind of simulation by estimating recall with a new formula. We calculate all the papers returned by all of the search engines and their ranking instead of the overall related papers because we do not have all of the related papers' numbers. We provide the recall formula in the following format:

$$\text{WH}=\sum_{i=0}^{n} r_i + \sum_{i=0}^{n} j_i + \sum_{i=0}^{n} k_i + \sum_{i=0}^{n} s_i$$

$$\text{R}=\frac{\sum_{i=0}^{n} r_i}{WH}$$

Where

r_i is the average rating provided for paper number i for system 1

j_i is the average rating provided for paper number i for system 2

k_i is the average rating provided for paper number i for system 3

s_i is the average rating provided for paper number i for system 4

m is the total number of papers

Figures 5.10, 5.11, 5.12, 5.13, and 5.14 demonstrate the result of recall measure in a chart of all search engines for the query terms “ontology," "semantic web," "annotation," "social network," and “web service," respectively:

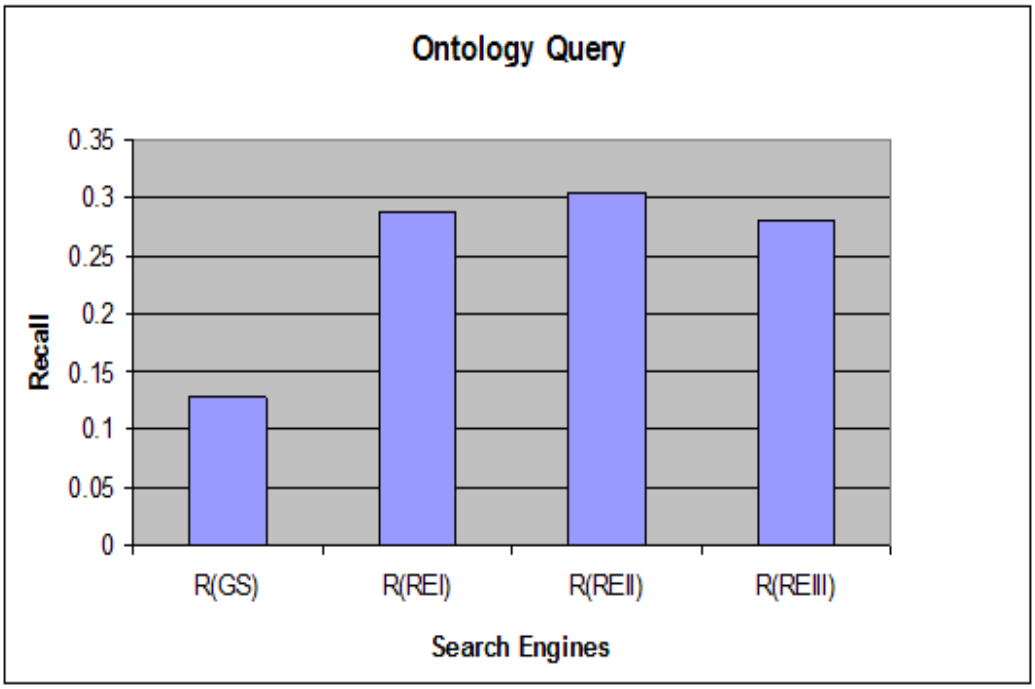

Figure 5-10 Recall of Four Search Engines for Ontology Query.

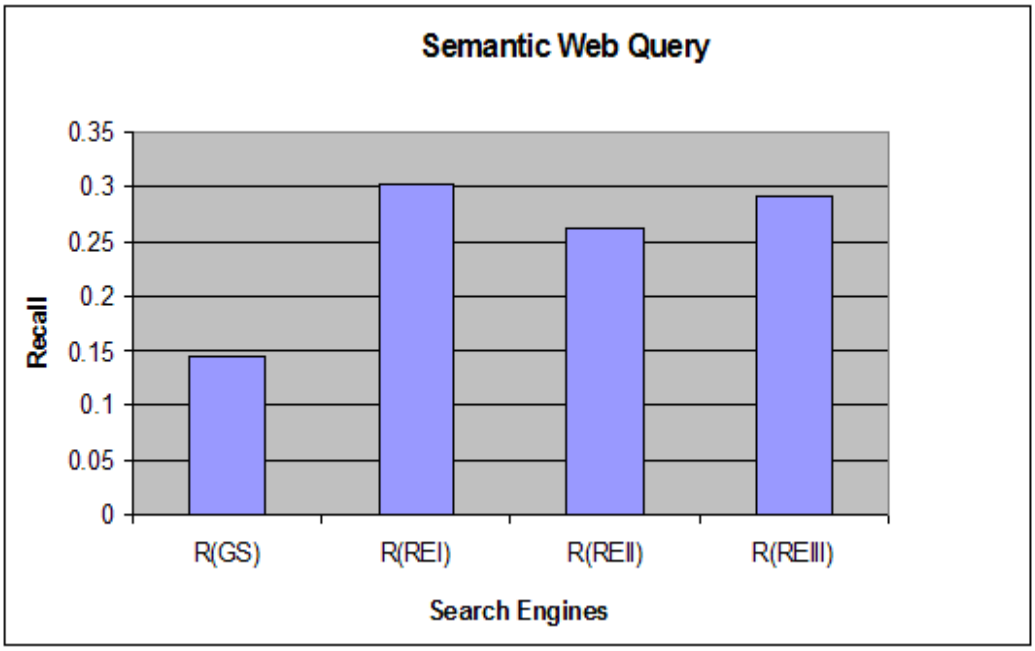

Figure 5-11 Recall of Four Search Engines for Semantic Web Query.

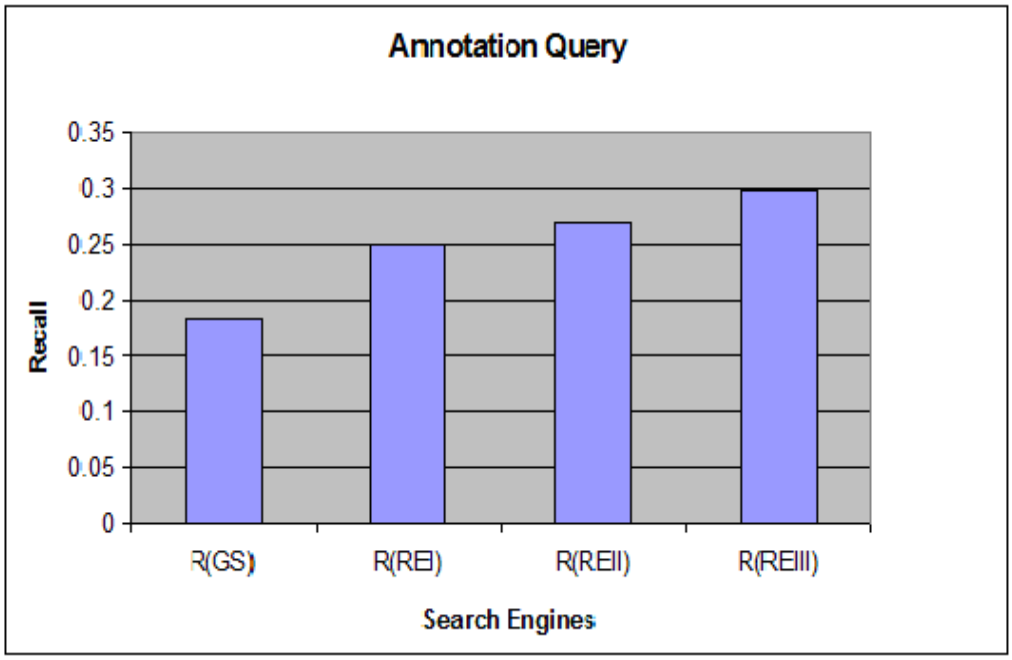

Figure 5-12 Recall of Four Search Engines for Annotation Query.

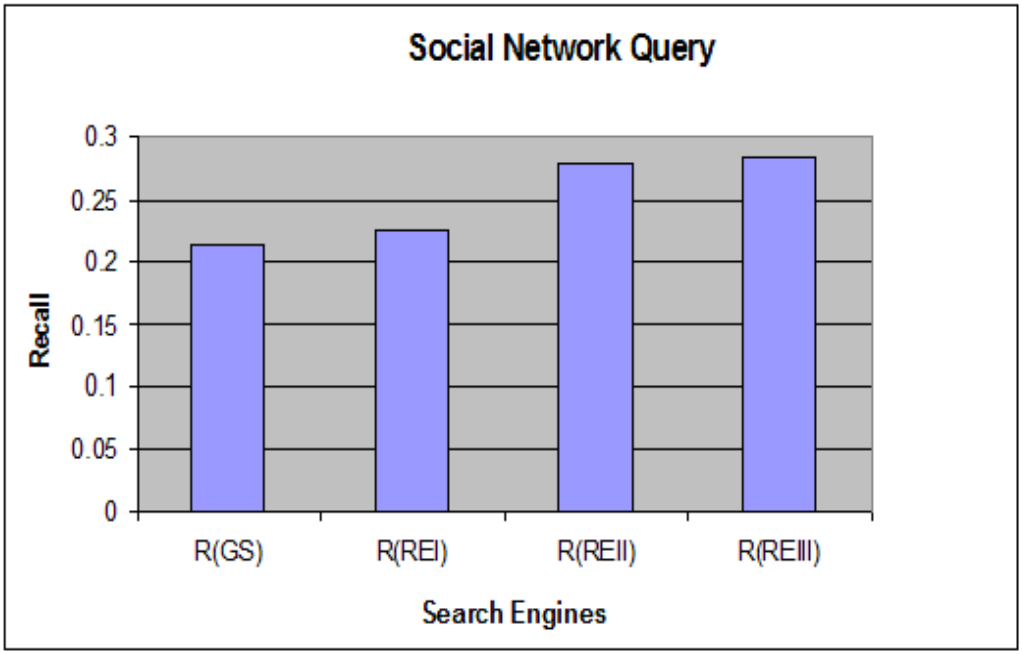

Figure 5-13 Recall of Four Search Engines for Social Network Query.

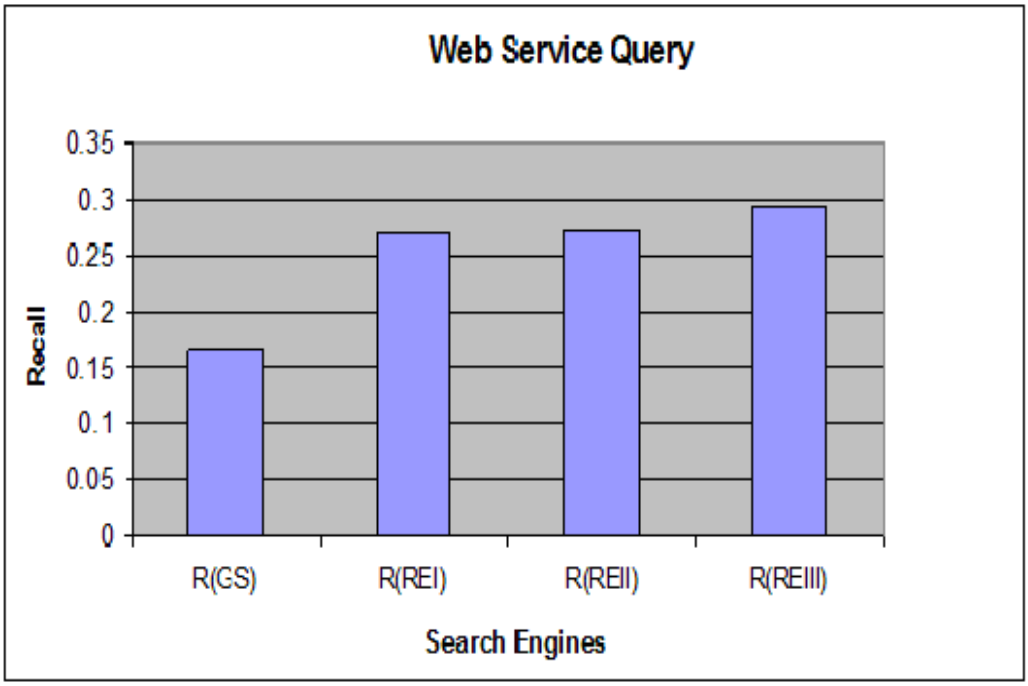

Figure 5-14 Recall of Four Search Engines for Web Service Query.

In conclusion, it can be maintained that all our three methods, namely REI, REII, and REIII are working more effectively than Google Scholar. As the results of the questionnaires show, the queries in GATE or Alchemy or their combination have different rankings. The reason for such a variation is that when the ontology is comprehensive, GATE tends to work better than Alchemy. However, when the ontology is not comprehensive, Alchemy has a tendency to outperform GATE. Subsequently, if the combination is less effective than GATE and Alchemy, it is an indication that one of them is outperforming the other, otherwise the combination will be the best.

Chapter 6:

6 Conclusion and Future work

6.1 Conclusion

Throughout this study, we have established that recommender systems continue to remain effective tools with the ability to prevail over the existing problem of information overload by making the most relevant contents available to the end users. To enhance their capabilities, semantic web technologies have found a strong foothold in recommender systems. This newborn phenomenon has been critically acclaimed by experts in the field and promoted by industry in hopes of drastically improving the quality of search engines. Despite the praise, there continues to be a dearth of analysis on the act of knowledge sharing in an effectual fashion, hence our decision as researchers to delve deep into this matter.

As such, semantic recommender systems which are an improvement to common recommender systems make use of a variety of tools including semantic web parts in an effort to recommend contents that fulfill the users´ long-term information needs and interests. Users' preferences and past behaviour are utilized through access to user profiles in an ongoing effort by recommender systems to predict materials the user may be interested in. Despite the advances made in the subsequent field, recommender systems continue to fall short of expectations. While existing recommender systems set out to use website ontologies to produce valuable web

recommendations, we have identified the need for developing a novel semantic recommender system with the ability to expand on existing systems.

In this thesis, we proposed designing a semantic recommender system that performs a semantic search and retrieves scientific papers in the field of Computer Science. This system finds the main topics of a given document, then annotates these topics using ontology-based components. We developed an annotator system with the capability to annotate academic Computer Science papers. The two comprehensive annotation techniques used in the current study are of Gate and Alchemy. We also used semantic-based matching techniques to measure the semantic similarity score of the given query phrase related to the main topics of a document and subsequently ranked the documents based on these measure scores.

We developed three different semantic search engines, based on annotation system and its corresponding semantic matching components, namely REI, REII, and REIII. These systems which are based on GATE, Alchemy, and their combination can search over an annotated dataset and find and rank the most relevant documents in relation to a given query. REI uses the domain-specific ontology-based annotation components and semantic similarity to rank documents based on their similarity score in order to accommodate the user. REII applies the LOD-based annotation component and the semantic similarity measuring component to rank documents based on similarity scores in an effort to provide the related papers corresponding to the query term. REIII is built on a combination of annotations provided by the domain-specific

ontology-based annotation component and the LOD-based annotation component, along with the semantic similarity measuring component.

In chapter 4 of this thesis, we have provided specific examples pertaining to the implementation of all the three aforementioned systems. In chapter 5, we thoroughly described and evaluated the performance of REI, REII, and REIII in order to retrieve the maximum number of relevant papers. We implemented the proposed system in the domain of Computer Science academic papers, and measured both precision and recall, and compared our results with Google Scholar.

6.2 Future Work

As a preliminary study, we were able to work on 100 papers, but any future investigation in this realm can expand the repository considerably. Future studies can enrich their experience by including a greater variety of topics and papers in the field of Computer Science and compare results with that of popular search engines like Google Scholar and Bing. Through expanding repositories and including all papers in the field of Computer Science, future studies can not only be greatly enhanced but also be comparable to real search engines.

As previously stated in the evaluation section of this study, GATE tends to function less effectively compared to Alchemy in some parts. This drawback could be due to the fact that our ontology was not comprehensive enough in some areas. Thus, we need to have a more comprehensive and satisfied ontology in order to ensure that GATE can work as effectively as Alchemy and is able to obtain valuable annotation

with a comprehensive ontology. In the current study, we used Alchemy as LOD-based annotation component, but for future research, the plug-in architecture that works with other annotation systems can be utilized. The ability to work with various components has the potential to make the system more flexible. In light of the time constraints of this study, the scope of the research was considerably limited. Nevertheless, the obtained experimental results show that our approach is promising and urges further research in a direction that will lead to even greater improvements to current semantics-based recommender systems.

Bibliography

[1] Adomavicius, Gediminas, and Alexander Tuzhilin. "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions." Knowledge and Data Engineering, IEEE Transactions on 17.6 (2005): 734-749.

[2] Felfernig, Alexander, Gerhard Friedrich, and Lars Schmidt-Thieme. "Guest editors' introduction: Recommender systems." Intelligent Systems, IEEE 22.3 (2007): 18-21.

[3] Breese, John S., David Heckerman, and Carl Kadie. "Empirical analysis of predictive algorithms for collaborative filtering." Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence. Morgan Kaufmann Publishers Inc., 1998.

[4] Middleton, Stuart E., David De Roure, and Nigel R. Shadbolt. "Ontology-based recommender systems." Handbook on ontologies (2009): 779-796.

[5] Zhang, Qian-Qian, and Ning Ye. "Collaborative Filtering Algorithm Adapting to Changes Over, Dynamic Time."

[6] Desrosiers, Christian, and George Karypis. "A comprehensive survey of neighborhood-based recommendation methods." Recommender Systems Handbook (2011): 107-144.

[7] Jannach, Dietmar. "Recommender Systems." ,University Klagenfurt, Universitätsstrasse 65-67,A-9020 Klagenfurt.

[8] Shoval, Peretz, Veronica Maidel, and Bracha Shapira. "An ontology-content-based filtering method." (2008).

[9] Knappe, Rasmus. Measures of semantic similarity and relatedness for use in ontology-based information retrieval. Diss. Roskilde University, Denmark 2005, 2006.

[10] Ramos, Juan. "Using tf-idf to determine word relevance in document queries."Proceedings of the First Instructional Conference on Machine Learning. 2003.

[11] Burke, Robin. "Hybrid recommender systems: Survey and experiments." User modeling and user-adapted interaction 12.4 (2002): 331-370.

[12] Robin Burke ,"Knowledge-based recommender systems", Department of Information and Computer Science,University of California, Irvine

[13] Malik, Sanjay Kumar, Nupur Prakash, and S. A. M. Rizvi. "Semantic Annotation Framework For Intelligent Information Retrieval Using KIM Architecture." International Journal of Web & Semantic Technology (IJWest) 1.4 (2010): 12-26.

[14] Wei, Kangning, Jinghua Huang, and Shaohong Fu. "A survey of e-commerce recommender systems." Service Systems and Service Management, 2007 International Conference on. IEEE, 2007.

[15] Yli-Koivisto, Jukka, and Juha Puustjärvi. "CoMet: An electronic newspaper prototype." Workshop on XML in Digital Media. In Proceedings of the 8th International Conference on Distributed Multimedia Systems (DMS'2002). 2002.

[16] Ricci, Francesco, Lior Rokach, and Bracha Shapira. "Introduction to recommender systems handbook." Recommender Systems Handbook (2011): 1-35.

[17] Popov, Borislav, et al. "KIM-a semantic platform for information extraction and retrieval." Natural language engineering 10.3-4 (2004): 375-392..

[18] Ghauth, Khairil Imran, and Nor Aniza Abdullah. "Learning materials recommendation using good learners' ratings and content-based filtering."Educational Technology Research and Development 58.6 (2010): 711-727.

[19] Shishehchi, Saman, et al. "Ontological Approach in Knowledge Based Recommender System to Develop the Quality of E-learning System." Australian Journal of Basic and Applied Sciences 6.2 (2012): 115-123.

[20] Melville, Prem, Raymod J. Mooney, and Ramadass Nagarajan. "Content-boosted collaborative filtering for improved recommendations." Proceedings of the National Conference on Artificial Intelligence. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2002.

[21] Dietmar Jannach, TU Dortmund, Gerhard Friedrich, Alpen-Adria Universität Klagenfurt ," Tutorial: Recommender Systems "International Joint Conference on Artificial Intelligence Barcelona, July 17, 2011

[22] Smyth, Barry, and Paul Cotter. "A personalised TV listings service for the digital TV age." Knowledge-Based Systems 13.2 (2000): 53-59.

[23] Linden, G., B. Smith, and J. York. “Amazon.com Recommendations: Item-to-ItemCollaborative Filtering”

[24] Miller, Bradley N., et al. "MovieLens unplugged: experiences with an occasionally connected recommender system." Proceedings of the 8th international conference on Intelligent user interfaces. ACM, 2003.

[25] Balabanović, Marko, and Yoav Shoham. "Fab: content-based, collaborative recommendation." Communications of the ACM 40.3 (1997): 66-72.

[26] Seyerlehner, Klaus. Content-based music recommender systems: Beyond simple frame-level audio similarity. Diss. PhD thesis, Johannes Kepler University, 2010.

[27] Konstan, Joseph A., et al. "GroupLens: applying collaborative filtering to Usenet news." Communications of the ACM 40.3 (1997): 77-87.

[28] Gipp, Bela, Jöran Beel, and Christian Hentschel. "Scienstein: A research paper recommender system." International Conference on Emerging Trends in Computing. 2009.

[29] Schafer, J. Ben, Joseph A. Konstan, and John Riedl. "E-commerce recommendation applications." Data mining and knowledge discovery 5.1 (2001): 115-153.

[30] Fernandez-Lopez, Mariano, and Oscar Corcho. Ontological Engineering: with examples from the areas of Knowledge Management, e-Commerce and the Semantic Web. Springer Publishing Company, Incorporated, 2010..

[31] Takács, Gábor, et al. "Scalable collaborative filtering approaches for large recommender systems." The Journal of Machine Learning Research 10 (2009): 623-656.

[32] Resnick, Paul, and Hal R. Varian. "Recommender systems." Communications of the ACM 40.3 (1997): 56-58.

[33] Schafer, J. Ben, Joseph Konstan, and John Riedi. "Recommender systems in e-commerce." Proceedings of the 1st ACM conference on Electronic commerce. ACM, 1999.

[34] Xiaoyuan Su, Taghi M. Khoshgoftaar, A survey of collaborative filtering techniques, Advances in Artificial Intelligence archive, 2009.

[35] Belkin, Nicholas J., and W. Bruce Croft. "Information filtering and information retrieval: two sides of the same coin?." Communications of the ACM 35.12 (1992): 29-38.

[36] Russell, Stuart Jonathan, et al. Artificial intelligence: a modern approach. Vol. 74. Englewood Cliffs: Prentice hall, 1995.

[37] Gruber, Thomas R. "A translation approach to portable ontology specifications." *Knowledge acquisition* 5.2 (1993): 199-220.

[38] Raskin, Robert G., and Michael J. Pan. "Knowledge representation in the semantic web for Earth and environmental terminology (SWEET)." *Computers & Geosciences* 31.9 (2005): 1119-1125.

[39] Wilks, Yorick. "Are ontologies distinctive enough for computations over knowledge." *IEEE Intelligent Systems* 19.1 (2004): 74-75.

[40] Smith, B. (2001). Onto logy: Philosophical and Computational, http://ontology.buffalo.edu/smith/articles/ontology_pic.pdf

[41] aDepartament de Llenguatges i Sistemes Informàtics, L. S. I. "Taking advantage of semantics in recommendation systems." *Artificial Intelligence Research and Development: Proceedings of the 13th International Conference of the Catalan Association for Artificial Intelligence*. Vol. 220. IOS Press, Incorporated, 2010.

[42] Gruber, Thomas R. "Toward principles for the design of ontologies used for knowledge sharing?." *International journal of human-computer studies* 43.5 (1995): 907-928.

[43] Gruninger, Michael, and Jintae Lee. "Ontology-Applications and Design." *Communications of the ACM* 45.2 (2002): 39-41.

[44] Grimes, Seth (January 21, 2010). "Breakthrough Analysis: Two + Nine Types of Semantic Search". InformationWeek. Retrieved March 25, 2012.

[45] John, Tony (March 15, 2012). "What is Semantic Search?".Techulator. Retrieved July 13,2012.

[46] Guha, Ramanathan, Rob McCool, and Eric Miller. "Semantic search." *Proceedings of the 12th international conference on World Wide Web*. ACM, 2003. Retrieved July 13, 2012.

[47] Sudeepthi, G., G. Anuradha, and M. Surendra Prasad Babu. "A Survey on Semantic Web Search Engine." International Journal of Computer Science Issues(IJCSI) 9.2 (2012).

[48]] Berners-Lee T., Hendler J., and Lassila. The Semantic Web. Scientific American, 2001.

[49] Menczer, Filippo, Gautam Pant, and Padmini Srinivasan. "Topic-driven crawlers: Machine learning issues." ACM TOIT (2002).

[50] Grimes, Seth. "Open Source Text Analytics". http://www.b-eye-network.com/view/9516.

[51] A JAPE tutorial from Press Association Images, UK. http://realizingsemanticweb.blogspot.ca/2009/07/jape-grammar-tutorial.html.

[52] Gibilisco, Stan. General Architecture for Text Engineering (GATE). Business intelligence - business analytics. December 2012.

[53] "*AlchemyAPI*". (2013). Term Extraction.

http://www.alchemyapi.com/api/keyword-extraction/

[54] Taylor, Julia M., Daniel Poliakov, and Lawrence J. Mazlack. "Domain-specific ontology merging for the semantic web." *Fuzzy Information Processing Society, 2005. NAFIPS 2005. Annual Meeting of the North American*. IEEE, 2005.

[55] Evermann, Joerg, and Jennifer Fang. "Evaluating ontologies: Towards a cognitive measure of quality." *Information Systems* 35.4 (2010): 391-403.

[56] Tymoshenko, Kateryna. *A General Framework for Exploiting Background Knowledge in Natural Language Processing*. Diss. University of Trento, 2012.

[57] Mihalcea, Rada, Courtney Corley, and Carlo Strapparava. "Corpus-based and knowledge-based measures of text semantic similarity." *AAAI*. Vol. 6. 2006.

[58] Corley, Courtney, and Rada Mihalcea. "Measuring the semantic similarity of texts." *Proceedings of the ACL Workshop on Empirical Modeling of Semantic Equivalence and Entailment*. Association for Computational Linguistics, 2005.

[59] Cunningham, Hamish. "GATE, a general architecture for text engineering." *Computers and the Humanities* 36.2 (2002): 223-254.

[60] Maynard, Diana, et al. "Ontology-based information extraction for market monitoring and technology watch." *ESWC Workshop" End User Apects of the Semantic Web"), Heraklion, Crete*. 2005.

[61] Milne, David, and Ian H. Witten. "Learning to link with wikipedia." *Proceedings of the 17th ACM conference on Information and knowledge management*. ACM, 2008.

[62] Gruber, Thomas R. "Toward principles for the design of ontologies used for knowledge sharing?." International journal of human-computer studies 43.5 (1995): 907-928

[63] Riloff, Ellen. "Information extraction as a stepping stone toward story understanding." Understanding language understanding: Computational models of reading (1999): 435-460.

[64] Kassim, Junaidah Mohamed, and Mahathir Rahmany. "Introduction to semantic search engine." *Electrical Engineering and Informatics, 2009. ICEEI'09. International Conference on*. Vol. 2. IEEE, 2009.

[65] Fang, Wei-Dong, et al. "Toward a semantic search engine based on ontologies." *Machine Learning and Cybernetics, 2005. Proceedings of 2005 International Conference on*. Vol. 3. IEEE, 2005.

Appendix

The following is a sample part of our ontology. We have selected this part of the ontology to show the main classes in our ontology:

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:protege="http://protege.stanford.edu/plugins/owl/protege#"
    xmlns:xsp="http://www.owl-ontologies.com/2005/08/07/xsp.owl#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns="http://www.owl-ontologies.com/Ontology1355552455.owl#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:swrl="http://www.w3.org/2003/11/swrl#"
    xmlns:swrlb="http://www.w3.org/2003/11/swrlb#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xml:base="http://www.owl-ontologies.com/Ontology1355552455.owl">
  <owl:Ontology rdf:about=""/>
  <owl:Class rdf:ID="Emerging_technologies">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Hardware"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Computing_or_technology_policy">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Social_and_professional_topics"/>
    </rdfs:subClassOf>
  </owl:Class>
```

```
  <owl:Class rdf:ID="Discrete_mathematics">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Mathematics_of_computing"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Network_components">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Networks"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Applied_computing">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Computing_Classification_System"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="General_and_reference">
    <rdfs:subClassOf
rdf:resource="#Computing_Classification_System"/>
  </owl:Class>
  <owl:Class rdf:ID="Security_in_hardware">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Security_and_privacy"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Data_management_systems">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Information_systems"/>
```

```
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Semantics_and_reasoning">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="Theory_of_computation"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Probability_and_statistics">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Mathematics_of_computing"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Enterprise_computing">
  <rdfs:subClassOf rdf:resource="#Applied_computing"/>
</owl:Class>
<owl:Class rdf:ID="Distributed_computing_methodologies">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="Computing_methodologies"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Network_services">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Networks"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="World_Wide_Web">
  <rdfs:subClassOf>
```

```
      <owl:Class rdf:about="#Information_systems"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Parallel_computing_methodologies">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#Computing_methodologies"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Collaborative_and_social_computing">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Human-centered_computing"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:about="#Theory_of_computation">
    <rdfs:subClassOf
rdf:resource="#Computing_Classification_System"/>
  </owl:Class>
  <owl:Class rdf:ID="Software_and_its_engineering">
    <rdfs:subClassOf
rdf:resource="#Computing_Classification_System"/>
  </owl:Class>
  <owl:Class rdf:ID="network_security">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#Security_and_privacy"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Law_social_and_behavioral_sciences">
```

```
    <rdfs:subClassOf rdf:resource="#Applied_computing"/>
  </owl:Class>
  <owl:Class rdf:ID="Arts_and_humanities">
    <rdfs:subClassOf rdf:resource="#Applied_computing"/>
  </owl:Class>
  <owl:Class rdf:ID="Organizations">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Proper_nouns_-
_People_technologies_and_companies"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Electronic_design_automation">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#Hardware"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Computers_in_other_domains">
    <rdfs:subClassOf rdf:resource="#Applied_computing"/>
  </owl:Class>
  <owl:Class rdf:ID="Hardware_test">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#Hardware"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Interaction_design">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#Human-centered_computing"/>
```

```
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Visualization">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#Human-centered_computing"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Security_services">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#Security_and_privacy"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:about="#Human-centered_computing">
    <rdfs:subClassOf
rdf:resource="#Computing_Classification_System"/>
  </owl:Class>
  <owl:Class rdf:ID="Network_protocols">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#Networks"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Printed_circuit_boards_">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#Hardware"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Network_algorithms">
```

```
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Networks"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Modeling_and_simulation">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Computing_methodologies"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Computer_forensics">
  <rdfs:subClassOf rdf:resource="#Applied_computing"/>
</owl:Class>
<owl:Class rdf:ID="Software_and_application_security">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Security_and_privacy"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Accessibility">
  <rdfs:subClassOf rdf:resource="#Human-centered_computing"/>
</owl:Class>
<owl:Class rdf:ID="electronic_commerce">
  <rdfs:subClassOf rdf:resource="#Applied_computing"/>
</owl:Class>
<owl:Class rdf:ID="Continuous_mathematics">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Mathematics_of_computing"/>
  </rdfs:subClassOf>
```

```
</owl:Class>
<owl:Class rdf:ID="Integrated_circuits">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Hardware"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Software_notations_and_tools">
  <rdfs:subClassOf rdf:resource="#Software_and_its_engineering"/>
</owl:Class>
<owl:Class rdf:ID="Power_and_energy">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Hardware"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Professional_topics">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Social_and_professional_topics"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Cryptography">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Security_and_privacy"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Operations_research">
  <rdfs:subClassOf rdf:resource="#Applied_computing"/>
</owl:Class>
```

```
  <owl:Class
rdf:ID="Human_and_societal_aspects_of_security_and_privacy">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#Security_and_privacy"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Dependable_and_fault-
tolerant_systems_and_networks">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Computer_systems_organization"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Document_management_and_text_processing">
    <rdfs:subClassOf rdf:resource="#Applied_computing"/>
  </owl:Class>
  <owl:Class rdf:ID="Systems_security">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#Security_and_privacy"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Real-time_systems">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#Computer_systems_organization"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Mathematical_analysis">
    <rdfs:subClassOf>
```

```
      <owl:Class rdf:about="#Mathematics_of_computing"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="User_characteristics">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#Social_and_professional_topics"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Document_types">
    <rdfs:subClassOf rdf:resource="#General_and_reference"/>
  </owl:Class>
  <owl:Class rdf:about="#Computing_methodologies">
    <rdfs:subClassOf
rdf:resource="#Computing_Classification_System"/>
  </owl:Class>
  <owl:Class rdf:ID="Information_systems_applications">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#Information_systems"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:about="#Security_and_privacy">
    <rdfs:subClassOf
rdf:resource="#Computing_Classification_System"/>
  </owl:Class>
  <owl:Class rdf:about="#Hardware">
    <rdfs:subClassOf
rdf:resource="#Computing_Classification_System"/>
```

```
  </owl:Class>
  <owl:Class rdf:ID="Information_theory">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#Mathematics_of_computing"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:about="#Social_and_professional_topics">
    <rdfs:subClassOf
rdf:resource="#Computing_Classification_System"/>
  </owl:Class>
  <owl:Class rdf:ID="Ubiquitous_and_mobile_computing">
    <rdfs:subClassOf rdf:resource="#Human-centered_computing"/>
  </owl:Class>
  <owl:Class rdf:ID="Design_and_analysis_of_algorithms">
    <rdfs:subClassOf rdf:resource="#Theory_of_computation"/>
  </owl:Class>
  <owl:Class rdf:ID="People_in_computing">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#Proper_nouns_-
_People_technologies_and_companies"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:about="#Networks">
    <rdfs:subClassOf
rdf:resource="#Computing_Classification_System"/>
  </owl:Class>
  <owl:Class rdf:ID="Software_creation_and_management">
```

```
  <rdfs:subClassOf rdf:resource="#Software_and_its_engineering"/>
</owl:Class>
<owl:Class rdf:ID="Software_organization_and_properties">
  <rdfs:subClassOf rdf:resource="#Software_and_its_engineering"/>
</owl:Class>
<owl:Class rdf:ID="Information_storage_systems">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Information_systems"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Models_of_computation">
  <rdfs:subClassOf rdf:resource="#Theory_of_computation"/>
</owl:Class>
<owl:Class rdf:ID="Formal_languages_and_automata_theory">
  <rdfs:subClassOf rdf:resource="#Theory_of_computation"/>
</owl:Class>
<owl:Class rdf:ID="Network_architectures">
  <rdfs:subClassOf rdf:resource="#Networks"/>
</owl:Class>
<owl:Class rdf:ID="Architectures">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Computer_systems_organization"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Network_properties">
  <rdfs:subClassOf rdf:resource="#Networks"/>
</owl:Class>
```

```
  <owl:Class rdf:about="#Proper_nouns_-
_People_technologies_and_companies">
    <rdfs:subClassOf
rdf:resource="#Computing_Classification_System"/>
  </owl:Class>
  <owl:Class rdf:about="#Information_systems">
    <rdfs:subClassOf
rdf:resource="#Computing_Classification_System"/>
  </owl:Class>
  <owl:Class rdf:ID="Concurrent_computing_methodologies">
    <rdfs:subClassOf rdf:resource="#Computing_methodologies"/>
  </owl:Class>
  <owl:Class rdf:ID="Computational_complexity_and_cryptography">
    <rdfs:subClassOf rdf:resource="#Theory_of_computation"/>
  </owl:Class>
  <owl:Class rdf:ID="Cross_computing_tools_and_techniques">
    <rdfs:subClassOf rdf:resource="#General_and_reference"/>
  </owl:Class>
  <owl:Class rdf:ID="Hardware_validation">
    <rdfs:subClassOf rdf:resource="#Hardware"/>
  </owl:Class>
  <owl:Class rdf:ID="Symbolic_and_algebraic_manipulation">
    <rdfs:subClassOf rdf:resource="#Computing_methodologies"/>
  </owl:Class>
  <owl:Class rdf:ID="Information_retrieval">
    <rdfs:subClassOf rdf:resource="#Information_systems"/>
  </owl:Class>
```

```
  <owl:Class rdf:ID="Mathematical_software">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#Mathematics_of_computing"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Human_computer_interaction_-HCI">
    <rdfs:subClassOf rdf:resource="#Human-centered_computing"/>
  </owl:Class>
  <owl:Class rdf:ID="Logic">
    <rdfs:subClassOf rdf:resource="#Theory_of_computation"/>
  </owl:Class>
  <owl:Class rdf:ID="Very_large_scale_integration_design">
    <rdfs:subClassOf rdf:resource="#Hardware"/>
  </owl:Class>
  <owl:Class rdf:ID="Physical_sciences_and_engineering">
    <rdfs:subClassOf rdf:resource="#Applied_computing"/>
  </owl:Class>
  <owl:Class
rdf:ID="Intrusion_or_anomaly_detection_and_malware_mitigation">
    <rdfs:subClassOf rdf:resource="#Security_and_privacy"/>
  </owl:Class>
  <owl:Class rdf:ID="Computer_graphics">
    <rdfs:subClassOf rdf:resource="#Computing_methodologies"/>
  </owl:Class>
  <owl:Class rdf:ID="Companies">
    <rdfs:subClassOf rdf:resource="#Proper_nouns_-
_People_technologies_and_companies"/>
```

```
  </owl:Class>
  <owl:Class rdf:ID="Communication_hardware_interfaces_and_storage">
    <rdfs:subClassOf rdf:resource="#Hardware"/>
  </owl:Class>
  <owl:Class rdf:ID="Life_and_medical_sciences">
    <rdfs:subClassOf rdf:resource="#Applied_computing"/>
  </owl:Class>
  <owl:Class rdf:ID="Machine_learning">
    <rdfs:subClassOf rdf:resource="#Computing_methodologies"/>
  </owl:Class>
  <owl:Class rdf:ID="Network_performance_evaluation">
    <rdfs:subClassOf rdf:resource="#Networks"/>
  </owl:Class>
  <owl:Class rdf:ID="Embedded_and_cyber-physical_systems">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#Computer_systems_organization"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Database_and_storage_security">
    <rdfs:subClassOf rdf:resource="#Security_and_privacy"/>
  </owl:Class>
  <owl:Class rdf:ID="Theory_and_algorithms_for_application_domains">
    <rdfs:subClassOf rdf:resource="#Theory_of_computation"/>
  </owl:Class>
  <owl:Class rdf:about="#Computer_systems_organization">
    <rdfs:subClassOf
rdf:resource="#Computing_Classification_System"/>
```

```
  </owl:Class>
  <owl:Class rdf:ID="Formal_methods_and_theory_of_security">
    <rdfs:subClassOf rdf:resource="#Security_and_privacy"/>
  </owl:Class>
  <owl:Class rdf:ID="Artificial_intelligence">
    <rdfs:subClassOf rdf:resource="#Computing_methodologies"/>
  </owl:Class>
  <owl:Class rdf:ID="Education">
    <rdfs:subClassOf rdf:resource="#Applied_computing"/>
  </owl:Class>
  <owl:Class rdf:about="#Mathematics_of_computing">
    <rdfs:subClassOf
rdf:resource="#Computing_Classification_System"/>
  </owl:Class>
  <owl:Class rdf:ID="Robustness">
    <rdfs:subClassOf rdf:resource="#Hardware"/>
  </owl:Class>
  <owl:Class rdf:ID="Randomness_geometry_and_discrete_structures">
    <rdfs:subClassOf rdf:resource="#Theory_of_computation"/>
  </owl:Class>
  <owl:Class rdf:ID="Network_types">
    <rdfs:subClassOf rdf:resource="#Networks"/>
  </owl:Class>
</rdf:R/
```

Curriculum Vitae

Candidate's full name:

Esraa Hassan Jawad Al-Wakel

Universities attended :

University of Babylon,

Bachelor of Computer Science, 2001

University of New Brunswick, 2014