# MULTILINGUAL FRAMEWORK FOR ONTOLOGY-BASED SEMANTIC ANNOTATION OF HEALTH AND NUTRITION WEBSITES

BY

**SAEED YOUSEF ALBUKHITAN**

A Dissertation Presented to the

DEANSHIP OF GRADUATE STUDIES

**KING FAHD UNIVERSITY OF PETROLEUM & MINERALS**

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

# DOCTOR OF PHILOSOPHY

In

**COMPUTER SCIENCE AND ENGINEERING**

**May 2014**

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN- 31261, SAUDI ARABIA

**DEANSHIP OF GRADUATE STUDIES**

This thesis, written by **Saeed Yousef Albukhitan** under the direction of his thesis advisor and approved by his thesis committee, has been presented and accepted by the Dean of Graduate Studies, in partial fulfillment of the requirements for the degree of DOCTOR OF PHILOSOPHY IN COMPUTER SCIENCE AND ENGINEERING

Tarek Helmy

Dr. Tarek Helmy El-Basuny
(Advisor)

Dr. Umar Al-Turk
Department Chairman

M. Elsh

Dr. Moustafa Elshafei
(Co-Advisor)

Dr. Salam A. Zummo
Dean of Graduate Studies

Dr. Muhammed Al-Mulhem
(Member)

2/6/14

Date

عمادة الدراسات العليا
DEANSHIP OF GRADUATE STUDIES

Dr. Moataz Ahmed
(Member)

Dr. Mahmoud Elish
(Member)

© Saeed Yousef Albukhitan

2014

## *Dedication*

*This work is dedicated to the soul of my father, who passed away during my PhD study; and to my beloved mother, who covers me with her blessing and prayers; and to my wonderful wife for her support, sacrifices and understanding. Also, this work is dedicated to my children, whom I had little time to spend with; to my brothers, my sisters, my uncles, and my other relatives. Moreover, this work is dedicated to my friends and colleagues who encouraged and supported me during my study.*

# ACKNOWLEDGMENTS

I would like to take this opportunity to express my gratitude to everyone who helped me in conducting the research leading to this thesis. First, I would like to thank my advisor, Dr. Tarek Ahmed Helmy El-Basuny, for his guidance throughout the period of this research work. I would also like to thank my committee members, Prof. Moustafa Elshafei, Dr. Muhammed Al-Mulhem, Dr. Moataz Ahmed, and Dr. Mahmoud Elish for their thoughtful insights and guidance. I would like acknowledge the support provided by King Abdulaziz City for Science and Technology (KACST) through the Science & Technology Unit at King Fahd University of Petroleum & Minerals (KFUPM) for funding this work research through project No.10-INF1381-04 as part of the National Science, Technology and Innovation Plan. I extend to the project consultants Prof. Yuri Tijerino (Director of Web Science Lab, Kwansei Gakuin University, Japan) and Prof. Jeffrey M. Bradshaw (Florida Institute for Human and Machine Cognition, USA) for their valuable feedback during the initial phase of the project. Further, I would like to acknowledge the useful discussions I had with Ahmed Al-Nazer and Ali Mazhar as members of the project. I would also like to warmly thank my mother and family for all those intangible bits of reassurance given to me at key moments. Finally, I would like to express my gratefulness to both the Information and Computer Science and Computer Engineering departments for providing all necessary support for me during my graduate studies at KFUPM University.

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# LIST OF ABBREVIATIONS

| | | |
|---|---|---|
| **MLSAF** | : | Multi-Lingual Semantic Annotation Framework |
| **XML** | : | eXtensible Markup Language |
| **RDF** | : | Resource Description Framework |
| **OWL** | : | Web Ontology Language |
| **NE** | : | Named Entity |
| **NER** | : | Named Entity Recognition |
| **NLP** | : | Natural Language Processing |
| **POS** | : | Parts-of-Speech |
| **RE** | : | Relation Extraction |
| **SPARQL** | : | SPARQL Protocol and RDF Query Language |
| **IE** | : | Information Extraction |
| **IR** | : | Information Retrieval |
| **ML** | : | Machine Learning |
| **KB** | : | Knowledge Based |
| **ML** | : | Machine Learning |
| **HTML** | : | Hypertext Markup Language |
| **RSS** | : | Really Simple Syndication, Rich Site Summary |
| **USDA** | : | United States Department of Agriculture |

# ABSTRACT

Full Name : Saeed Yousef Ahmed Albukhitan

Thesis Title : Multilingual Framework For Ontology-Based Semantic Annotation of Health and Nutrition Websites

Major Field : Computer Science & Engineering

Date of Degree : May 2014

In order to achieve the vision of the semantic Web, it is important to have enough amount of semantic content on the Web. To produce such content on the existing Web, semantic annotation on the Web content is required. Semantic annotation adds machine-readable content to the Web sources. Because the Web is growing at an expositional rate, annotating content semantically by hand is not possible. In this thesis, we present an automatic multilingual semantic annotation framework (MLSAF) for semantic annotation of Web sources based on the domain ontologies. The MLSAF is a sub-framework of an ontology-based semantic annotation and personalized information retrieval (OSAPIR) framework developed by the project team members, including me, during this thesis research work. We present a semi-automatic learning approach that utilizes public multilingual resources, such as Wikipedia and WordNet, for building multilingual ontology. Moreover, we present different approaches to extract name entities and relationships from Web sources. As a case study, we have developed and expanded a set of multilingual ontologies related to food, nutrition, and health through a set of processes. We have also developed the MLSAF prototype, and we show how it can utilize these ontologies to extract name entities and relationships from different Web sources in order

to annotate and store them in the knowledgebase. We conducted several experiments to test the capability of MLSAF in recognizing the name entities and relationships using different approaches. Empirical evaluations of the MLSAF show promising performance results in terms of precision, recall, and F-measure. The outcome of the presented framework could be utilized by semantic Web searching applications to retrieve precise answers to the end user queries and to answer complex queries. An important feature of MLSAF is that it could be ported to other domains with minimal extension. This thesis also contributes to the vision of the semantic Web in the target domains, especially for Arabic Web sources.

# ملخص الرسالة

**الاسم الكامل:** سعيد يوسف احمد البخيتان

**عنوان الرسالة:** اطار متعدد اللغات للشرح الدلالي التلقائي بأستخدام شبكات المعاني لمواقع الغذاء و الصحة على الانترنت

**التخصص:** علوم و هندسة الحاسب الالي

**تاريخ الدرجة العلمية:** رجب 1435 هـ

إن نجاح تقنية الويب الدلالى يتطلب وجود كمية كافية من المحتوى الدلالي فى صفحات الويب. إحدى الطرق لإنتاج مثل هذا المحتوى هي من خلال الشرح الدلالي لمحتوى الويب الحالي. والمقصود بالشرح الدلالي هنا هو عملية إضافة علامات توضحية على المحتوى وذلك لمساعدة الحاسوب على فهم المحتوى النصي لصفحات الويب. وبالنظر للكمية الضخمة والمطّردة لمحتوى شبكة الانترنت فإن إتمام عملية الشرح الدلالى لمحتوي الويب بطريقة يدويه اصبح غير ممكن ولابد من اتمامه تلقائيا بواسطة برامج الحاسوب .في هذه الأطروحة، نقدم إطاريقوم تلقائيا بعملية الشرح الدلالي متعدد اللغات لصفحات الويب بإستخدام الانطولوجي (شبكة المعاني) الخاص بمحتوى صفحة. هذا الإطار المقترح هو جزء من إطارعام يستخدم شبكات الأنطولجى للشرح الدلالي لمحتوى الويب وكذلك لفهمً أسئلة المستخدم دلالياً ومن ثم إسترجاع المعلومات التي تتناسب مع احتياجات المستخدم. تقدم الأطروحة ايضا طريقة لبناء الأنطولوجى متعدد اللغات بشكل تلقائي إعتماداً على مصادر عامة مثل الويكيبيديا والوردنت. علاوة على ذلك تقدم الأطروحة أداة للشرح التلقائى لصفحات الويب بإستخدام أساليب متعددة لإستخراج أسماء الكيانات والعلاقات فيما بينها من مصادر الويب المختلفة. كدراسة موجهة قمنا بتطويرمجموعة من الأنطولوجى متعددة اللغات في مجال التغذية والصحة، ومن ثمّ استخدمنا الاداة المقترحة لإتمام عملية الشرح الدلالى لمجموعة من صفحات الويب في مجال التغذية

والصحة وذلك لإستخراج المعرفة من تلك الصفحات. وقد أجرينا عدة تجارب معملية بهدف دراسة كفائة وقدرة الإطار المقترح على التعرف على أسماء الكيانات والعلاقات فيما بينها من مصادر الويب المختلفة. حيث اظهرت النتائج كفائة عالية من حيث الدقة والشمولية فى إتمام عملية الشرح الدلالى لصفحات الويب في مجال التغذية والصحة. يمكن الاستفادة من مخرجات الإطار المقترح في تطبيقات البحث الدلالى للويب وذلك لإسترجاع إجابات دقيقة لإستفسارات المستخدمين وكذلك للإجابة على الاستفسارات المعقدة. علاوة على ذلك يساهم هذا العمل أيضا في تحقيق رؤية الويب الدلالي في المجالات المستهدفة خصوصا في اللغة العربية.

# CHAPTER 1

# INTRODUCTION

The Web has become the main source of information sharing. Most information produced by people is available on the Web. The number of documents on the Web related to domains such as health and nutrition is growing at an exponential rate. Most Web content is structured and represented to people's use only. In order to fully utilize the information available on the Web, it is necessary to have an automatic representation of the information in a form that machines could understand.

The semantic Web complements the existing Web by augmenting its content with meta-data layers that can be understood by machines. The semantic Web was first introduced by Tim Berners-Lee's in his book [1] in 1999, and 2001 marks the birth of the semantic Web [2]. Since then, the semantic Web has gained interest with the public and in research communities. The semantic Web defines knowledge at three levels [2]: syntactic, semantic using ontologies, and reasoning.

Semantic annotation is a process of adding a machine-understandable layer to existing Web content. Current search engines extract keywords from Web documents without actually having any clue about their meaning. Semantic annotation links Web content to predefined domain ontology resources such as concepts, instances, relationships, and

attributes. Semantic annotation provides the foundation for many semantic applications such as semantic search and semantic document clustering.

People use the Internet to search for health information for themselves, friends, and family. Most of the people who search the Internet for such information use it to make personal decisions that influence their health without consulting health professionals [3].

However, people searching for information related to health and nutrition on the Internet face many challenges, as described in [4]. According to [3], people use public search engines as starting point to look for health- and nutrition-related information. Users of search engines need to enter suitable keywords to get precise answers for what they are looking for. Search engines are limited with respect to the context and users' needs because they rely on the keywords the users provide to answer their queries. Another issue is related to the source of the provided information to the user; that is, the user may be provided with information from untrustworthy sources (e.g. social media, blogs, etc.).

In order to resolve these issues, Web search engines have to access knowledge-bases of semantically annotated data. To build a KB, websites have to be semantically annotated according to an agreed ontology related to domains of interest.

## 1.1 Motivations

The growth of the Internet has affected health and nutrition information providers' storage and sharing of their resources. However, these resources are usually scattered, independent, and often open to external access. A lot of human effort is wasted in

retrieving large sets of data from the Internet, despite its richness in content. If distributed data were integrated, it would be easy to discover more inferences between them. Some of these challenges motivated us to start this research in health and nutrition domains. We highlight some of the most important issues and motivations below.

**First**, health and nutrition information domains, which we are focusing on, are critical, and it is difficult to find precise information using general search engines.

**Second**, there is insufficient multilingual semantic infrastructure to help local users in finding relevant information about health and nutrition.

**Third**, the results retrieved from general search engines may come from untrustworthy sources.

**Fourth**, there is a lack of coordination and cooperation between different health and nutrition content providers, which results in duplicate efforts, even between noncompetitive establishments such as different governmental organizations.

**Fifth**, creating new content out of existing Web content could be done manually by linking websites together. The manually created content could quickly become inaccurate or obsolete if the source websites undergo changes in their structure or go out of service.

**Sixth**, to the best of our knowledge, there no annotation system for Web content targeting food, nutrition, and health domains that supports multiple languages, such as Arabic and Urdu.

## 1.2 Objectives and Methodology

The main aim of this work is to investigate the current status of multilingual semantic annotation research and to develop an automatic multilingual semantic annotation framework (MLSAF) for semantic annotation of Web sources based on the predefined ontologies for food, nutrition, and health domains. Detailed objectives of this thesis will include:

- Surveying the existing techniques and tools and then extending or developing a multilingual annotation generation tool for this research.
- Developing an integrated multilingual ontology for the health and nutrition domains suited for semantic Web annotation.
- Developing a multilingual semantic infrastructure that consists of information extraction (IE) and knowledge base of aggregated semantic information collected from trusted health and nutrition content providers.
- Testing and evaluating the developed framework in real-world test cases and scenarios.

## 1.3 Contributions

The research was conducted following the above research challenges and objectives, and it has resulted in the following major contributions of this thesis:

- Conducted a state-of-the-art intensive literature review of multilingual semantic annotation approaches.

- Proposed multilingual ontology learning framework that starts with initial ontology in one language and expands the ontology using pubic multilingual sources, such as Wikipedia, to include more concepts and spawn multiple languages based on article links between different language editions of Wikipedia.
- Construction of an integrated multilingual ontology for food, nutrition, and health domains that was carefully selected, amended, and modified to suit the purpose for extraction and manipulation of health and nutrition information from the Web.
- Collected and manually annotated multilingual data sources for health and nutrition domains for training and validation due to unavailability of such resources. Those annotated sources are available for public access.
- Presented different approaches to extract name entities and relationships from Web sources.
- Proposed an automatic semantic annotation framework for health and nutrition Web sources.
- Provided a prototype implementation of the proposed framework with a set of tests to measure its performance for food, nutrition, and health domains.
- Provided a rich multilingual knowledge base of health and nutrition information that could act as a backbone for any semantic Web application related to those domains.

.

## 1.4 Thesis Organization

The rest of this thesis is structured as follows. Chapter 2 presents a thorough survey on relevant research topics about multilingual semantic annotation systems based on a set of criteria. Chapter 3 presents the ontology-based semantic annotation and personalized information retrieval (OSAPIR) framework requirements and components, of which this thesis is tackling a major component. Chapter 4 presents an automatic MLSAF for semantic annotation of Web sources based on the domain ontologies. Chapter 5 presents the multilingual ontology learning approach. Chapter 6 presents the processes of developing the ontologies and details of developed ontologies related to food, nutrition, and health. Chapter 7 goes through the implementation details of the proposed framework. Chapter 8 presents experimental and comparison results with different settings. Finally, Chapter 9 concludes the thesis and highlights the directions of future research topics.

# CHAPTER 2

# LITERATURE REVIEW

In this chapter, we will discuss the work done so far in the area of semantic annotation of Web resources. We will present an overview of semantic annotation systems that support multilingual annotation of Web content. First, we will highlight different criteria that could be used to classify any multilingual annotation system. Then we will compare the existing annotation systems against those criteria. We will identify the limitations of all surveyed systems that motivated us to come up with the proposed framework.

## 2.1 Classification of Multilingual Semantic Annotation

Multilingual semantic annotation systems could be classified using different criteria. We limited our study to the most significant criteria that should exist in any annotation systems targeting multilingual Web sources and that are able to be adapted to any domain, to any language, and to different source formats.

Twelve criteria are used to evaluate the annotation systems with respect to input, output, usability and process of annotation. For Web content as an input, the related criteria are source acquisition, format, structure, domain, and language. Ontology representation is one criterion related to the input of the annotation system. The output-related criteria are the produced annotation factors such as output format, location, and ownership. Usability-

related criteria are the link to Linked Open Data (LOD) and Open API interfaces. The criteria related to the annotation process are the scope and approach used for annotation..

### 2.1.1 Data Acquisition

Acquiring Web data is an important task for a system-targeting Web source. A system could provide an engine that automatically acquires data from the Web and only accepts annotation from locally stored files. Commercial systems such as OpenCalais [5] and KIM [6], on the other hand, leave data acquisition tasks to the application that calls their application programmable interface (API) to provide the data in raw format or in URL links. The proposed framework addresses this task by providing a dedicated data acquisition layer that addresses all forms of data in addition to trustworthiness and reliability of the source required by some domains, such as health and nutrition.

### 2.1.2 Multilingual Handling

There are three main approaches to handle multilingual content. The first approach is to use independent monolingual annotation in which a system is built for each language and uses either a common ontology for all systems or a separate ontology for each language. In this approach, the integration between the annotations is either done at the ontology level or left for the semantic application to handle it [7]. Most monolingual annotation systems could be classified into this category if their ontologies are used or mapped by other annotation systems that support different languages.

In the second approach, annotation is performed in two stages. In the first stage, a language dependent annotation is performed using language-specific tools to analyze the content and then transform it into an intermediate model. In the second stage, the

generated intermediate model from the first stage is transformed into a final semantic annotation using language-independent tools [8].

The third approach is based on translating the information in the source language into a predefined language, after which the monolingual annotation is performed on the translated content. The performance of the annotation system is heavily depending on the quality of the translation process [9].

The proper approach to use depends on the intention of the generated annotation and the performance acceptance level of the annotation. For our target domains of health and nutrition, we adopted the second approach due to the language portability of this approach and the intended use of the annotation framework for semantic Web searching.

### 2.1.3 Multilingual Ontological Representation

There are different methods to represent multilingual information in ontologies. The most common method is based on using ontology annotation properties such as “rdfs:label,” “rdfs:comment,” or “skos:prefLabel” to describe the class, instances, attributes, and relations in different languages. For each language, there will be a separate value for those properties. This kind of representation is used in the DBpedia, Yago, Bebel, and Freebase multilingual ontologies, as shown in Figure 2.1.

The second approach of multilingual ontology representation is the use of a separate ontology for each language with mapping between corresponding ontology components. The mapping could be one-2-one between each pair of languages or using an agnostic ontology that acts as middleware between ontologies where mapping is only performed with this ontology to each language version of the ontology.


**Figure 2.1 Annotation Representation of Multilingual Ontologies**

The third approach is to use a common ontology with elements that have links to external data store for multilingual data. This approach is adapted by the LIR model [10]. Detailed comparisons of those three approaches are available in the study conducted by Espinoza et al. in [11]. The proposed framework accepts all three representations through the ontology management component [12] provided by the OSAPIR framework.

### 2.1.4 Automatic Annotation Approaches

There are four methods that could be used for annotating the content automatically [13]. The first method is to use wrappers or rules, which are most effective for content that follows regular patterns and formats such as lists and tables. These wrappers or rules are usually written by hand or facilitated by the tool in order to capture patterns for annotations [13].

The second method is based on IE that incorporates supervised learning, which relies on the availability of manually annotated content. The challenges for this approach are the selection of good and enough samples that represent the input space [13].

The third method is based on unsupervised machine learning (ML), which involves machines learning to annotate without the need for user supervision. The limitations of this approach are low accuracy and the need for huge datasets. Datasets are usually

collected from the Web starting with initial small seeds given by hand, ontology, or sample documents.

The fourth method is based on natural language processing (NLP) that includes part-of-speech (POS) tagging, morphological analysis, coreferencing, stemming, name phrase chunking, and parsing. NLP is better suited for textual content that adheres to grammatical constraints. The aim of NLP is to have a full understating of the textual content of the input document. The challenges are the availability and quality of those tools for all languages. The performance of NLP tools for Latin languages is highest because they receive the most research attention in both academic and commercial fields. Another challenge is the processing speed, which is important for the large-scale processing of Web resources. The most commonly used NLP toolkits offering multilingual processing capability and that support for heterogeneous document formats are LingPipe,[1] Apache OpenNLP,[2] Apache UIMA,[3] Stanford NLP,[4] GATE,[5] Mallet,[6] Natural Language Toolkit (NLTK),[7] and ScalaNLP.[8] The proposed framework provides the flexibility to use all three approaches with the possibility to build an ensemble from the output of multiple annotation methods.

[1] http://alias-i.com/lingpipe
[2] http://opennlp.apache.org
[3] http://uima.apache.org
[4] http://nlp.stanford.edu
[5] http://gate.ac.uk
[6] http://mallet.cs.umass.edu/
[7] http://nltk.org
[8] http://www.scalanlp.org/

### 2.1.5 Supported Document Formats and Structure

Most Web sources have encoded markup formats, such as HTML and XML, which are supported by most annotation tools. MS Office files and PDF are common formats among the Web documents. Some annotation tools are capable of parsing those formats. In addition to textual data, there are a lot of media data such as images and video files exist in the Web. There are languages encoding those Web sources, such as ISO, UTF-8, UTF-16, and ASCII, which need to be handled by a multilingual annotation system besides the heterogeneity of the document formats.

Input documents could have different structural formats. The context could be pure text, represented in natural language text that follows grammatical rules, or in a semistructural format, such as lists and tables. Some annotation systems are capable of handling more content formats than other systems. The proposed framework supports most public documents' formats through the use of reliable document parsers.

### 2.1.6 Supported Languages

Semantic annotation systems could be classified based on the number of languages they support and the diversity of languages. Most multilingual semantic annotation systems support Latin languages because they share common characteristics; furthermore, there is a wide availability of linguistic resources for those languages. Asian, African, and Middle Eastern languages have different characteristics than Latin languages [14], with regard to representing syntactic and semantic information, for example. The proposed framework initially supports two distinct languages, namely, Arabic and English. Adding a new

language requires very little effort; it simply involves selecting a set of NLP tools for the new language along with some customization for entity and relation tasks.

### 2.1.7 Supported Domains

Annotation systems could be classified based on whether it is possible (and if so, the effort needed) to port the system to a new language and a new domain. For the language portability, the three different approaches mentioned in Section 2.1.1 have different capabilities. In the first approach, adding a new language requires adding a complete annotation pipeline for the new language and may also require adding a new ontology for the language and mapping to ontologies for other languages. In the second approach, it requires only adding a language-dependent part by providing annotation tools for the new language. In the third approach, it requires having a robust translation of the new language to be supported by the system.

For the domain portability, it is possible and easy to adapt new domains by the system. Some annotation systems are domain specific and some are open domains. The performance of a domain-specific annotation system is expected to be higher and richer on that domain compared to the open domain systems. To port the domain-specific annotation systems such as OntoMiner [15] into a new domain requires adding a new ontology describing the new domain and annotation tools specific to the new domain. To port open domain-specific annotation systems such as KIM [6], it is first necessary to add domain ontology to filter or select the domain-specific information from the output of the system and to map the selected items to an ontology of the new domain. In the proposed framework, we adapt the second approach and facilitate the portability to a set of

guidelines and segregation between domain-independent and domain-specific components.

### 2.1.8 Language and Domain Portability

Annotation systems could be classified based on the possibility and effort needed to port the system to a new language and a new domain. For the language portability, the three different approaches mentioned in Section 2.1.1 have different capability. In the first approach, adding a new language requires adding a complete annotation pipeline for the new language and may also requires adding a new ontology for the language and mapping to ontologies for other languages. In the second approach, it requires only adding a language dependent part by providing annotation tools for the new language. In the third approach, it requires to have robust translation of the new language to be supported by the system.

For the domain portability, it is a measure of possibility and ease to adapt new domain by the system. Some annotation systems are domain specific and some are open domains. The performance of a domain specific annotation system is expected to be higher and richer on that domain compared to the open domain systems. To port the domain specific annotation systems such as OntoMiner [15] into a new domain requires adding a new ontology describing the new domain and annotation tools specific to the new domain. To port open domain specific annotation systems such as KIM [6], it requires first to add domain ontology, to filter or select of the domain specific information from the output of the system and to map of the selected items to an ontology of the new domain. In the

proposed framework, we adapt the second approach and facilitate the portability to a set of guidelines and segregation between domain independent and dependent components.

### 2.1.9 Location and Ownership of Annotation

Semantic annotation of Web resources could be either integrated or embedded with the source document or kept separate in a separate location, either locally or remotely. For semantic Web, the second approach is assumed in order to be able to do the inference more efficiently when annotations are stored on a remote server. Ownership of the annotation in this case will be on a remote server, and problems could arise if the information is sensitive and the document owner would like to expose that information to end users. Semantic annotation systems could be classified on the type of approach adopted with regard to the storage location and ownership handling. The proposed framework supports local and remote storage of annotation through the use of adapters and could be customized for each type of storage option. In the adapter, ownership and accessibility options could be specified.

### 2.1.10 Scope and Depth of Annotation

Systems could be characterized according to what part of the ontology is used to annotate the document's content. Ontology consists of concepts, instances, attributes, and relationships between instances. Annotation systems such as KIM [6] that produce high-quality, rich annotations are preferred over limited annotation systems such as DBpedia Spotlight [8], which is able to annotate concepts only. The proposed framework supports automatic annotation of concepts, instances, and relationships using different approaches and methods.

### 2.1.11 Entity Link to Linked Open Data Resources

Linked Open Data (LOD) knowledge bases are rich resources for open domains. Annotation systems that reference their annotated items to LOD items in addition to their own ontology could support the semantic application to display and integrate additional information for the annotated items. Common LOD knowledge bases such as DBpedia and Freebase are rich multilingual resources with various relationships between entities growing every day with new knowledge and supporting tools. Linking annotation items to LOD could help in benchmarking the annotation system with other systems that do the same, making LOD as a reference for comparisons and performance evaluation.

### 2.1.12 API interface

Providing external interfaces using Web services, SDK, and rich API would give the system's users the flexibility to extend and customize the annotation system response tailored to their needs. Also, this type of interface helps expand the system with new functionalities not foreseen at the inception time. Most of the commercial annotation systems, such as AlchemyAPI [16] and OntoMiner [15], provide this type of external interface, with different pricing plans.

## 2.2 Multilingual Annotation Systems

In this section, we survey the multilingual semantic annotation systems that make use of ontology during the extraction process and support more than one language.

A multilingual real-time news event extraction system was built to support six European languages [17]. The purpose of information redundancy from different languages and sources is to validate facts and increase recall and precision against the monolingual

system. Events are extracted using a monolingual event extractor, normalized, and then translated into English. Knowledge fusion is done using English language data store.

Multilingual OntoES (ML-OntoES) is the initial proposal for a multilingual extraction system that supports extraction from two languages, namely, English and Japanese [18]. This system is based on the Ontos extraction engine, which is capable of extracting information from semistructured data sources. It uses an object-oriented systems model (OSM) to store the ontology, with the extraction rule embedded in the ontology itself. The system was tested on a car advertisement domain. The extracted information is stored in canonical form, and conversion rules are used for cross-lingual information access. The extracted information is stored using single agnostic ontology. Adding a new language requires only building extraction rules for that language and conversion rules from and to the agnostic ontology. After that, the IE and query are possible with the new language.

The Interlingua QA system is a new knowledge representation model presented in [19] that supports multilingual IE. The system uses the universal networking language (UNL) as a formal knowledge representation. During the 1990s, UNL was developed by the University of the United Nations as a medium for the language-independent representation of Web content in order to overcome linguistic obstacles on the Internet. Words in the UNL are represented in a semantic way so that ambiguity is eliminated. It uses a modified form of English. Word Dictionary is used to store all language works with mapping to human languages. The extraction system consists of two modules: a language-dependent module and a language-independent module. The language-dependent module is responsible for word extraction from specific documents and queries using a shallow analysis of tokenization and lemmatization. Then it sends the extracted information to a

language-independent module for further processing. The language-independent module maps the extracted word to UL Words and stores the information in a UL document. UL documents are searched for the semantic query, and results are rendered back to the user in his or her query language.

The KIM annotation system [6] constructed a huge knowledge base of annotations using the IE technique. KIM annotates common named entities such as people and locations. KIM uses the general architecture for text engineering (GATE)[9] NLP toolkit for NLP processing in order to generate features for named entity recognition (NER) and relationships recognition. KIM uses a set of generic ontologies as core such as KIMO.[10] Large gazetteers built from ontologies and knowledge bases are used to recognize entities. GATE Jape rules are used to recognized relationships between entities. It is a challenging task to extend KIM to domain-specific ontologies.

DBpedia Spotlight [8] is an open-source system for automatically annotating natural language text with entities and concepts from the DBpedia knowledge base. The input of the process is a portion of natural language text, and the output is a set of annotations associating entities or concept identifiers (DBpedia URIs) to particular positions in the input text. DBpedia Spotlight provides programmatic interfaces for phrase recognition and disambiguation (entity linking), including a Web API supporting various output formats (XML, JSON, RDF, etc.). The annotations generated by DBpedia Spotlight may refer to any of 3.77 million things in DBpedia, out of which 2.35 million are classified according to cross-domain ontology with 360 classes. Through identity links, DBpedia also provides links to entities in more than 100 other languages and tens of other data sets.

[9] http://gate.ac.uk
[10] http://www.ontotext.com/kim/kimo.rdfs

The OpenCalais system [5] is a multilingual text analysis tool for enriching texts with semantic metadata. It uses the ClearForest text mining engine, which was acquired by Thomson Reuters in 2007. It is capable of processing unstructured documents with text, HTML, and XML formats using NLP engine and Machine Learning. It can produce semantic outputs with microformats, simple formats, JSON, N3, RDF with relevance estimation, and links to external data sources. It provides Web Service API and plugins to social media services such as WordPress and Fliker.

OntoMiner [15] is a multiagent system that uses linguistic rules based on a specified domain ontology. The rules were built by domain experts. OntoMiner extracts domain concepts and relationships from a given document, and the extracted information is then converted into RDF format and stored in a knowledge base. OntoMiner uses the GATE[11] NLP engine as its base, with additional tools and customization.

Rosoka Cloud [20] is an entity and relationship extraction engine that supports multilingual extraction. It is configured as a Web service that could be accessed using API. It accepts most common file formats (e.g. PDF, TXT, HTML, XML, MS Office files, etc.) and produces results in the form of XML or JSON metadata as well as embedded annotation in XML format. It returns the importance of recognized entities to the submitted document based on the linguistic context of the entity in the document. The Rosoka Cloud engine supports processing of more than 230 languages, including Arabic, Chinese, Bahasa Indonesia, English, French, Japanese, Korean, Portuguese, Russian, and Spanish.

[11] http://gate.ac.uk

PoolParty Extractor (PPX) [21] performs entity extraction using SKOS thesauri and the mappings from XML to SKOS. PPX uses a set of rules for sentence splitting, tokenization, stemming, and phrase construction on natural language documents. The words and phrases extracted from the document are assigned a relevancy score, and the candidate entities with the highest scores are chosen as keywords and key phrases to represent the input document. The scoring function of the extractor takes of the frequency and the position of the words into account (the more frequently a word is used, the higher the score it gets, and the words at the beginning of the text are considered more relevant, so they get a higher score than words at the end of the document). PPX initially processes text documents and perfumes statistical analysis on them. As PPX creates an extraction model from a thesaurus, the suggested tags can be controlled semantic tags instead of conventional keyword tags. (That is, PPX checks whether the analyzed text contains any words or phrases that match any concept label stored in the loaded thesaurus.) These concepts can be associated with synonyms, abbreviation, multilingual labels, and so on so that they will serve as better tags and enhance searches for tagged content. Additionally concepts can be mapped to resources from the Linked Data cloud, where even more background knowledge (like longitude and latitude or category and type information from DBpedia or Yago) is available, thereby enriching the semantic fingerprint of a document tagged with such concepts and offering more possibilities for improved content recommendation and similarity calculations. The calculated tag suggestions can also be used to classify documents according to predefined categories.

Rocket AeroText [22] is capable of recognizing common entities such as people, products, dates, and places. It is also able to discover the relationships between recognized

entities and perform event searches such as contract terms and customer transactions. Rocket AeroText supports IE in multiple languages and can be customized to recognize terms from different domains. AeroText is designed to be independent of domain and document type. The system requires a knowledge base to extract information. These knowledge bases are usually constructed by domain experts using AeroText's suite of helpful tools. AeroText provides sample applications, including biomedical research, counterterrorism, business intelligence, law enforcement, human resources, and document management. AeroText provides some knowledge bases for out-of-the-box functionality. The English knowledge base is the most highly developed as it contains rules for extracting nearly 100 entity types. Knowledge bases are also available in Arabic, Chinese (simplified and traditional), Spanish, and Indonesian languages. Users can immediately use the provided knowledge bases for extraction, but it is generally expected that they will bootstrap off the provided knowledge bases to meet the needs of their applications. Users can also develop a knowledge base from scratch, if desired.

The Attensity Exhaustive Extraction (AEE) system [23] extracts common 35 key entities such as people, places, and things automatically. It also extracts entity roles, relationships, and events in multiple languages, with no need to construct new rules. Using Attensity Triples, the system makes it possible to use contextual information to disambiguate entities. AEE could easily be integrated with applications that process textual information and enable users to have relevant, meaningful structured data from unstructured text.

NetOwl [24] uses three approaches to extract entities, relationships, and events: NLP, computational linguistics, and ML. The extracted information is used for advanced tasks

such as sentiment analysis, assignment of latitude/longitude to geographical references in text, translation of names between languages, name matching, and identity resolution.

SCOOBIE [25] extracts entities based on domain ontology using vocabulary, instances, regular expression, and word lists. It takes unstructured domain documents corpus as inputs, preprocesses them offline, and finally uses the resulting models, data, and index structures for online information extraction. Besides vocabularies, SCOOBIE requires Thomas's concepts and work items that are represented as instances inside its PIMO.

## 2.3 Annotation Systems Comparison

A summary of previously mentioned annotation systems using the criteria discussed earlier is shown in Table 2.1 and Table 2.2. Only two open-source systems are available that support annotating multilingual Web content. These two systems are the ones that we will focus on due to the availability of the system in terms of code and resources such as documentation, datasets, benchmarking, and research. For data acquisition, most of the systems left this task to the users, except for KIM and OntoMiner, which provide their own crawlers. For the multilingual handling, only two systems—NetOwl Extractor [24] and Rosoka [20] support the annotation time and ontology level. Both of those systems are commercial systems and must be accessed through a pricing plan. For the multilingual representation of ontologies, most of the surveyed systems support only the ontology representation based on the annotation properties, with no evidence of support for other formats. For supported domains, seven systems are domain-independent systems, and the other six provide adaptation for some domains. None of these domains provides annotation to food, nutrition, and health domains. For supported languages, three systems

support Arabic, and they are all commercial systems using linguistic rules and NLP processing. For the annotation approach, most of the systems provide a single method of annotation—either NLP or ML. All commercial systems and open source systems provide API interfaces that could allow developer to develop their own applications based on these systems. For document formats, all systems support HTML and textual data, and a few support other common formats. For document structure, two systems—OntoMiner and ML-OntoES—support annotation of semistructured information stored in tables.

**Table 2.1 Comparisons Summary of the Annotation Systems using Factor (1-6)**

| System | Data Acquisition | Multilinguality Handling | Supported Domains | Supported Languages | Automatic Annotation Approaches | Document Supported Formats |
|---|---|---|---|---|---|---|
| **AeroText** [22] | Left to the user | N/A | domain-independent | Arabic English, Spanish, Chinese, Indonesian | linguistic rules | plain text, HTML, XML |
| **AlchemyAPI** [16] | Left to the user | Ontology Level | multiple domains | 8 Latin Languages | Deep NLP, machine learning | plain text, HTML |
| **Attensity Exhaustive Extraction** [23] | Left to the user | Application level | domain-independent | 18 Latin,4 Asians , Arabic, Farsi, Bokmål | NLP | plain text |
| **DBpedia Spotlight** [8] | Wikipedia, Left the user | Ontology Level | domain-independent | 9 Latin, Turkish | NLP, statistical methods, machine learning | plain text, HTML |
| **NetOwl Extractor** [24] | Left to the user | Annotation Time, Ontology Level | More than 16 domains | Arabic, Persian English, Spanish , French, Russian Korean, Chinese | NLP | plain text, HTML, XML, PDF, MS Office |
| **OpenCalais** [5] | Left to the user | Ontology Level | domain-independent | English, French, Spanish | NLP, machine learning | plain text, HTML, XML |
| **PoolParty Extractor** [21] | Left to the user | Ontology Level | domain-independent | English, German, Spanish, French | NLP, machine learning, statistical methods | plain text, HTML, DOC, ODT, CMS |
| **Rosoka** [20] | Left to the user | Annotation Time, Ontology Level | multiple domains | Multilingual (230) | NLP | plain text, HTML, XML, SGML, PDF, MS Office |
| **SCOOBIE** [25] | local files | Ontology Level | domain-independent | English, German | NLP, machine learning | plain text, HTML |
| **OntoMiner** [15] | Crawler | Annotation Time | news (English, German, Russian, French), Medical (Russain), Wine (English), Travel, Shopping | English, German, Russian, French | NLP (GATE), linguistic rules (JAPE) | HTML, plain text |
| **ML-OntoES** [18] | Single URL and local files | Ontology Level | Car Adv, | English, Japanese | Gazetteer, Regular expression | HTML, plain text |
| **Interlingua QA system** [19] | local files | Machine Translation | domain-independent | | | plain text |
| **KIM System** [6] | Crawler and local files | Annotation Time | News domain | English | NLP (GATE), linguistic rules (JAPE) | HTML, plain text |
| **Proposed Framework** | Focused Crawler, RSS, local files | Annotation time, Ontology Level | Initially: food, nutrition, health. with provision for other domains | English and Arabic with provision for other languages | Gazetteer, Shallow NLP Deep NLP, linguistic rules (JAPE) machine learning, | All standard formats, custom format is supported |

**Table 2.2 Comparisons Summary of the Annotation Systems using Factor (6-12)**

| System | Supported Content Structure | Output Model | Annotation Scope | API Interface | Entity Link (LOD) | License |
|---|---|---|---|---|---|---|
| **AeroText** | Unstructured text | proprietary | named entities, relationships, events | Web Service | None | Commercial |
| **AlchemyAPI** | Unstructured text | XML, RDF-OWL, JSON, Microformat | Named entities, Relationships, Relation Sentiment, Keyword extraction | Java SDK | DBpedia, Yago, OpenCyc, Freebase, US Census, GeoNames, UMBEL, MusicBrainz, CIA, Factbook, CrunchBase | Commercial |
| **Attensity Exhaustive Extraction** | Unstructured text | RDFa | 35 key named entities, relationships, Events | Java SDK Web Services | None | Commercial |
| **DBpedia Spotlight** | Unstructured text | RDFa | named entities | Java SDK Web Services | GeoNames, Musicbrainz, CIA World Fact Book, DBLP, Freebase, OpenCyc, UMBEL, Project Gutenberg, DBtune Jamendo, Eurostat, Uniprot, Bio2RDF, US Census data | Open Source |
| **NetOwl Extractor** | Unstructured text | XML, RDF-OWL, others | named entities, relationships, events | APIs for : - Java - C Web services | None | Commercial |
| **OpenCalais** | Unstructured text | Microformats, Simple Format, JSON, N3, RDF | entities, facts, events, categories | Web Service - REST API - SOAP API | Dbpedia, Freebase, Reuters.com, GeoNames, Shopping.com, IMDB, LinkedMDB | Commercial |
| **PoolParty Extractor** | Unstructured text | RDF, OWL | named entities, relations, concepts that categorize the text, enrichments | Web Service | Dbpedia Yago | Commercial |
| **Rosoka** | Unstructured text | XML, JSON, RDF, others | named entities, relationships, attributes | Java SDK, Web Service - REST API - SOAP API | None | Commercial |
| **SCOOBIE** | Unstructured text | RDF, RDFa | instances, property values, RDFS types | None | None | Open Source |
| **OntoMiner** | Unstructured text, lists, Tables | XML, OWL | named entities, relations | Web Service | None | Commercial |
| **ML-OntoES** | Unstructured text, lists, Tables | Object-Oriented Systems Model (OSM) | named entities, relationships, attributes | None | None | Open Source |
| **Interlingua QA system** | Unstructured text | Universal Networking Language (UNL) | named entities, | None | None | Open Source |
| **KIM System** | Unstructured text | RDF, OWL | named entities, relationships, attributes, | Web Service, SDK | None | Commercial |
| **Proposed Framework** | Unstructured text, lists, Tables, structured data | All standard formats: OWL, RDF, RDFa, N3, | named entities, relationships, attributes | Java APIs and Web Service | Dbpedia at two level: annotation and ontology | Open Source |

From this survey, we highlighted some challenges and issues related to the annotation systems that support multilingual Web content as follows:

- We found limited support for domain specific acquisition of Web sources that address the trust and reliability of the source, which are important factors for critical domains such as health and nutrition.
- We found that there is no system that supports annotation for food, nutrition, and health domains.
- We found limited support for portability for domains such as food, nutrition, and health.
- We found a lack of resources for languages such as Arabic in order to process and annotate high-quality content.
- There is no clear distinction between language dependent and independent components in the surveyed systems.
- We found no validation of recognized information with respect to ontology, existing annotation, and the knowledge base provided by the surveyed systems.
- There was limited support of different output format and connection to different repositories.

Some of these issues and challenges will be addressed in this thesis, and the rest we will highlight as a direction for our future work.

# CHAPTER 3

# ONTOLOGY-BASED SEMANTIC ANNOTATION AND PERSONALIZED INFORMATION RETRIEVAL (OSAPIR) FRAMEWORK

In this chapter, we present the OSAPIR framework, which can adapt to any domain, such as the health and food domains. We start with the motivations for developing such framework, and then we list the requirements and describe the proposed framework.

## 3.1 Motivations

As we discussed in the earlier sections, current Web search engines are not able to provide meaningful results, and various domain experts are working independently in their knowledge areas, without any relation to each other, to solve the problems of the current Web. We also discussed how the semantic Web rescues us from this problem by providing ontologies for these knowledge areas, allowing us to add meaning to these Web resources when annotated by the semantic Web annotation process. These annotated

documents become understandable to machines and can be semantically reasoned for more relevant and concise answers to the user questions. The semantic Web can be a great value in the area of food and health, providing useful, concise, and trustworthy information to users.

There are hundreds of frameworks available for Web applications based on different design patterns and providing powerful building blocks such as model/view/controllers (MVC) for developing the application. Unfortunately, in the case of the semantic Web, we don't have the frameworks available to build end-to-end solutions leveraging the semantic technologies. Moreover, our objective is to bring the knowledge from these heterogeneous information sources not just for a single domain but, in fact, for multiple correlated domains where questions referencing multiple domains could be reasoned. We are proposing a framework that can be used to achieve such annotation and reasoning tasks, which will be explained in the sections to follow. Such a framework can be very useful in the area of food and health.

## 3.2 Requirements

A framework is a software platform for developing the application. It provides a basic foundation for software developers to creating application for a given platform. Generally, frameworks provide API for accessing their components where the framework itself serves as a pillars for building up the application and developers don't have to do everything from scratch. A framework may also include additional software libraries and other programs used in the software development process. Hence, these can be considered basic requirements for any common framework for development.

We aim to build a multilingual, cross-domain, personalized semantic Web search framework that can adapt to any domain, such as the health and food domains. Below we present the requirements for such a semantic Web search framework.

a) The framework should be applicable to any domain with minimal customization.
b) The framework should support multiple languages with respect to ontologies, Web sources, knowledge bases, and users' queries.
c) The framework should facilitate cross-domain integration of ontologies and knowledge bases.
d) The framework should support acquiring and annotating Web sources in heterogonous formats.
e) The framework should provide a mechanism to decide the trust level of the acquired Web sources.
f) The framework should generate standard semantic annotation formats for the acquired Web sources based on the domain ontologies.
g) The framework should semantically manipulate the users' queries.
h) The framework should provide reasoning capabilities for answering users' queries.
i) The framework should capture and model the users' preferences.
j) The framework should personalize the retrieved results.
k) The framework should support standard ontology representation format.
l) The framework should provide the required ontology management services to achieve the desired objectives (i.e., alignment of ontologies from different domains and languages)..

## 3.3 Proposed Framework

Based on an intensive literature review and discussions among the project team members, including the consultants,[12] we propose a framework that addresses the above requirements, namely OSAPIR. The proposed framework is capable to adapt to any domain by defining the domain ontologies, lexical resources, trust level, and seed Web sources. Furthermore, the framework supports multilingual ontologies, Web sources, and users' queries. Figure 3.1 shows the architecture of the proposed framework.

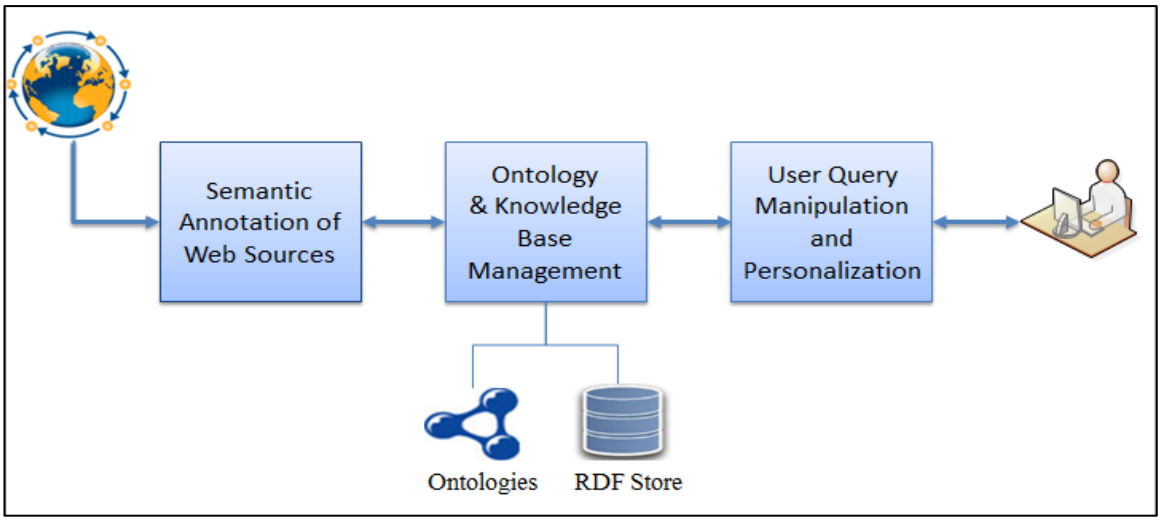



**Figure 3.1 Architecture of OSAPIR Framework**

The framework is divided into three major components: data acquisition and semantic annotation, ontology management, and semantic query manipulation and personalization components. Below is a brief description for each component.

[12] Dr. Jeff, Dr. Yuri

### 3.3.1 Data Acquisition and Semantic Annotation Component

The main goal for this component is to collect and annotate the contents of multilingual Web sources based on predefined domain ontologies. This component consists of two major layers: the acquisition layer and the semantic annotation layer.

The acquisition layer consists of multiple data integration tasks for the purpose of collecting data from Web sources related the targeted domains. The collected data from Web sources are then used by the annotation layer for semantic enrichment. The acquisition layer can be configured to collect data from specific Web sites based on certain criteria such as trust level or predefined seed Websites. The relevant Web sources are collected based on their relevancy to the domain ontologies. This layer supports processing all common Web document formats such as HTML, XML, PDF, Office documents, and multimedia.

The semantic annotation layer annotates the acquired Web sources based on the domain ontologies and the predefined cross-domain integration. Moreover, it provides multiple mechanisms to perform automated annotations for semistructured (i.e., tables) and unstructured (i.e., paragraphs) Web sources. This layer can produce embedded annotation inside the Web document using standard annotation languages such as RDFa, Microformat, and Microdata. In additional, it can produce stand-alone annotation using standard annotation languages such as RDF, N3, and Turtle.

In the next chapter, we will elaborate more on the architecture of this component with the title of an automatic MLSAF as the main focus of this thesis.

### 3.3.2 Ontology Management Component

The Ontology Management component takes care of managing a network of heterogonous ontologies and knowledge bases required by the main framework, namely, integration models for cross-domain and/or multilingual ontologies. It also provides different ontologies' management tasks for processing information (i.e., mapping various ontologies for more efficient sharing and reuse. This component can process any standard ontology representation language. It also provides API interfaces to access the ontologies by other two components of the proposed framework. In addition, it provides the knowledge bases with reasoning capabilities to allow semantic answers to users' queries. More details about this component are explained in the master's thesis titled "A Network of Heterogeneous and Distributed Ontologies for Health and Nutrition Information System" [12]. Figure 3.2 shows the architecture of this component.

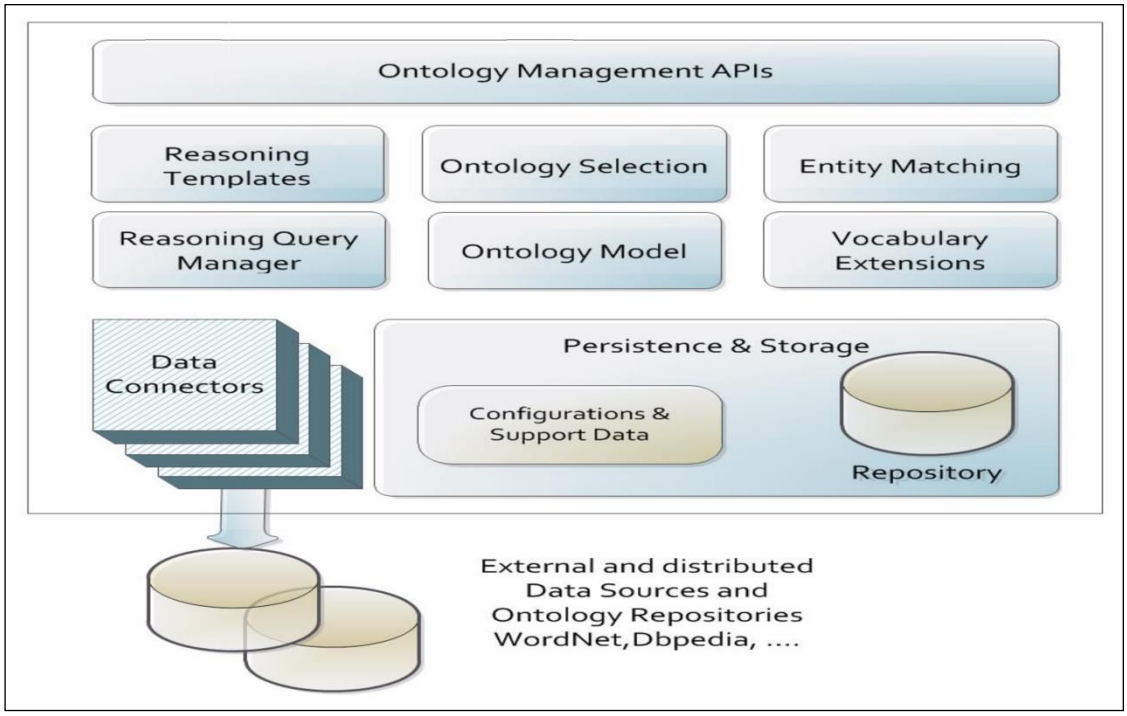



**Figure 3.2 Ontology Management Component**

### 3.3.3 Semantic Query Manipulation and Personalization Component

This component is used to interface with the end user, as it captures and models the user's preferences into a user profile. It semantically manipulates the multilingual user's queries and enriches them with more information from the user's profile. This component interacts with the ontology management component for query reasoning based on the domain ontologies and knowledge bases. Moreover, it personalizes the retrieved results and captures the user's interactions to enhance the user's profile and provide more relevant answers. More details about this component are explained in the PhD thesis titled "Agent-Based Framework for Semantic Query—Manipulation and Personalized Retrieval of Health and Nutrition Information" [26]. Figure 3.3 shows the architecture of this component.

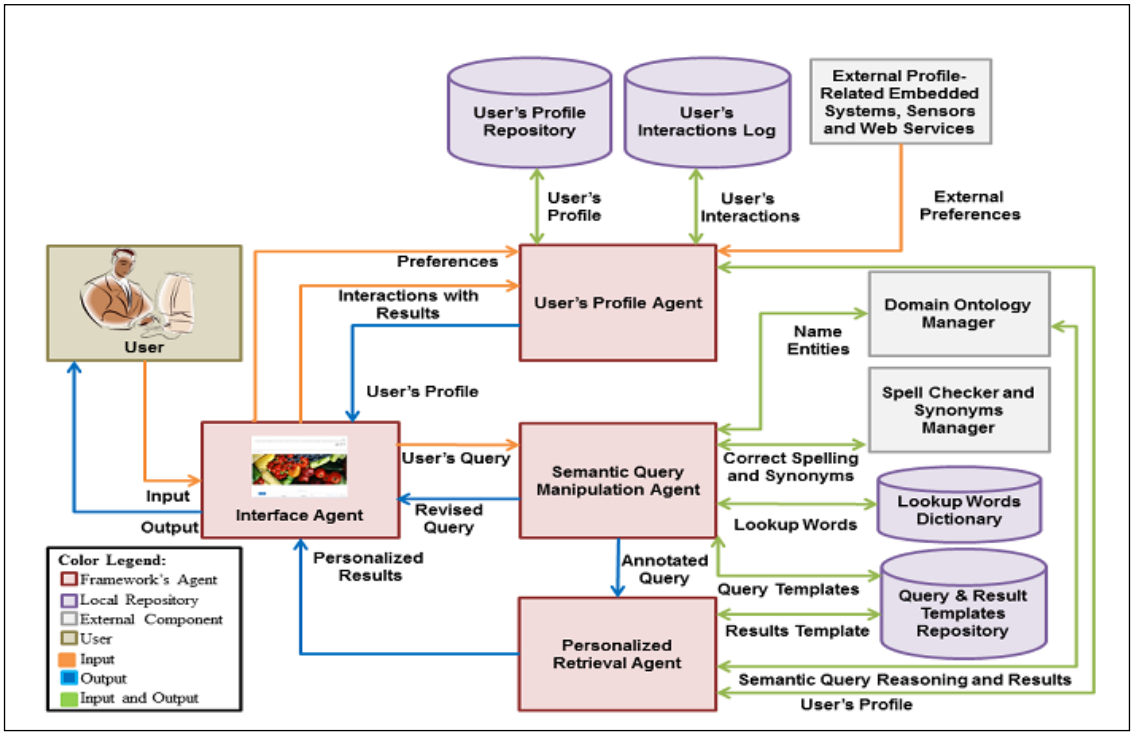



**Figure 3.3 Semantic Query Manipulation and Personalization Component (ASPIR)**

# CHAPTER 4

# THE AUTOMATIC MULTILINGUAL SEMANTIC ANNOTATION FRAMEWORK (MLSAF)

In this chapter, the aim is to present the automatic MLSAF for semantic annotation of Web sources based on the domain ontologies with the details of annotation tasks handled by the framework. The framework addresses the acquisition of Web sources, recognition of collected data, and storing the results into standard format. We will start with a set of requirements needed for a system with multilingual support that is capable of addressing annotation of domains such as food, nutrition, and health. We will highlight the features and capabilities of the proposed framework.

## 4.1 The Multilingual Semantic Annotation Requirements

Based on the intensive literature survey, we found that an open framework for multilingual semantic annotation of Web resources should have the following core characteristics:

- The architecture should be open for any language.
- The architecture should be open for any domain using domain ontologies.

- The architecture should be open for different representations of multilingual ontologies.
- The architecture should be open to process different document formats (e.g., HTML, XML, PDF, MS office, etc.).
- The architecture should be open for a different type of textual data (i.e., structured, semistructured, and unstructured data).
- The architecture should be open to add a new language through the dependent components that can be integrated without affecting the performance of the other multilingual components.
- The architecture should be open to generate different embedded annotation languages such as RDFa, microformat, and microdata in addition to standalone annotation languages such as RDF, N3, and Turtle.
- The architecture should be able to store the annotation result directly into any triple repository supporting a standard interface.
- The architecture should include a component that handles the acquisition of Web sources.

In addition, any implementation of this framework should consider the nonfunctional requirements such as modularity, scalability, speed, and usability. To achieve modularity, every processing task works as an independent module with clear interfaces and can be executed ad hoc or integrated in a processing pipeline. In order to achieve scalability, the system should support the handling of the usage of simultaneous ontologies and algorithms for concepts and relations recognition, as well as a huge number of Web

sources. To achieve usability, it should be easy for developers and researchers to use predefined pipelines, implement custom pipelines with provided modules, and/or implement new modules respecting previously specified interfaces. Moreover, typical preprocessing and basic NLP processing modules, such format parser, tokenization, and sentence splitting, should be part of custom pipelines and available for direct use and extension.

## 4.2 The Proposed Multilingual Semantic Annotation Framework

We propose MLSAF in order to support a variety of semantic annotation algorithms. The framework is designed as modular structure, as shown in Figure 4.1. The acquisition component collects Web sources, which will be then consumed by the semantic annotation pipeline to produce annotation that could be stored into standalone annotation files, embedded annotation inside the source documents, or directly into a semantic repository. The details of each component will be presented in the next sections.

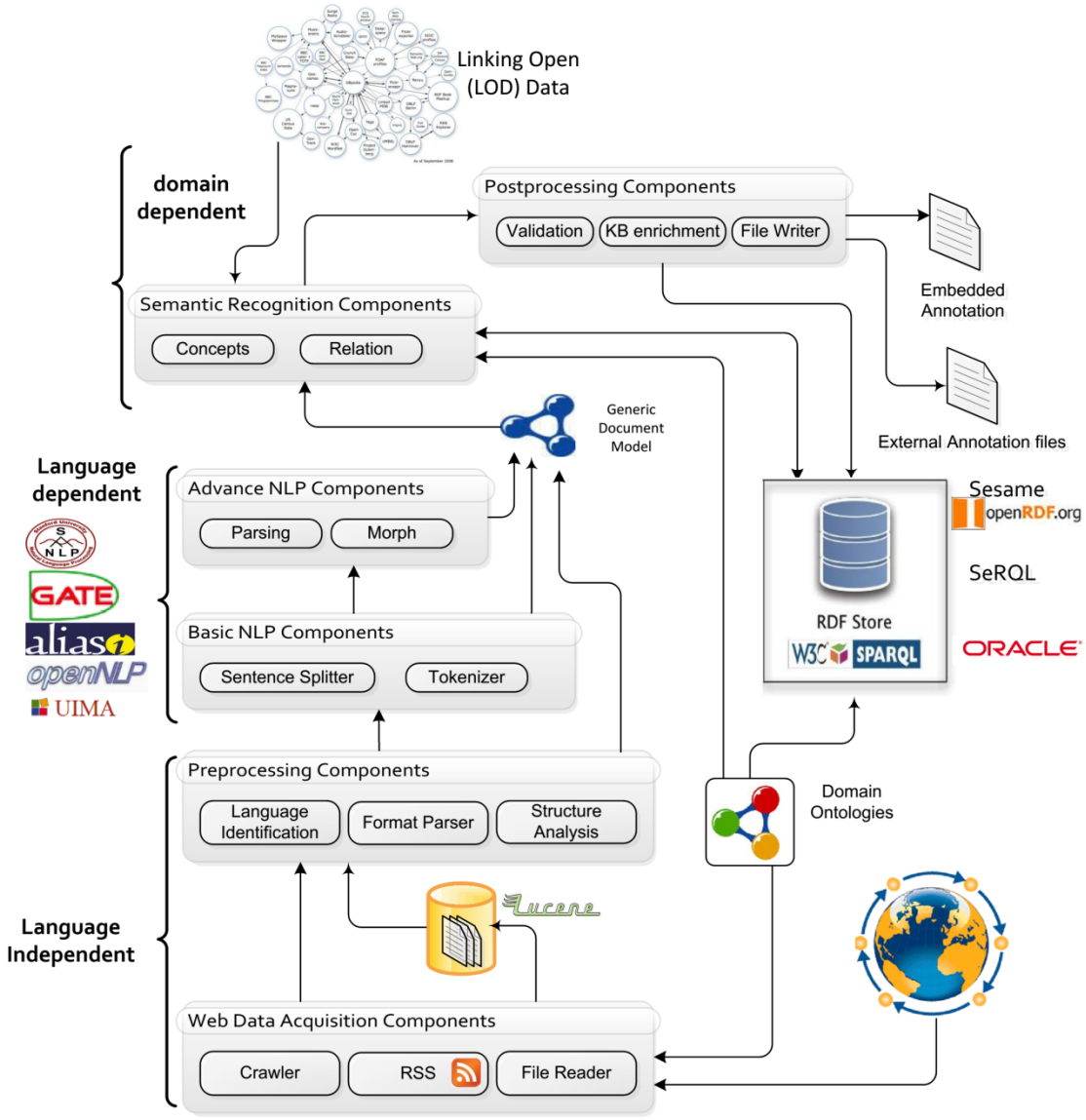



**Figure 4.1 High-level View of MLSAF Architecture**

## 4.3 Framework Components

The MLSAF framework consists of six layers, and each layer has multiple components. The first layer is Web data acquisition, which is composed of a crawler and other components to collect data from Web sources. The second layer is preprocessing, which is mainly composed of document format parsers, structural analyzers, and dominant

language identification. The third and fourth layers contain basic and advanced NLP processing components. The fifth layer is the core semantic annotation, which is composed of concept and relation recognition components. The last layer is postprocessing, which is responsible for the validation and storage of the annotation into persistence space. The language-dependent layers are only third and fourth, whereas the others are language-independent layers. The preprocessing and NLP layers will feed the recognition layer with a generic document model of Web documents. The recognition layer can be seen as the domain-dependent part of the framework. A more detailed explanation about each layer will be presented next.

### 4.3.1 Web Sources Acquisition Layer

In order to be able to annotate Web resources, acquisition of those resources is a requirement of any annotation system. A MLSAF framework contains a dedicate layer with a function to collect the Web resources related to the target domains. The collection is done using focused crawling guided by domain ontologies [27]. For a website that publishes RSS feeds for any updates, an RSS feed receiver is used to retrieve the new updates and to push them through the annotation pipeline.

#### 4.3.1.1 Focused Crawler

In order to acquire Web data, it is necessary to have a component to collect relevant resources from the Web in an automatic and efficient manner. The common approach is to use a tool called a crawler for this task. The crawler can be restricted to collecting Web resources related to set of domains of interest. The common name for such crawler is focused crawler. The advantage of using a focused crawler is to avoid processing Web pages that are unlikely to be valuable (irrelevant), which will ultimately minimize the

crawling time and effort. Our focused crawler workflow is shown in Figure 4.2. The crawling processing starts with an initial set of URLs and stores collections of Web resources into a document index, such as a Lucene index. The document index is to be used by the annotation pipeline tasks as an input. The crawler engine uses domain ontologies to measure relevancy of a given page to the domain represented by the ontologies.

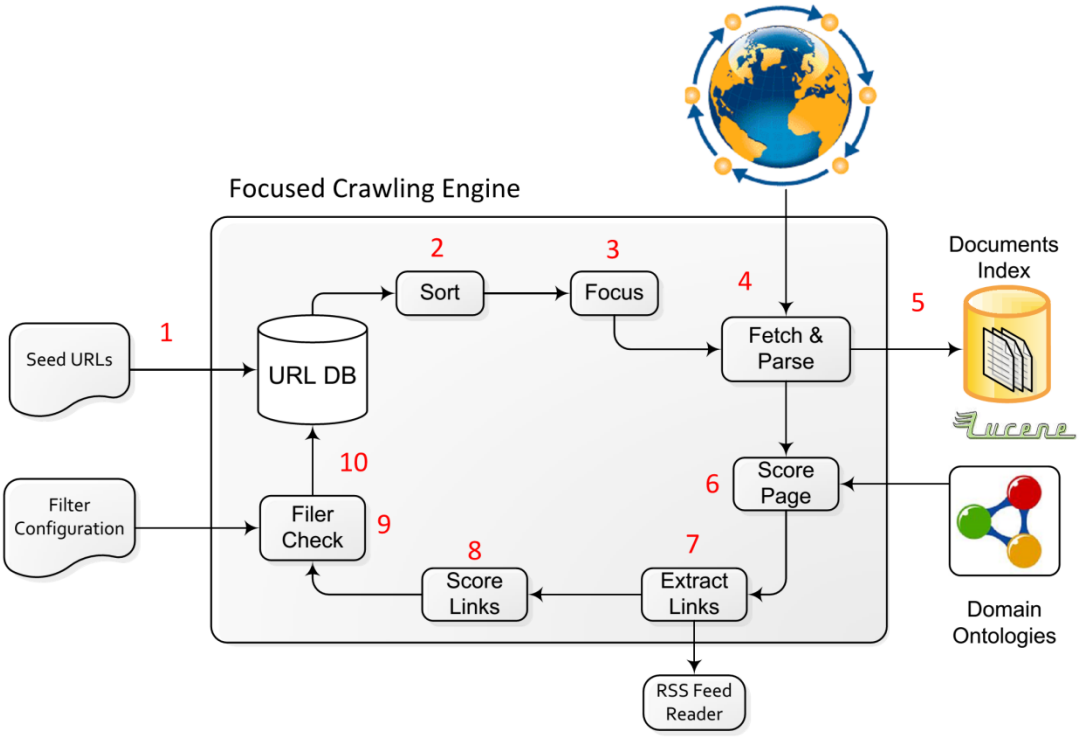



**Figure 4.2 Focused Crawling Engine**

Next, we are going to simply explain the steps of the focused crawler process, and the implementation details will be presented in the implementation chapter.

1. In the first step, the seed URLs are loaded onto the URL database. The seed URLs can be manually provided or acquired from directory service Web sites such as Alexa.[13]
2. The first loop can start after loading the initial URL from the seed list. In each loop, the first loop starts with the extraction of unprocessed links and then sorts the URLs based on the link score. The score of a URL is calculated based on the sum of the scores of the pages that contain a link to this URL. The page score is divided equally to contain all URL in the page.

$$Score(URL_i) = \sum_{j=1}^{n} \frac{Score(Page_j)}{linkCount(Page_j)} \quad (4.1)$$

where n is equal to the total number pages found pages containing $URL_i$

3. In the focus step, a set of URLs is selected to be processed for the loop-based criteria, such as minimum URL score or simply a fixed number of URLs.
4. The selected URLs are then fetched, considering efficient and polite fetching. The fetched pages are then parsed to extract textual context for indexing.
5. The fetched pages are stored in a document index or in a local folder in the network.
6. The parsed page is then scored based on domain ontologies. The score measures how close the page to domain ontologies. Typically this is a value from 0.0 to 1.0, with higher scores being better.

[13] http://www.alexa.com

7. For the scored pages, each out-linked URL in the page is extracted. If one of the out-links is detected as RSS feed then the feed database will update it in RSS feed reader component.
8. The score for the page will be inherited by all out-links found in the page.
9. The extracted URL is checked based on filter configuration. URLs that do not meet the specified criteria will be filtered out. A common filter is trustworthiness, which is an important criterion for considering data from medical Web sites.
10. In the last step in each loop, the URL database is updated for each successful or failed fetch attempts, newly discovered URLs, and any update on existing URLs score.

The crawling loop continues until a constraint is met. A constraint could be based on total crawling time, for example, or number of pages fetched, number of URLs visited, or minimum URL scores.

The first novelty of our approach is the use of domain ontologies to measure the relevancy of a given Web page to the domains represented in given ontologies. Second, the update to feed the database is generated by the crawler engine when discovering new RSS features provided by a Web site. Third, each Web site is given a relevancy score based on the score of the pages crawled from this Web site. The score could be then used for raking and other purposes. The fourth feature is the flexibility to store the crawled Web resources in any indexing engine or folder in the local network. Finally, the filer check is an important feature added to the crawler engine that could be easily customized for specific domain or application needs.

#### 4.3.1.2 RSS Feed Reader

In order to support the dynamic nature of some Web sites, it is necessary to have a component to cope with the automatic updates of those Web sites. One common method used by many Web sites is the use of RSS feeds to publish updates in any dynamic pages.

**RSS** is used by Web sites to publish updates in site content. An RSS document could contain either a summary of the update or its complete content. RSS helps the annotation system to fetch the update of dynamic Web site without the need to recrawl the Web site in an automated manner.

A list of the URLs of the Web sites providing RSS service has to be provided as a configuration file. The list should contain a set of attributes for each feed, as shown by the table below. The workflow of RSS feed reader is shown in Table 4.1 .

**Table 4.1 Feeds configuration**

| | |
|---|---|
| Type | RSS or Atom (Atom does not work well) |
| Agency | Name of the agency |
| Name | Name of the feed/channel |
| MachineName | Machine name of the feed/channel |
| Category | Category for feeding |
| url | URL of the feed/channel |

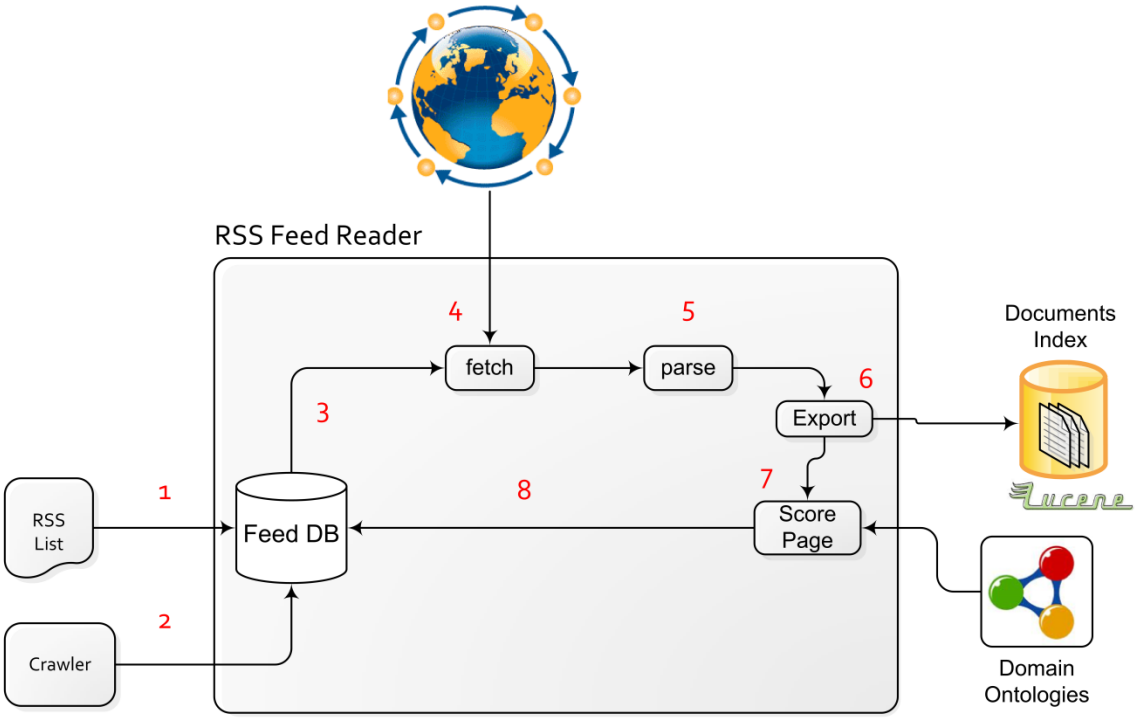



**Figure 4.3 RSS Feed Reader Engine**

Next, we are going to simply explain the steps of our RSS feed reader process, and the implementation details will be presented in the implementation chapter.

1. As the first step, an initial RSS item list is loaded to the feed database.
2. The RSS feed item can be acquired from the crawling process when a new one is discovered.
3. The feed scheduler selects the feed item to fetched.
4. The new feed item is downloaded from the selected Web site.
5. The downloaded feed item is then parsed using the HTML parser.
6. The fetched pages are exported into the document index.
7. The parsed page is then scored based on domain ontologies. The score measures how close the page is to domain ontologies. Typically, this is a value from 0.0 to 1.0, with higher scores being better.

8. The feed item entry gets updated for the next cycle.

The RSS feed reading loop continues until there are no new feeds from any feed items in the feed database. The process will remain idle until a new feed item is posted to the registered Web sites, the crawler finds a new Web site with RSS feed service, or the system's user adds a new RSS feed entry to the database.

The novelty of our approach in RSS feed reader is the use of domain ontologies to measure the relevancy of a given Web page to the domains represented in given ontologies. Furthermore, the update to feed database is generated by the crawler engine when discovering new RSS features provided by a Website.

#### 4.3.1.3 File Reader

The role of the file reader component is to be able to load the file stored in local network which could be accessed using file access protocol supported by either annotation server operating system or standard file transfer protocol tool such as FTP. Those files are either provided by a system user or collected from a Website using offline caching tool such as HTTrack.[14] This component is very helpful to test the annotation time independent from external network latencies.

### 4.3.2 Preprocessing Layer

Preprocessing is the first phase in the annotation pipeline after the acquisition is done. It is both language and domain independent phase. The aim of this phase is to prepare the document for processing by NLP and other tasks. It consists of document format parser, document structure analysis, and document dominant language identifier.

[14] http://www.httrack.com/

#### 4.3.2.1 Format Parser

In order to annotate a given document, textual information contained in the document needs to be extracted and presented to the annotation pipeline for processing. Web resource format parser is the process of parsing the document in order to extract textual as well as both semistructured and structured data from the Web resource. Web resources have different storage and serialization formats, such as HTML, XHTML, XML, PDF, and MS office documents. Additionally, the Web provides data in a form of Web services, database systems, and repositories. For each type of document format, there are different parsers with different capabilities.

Figure 4.4 shows the workflow of the document parser. The input for such a parser is a document collected from acquisition components and produces two outputs: one of the generic document model and the other an XHTML representation of the input document.

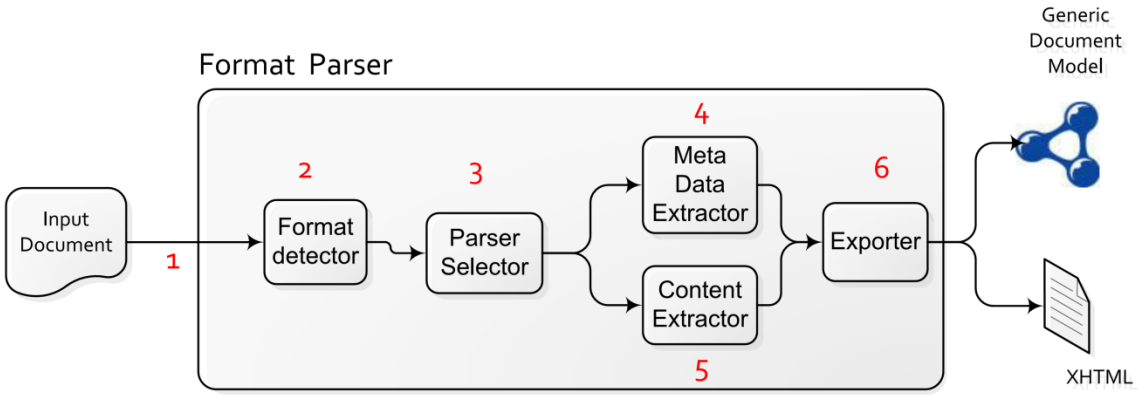



**Figure 4.4 Format Parser Workflow**

Apache Tika[15] is the most commonly used package, and it is capable of parsing all common document formats to extract textual content of the document as well as the metadata information of the document. It could be used for steps 2 through 5 in the proposed workflow. Customization is required to model the document into a generic document model. Tike provides an open interface to support new document format that is not included in the supported list.

#### 4.3.2.2 Structure Analyzer

The presentation of Web pages was customized for human consumption through the use of various presentation methods, such as navigation menu and decoration, which provide no useful information for the domain. Therefore, detecting the useful content in a Web page improves the performance of Web IE.

The role of the structure analyzer is to filter out irrelevant data in the source documents and separate the page into useful segments for processing. Web documents come with different structural styles; thus, segmentation of these documents needs to be performed for each content type in order to be able to annotate it.

The main steps of the structure analyzer component are shown in Figure 4.5. The steps of the analysis are as follows:

1. Once the document has been parsed into XHTML document, it can be opened using JDOM,[16] which enables the creation of a Java-accessible object representation of the fetched Web document.

[15] https://tika.apache.org
[16] http://www.jdom.org/

2. Starting from the DOM's root, it will recursively navigate through the DOM tree in order to extract the structural information.
3. The next step is to flatten the DOM tree hierarchy of all elements that are not structurally significant.
4. The next step is to identify the potential nodes that could be roots of a substructure, such as group of rows, paragraphs, and list elements.
5. The last step is to write the classified information into a generic document model that will be used in the next tasks.

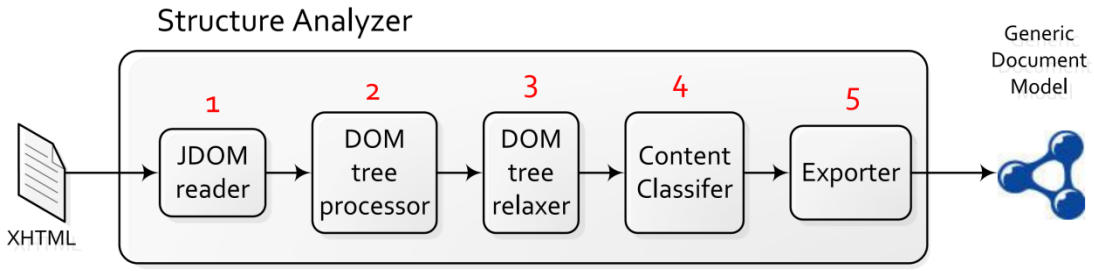



**Figure 4.5 Structure Analyzer Workflow**

This component will help the next tasks to focus on the part of the Web content that are concerned with such as textual, semi-structure and structure segments.

#### 4.3.2.3 Dominate Language Identifier

A prerequisite for any multilingual information processing system is the identification of the dominant language of the documents in a text corpus. It will help to select the right NLP processing toolkit suitable for processing the textual data. Failure to detect the right language will result in poor annotation results at the pipeline.

A common method to identify document language is based on the use of bigram frequency matrices [28]. The observation for every language is collected based on its own

orthographic characteristics that appear more frequently. A sequence of two characters is called a *bigram*. If a text contains several languages, paragraph or sentence separators could be used. A frequency table is generated that contains all possible bigram frequencies for one language. The frequency table is used by the language identifiers for all supported languages. Bigram frequency tables could be generated from good descriptive texts from every supported languages and domains.

In every language, there is a distinct frequency for each bigram that indicates its probability in each language. Let L be the set of languages supported by the language identifier, *ab* a bigram, and table $T_l$ the frequency table for bigrams in language *l*. The probability of *ab* belonging to language $l \in L$ is as follows:

$$P_l(ab) = \frac{T_l(ab)}{\sum_{k \in L} T_k(ab)} \tag{4.2}$$

The probability of a word *w* belonging to language *l* could be estimated as:

$$P_l(w) = \frac{P_l(\_w[1]) + P_l(w[|w|]\_) + \sum_{i=1}^{|w|-1} P_l(w[i]w[i+1])}{|w| + 1} \tag{4.3}$$

In equation (4.3), "_" stands for a word separator, |*w*| stands for number of characters in *w*, and $w[i]$ stands for the $i^{\text{th}}$ character in *w*.

If the text includes words injected from other languages, such as long words and proper names, then the performance of language recognition may be negatively affected. In order to resolve this issue, it is necessary to include context when trying to identify which language a word *w* belongs to. The context probability is weighted by the distance of each word from the word *w* using factor $1/(1 + x)$, where *x* is the distance in words. First, we need to calculate the cumulative probability of the $i^{\text{th}}$ word $w_i$ in the sentence or paragraph

and the weighted probabilities of the *s* closest neighbor words of $w_i$ to be of language *l*, as follows:

$$Q_l^s(w_i) = P_l(w_i) \sum_{i=1\ldots n}^{\lfloor s/2 \rfloor} \left( P_l(w_{i-j}) + P_l(w_{i+j}) \right) \times \frac{1}{1+j} \tag{4.4}$$

Next, we need to normalize these cumulated values in order to obtain the heuristic probability of the $i^{\text{th}}$ word in the sentence or paragraph $w_i$ for language *l* and the window size *s*, $P_l^s(w_i)$, as shown in the following equation:

$$P_l^s(w_i) = \frac{Q_l^s(w_i)}{\sum_{k=1}^{|L|} Q_k^s(w_i)} \tag{4.5}$$

Next, we decide on the language a paragraph or sentence belongs to by merging the probabilities of all words in the paragraph. If we let *p* be a paragraph with *n* words, the probability that *p* belongs to language *l* could be calculated as follows:

$$P_l(p) = \frac{\sum_{i=1}^{n} P_l^s(w_i)}{n} \tag{4.6}$$

Finally, the dominant language of a paragraph is the one with the highest probability $P_l(p)$. The dominant language of whole document is the one with the highest probability of most of its paragraphs.

### 4.3.3 Natural Language Processing Layers

NLP for textual data is performed to provide input for recognition tasks, such as concept and relation recognition. This component is a language-dependent component that depends on the system's supported languages. It includes basic NLP tasks such as tokenization, token normalization, sentence splitting, part-of-speech tagging, phrase

chunking, coreferences, and stemming. It also includes a set of advanced NLP tasks such as morphological analysis, parsing, discourse analysis, key phrase extraction, semantic role labeling, and polarity analysis. For speed performance, the advanced NLP tasks will only be used according to demand—for example, when complicated sentences are encountered and a naïve approach fails to discover relationships between domain entities. The performance of the annotation system depends on the performances of those tools. The availability and maturity of NLP tools varies between languages. For common Latin languages,[17] there are many available NLP tools with excellent performance that are easy to integrate and use in many environments and applications [29], [30]. However, there are a lot of challenges when dealing with less researched languages such as Arabic [31].

#### 4.3.3.1 Basic NLP Processing Layer

The basic NLP tasks are lightweight tasks that are fast and available for most languages. The execution time of those tasks is not a bottleneck in any textual information processing application. The basic NLP tasks include tokenization, token normalization, sentence splitting, part-of-speech tagging, phrase chunking, coreferences, and stemming. Those tasks for common Latin languages are provided by most common NLP toolkits such as Apache OpenNLP, GATE NLP, LingPipe, and Stanford CoreNLP. Tasks for other languages can be either be easily developed or integrated into those common NLP toolkits.

**Tokenization** is the task of splitting a text into words, numbers, symbols, or other elements called *tokens*. Tokens act as an input for other processing tasks such as chunking and parsing. It is a prerequisite for most linguistics processing tasks. Tokenization is

[17] such as English, German, French and Spanish

straightforward for languages that use spaces to separate words. For most Western European natural languages, tokenization is done using common rules that process tokens surrounded by explicit separators such as spaces and punctuation. On the other hand, for languages written in continuous script, such as Ancient Greek, Chinese, or Thai [32], tokenization is more difficult because there are no explicit boundaries between words. For specific domains, such as the biomedical field, additional customization in each language may be needed due to the use of special symbols related to that domain [33].

**Token normalization** is the process of converting strings that mean the same thing to one common form by correcting some common typing mistake, removing some adjustment to lettering, removing accents and umlauts, and normalizing abbreviations. The appearance of the same words with different spellings is common in Arabic, and each spelling refers to one unique word. The variation of spellings can be caused either by the writer's stylistic choice or by the writer's neglect of complex Arabic orthographic rules. The MLSAF framework supports both approaches with the possibility for extension based on domain requirement.

**Sentence splitting** is the process of dividing a given text into fragments based on predefined boundaries. There are two common approaches to detect the boundaries between sentences in a given text. One common sentence splitting approach is based on the use of finite-state transducers in cascade. It uses an abbreviation list to distinguish sentences that mark with full stops from other kinds. Another common splitting approach is based on the use of patterns of regular expressions. Both approaches are supported by the MLSAF framework, with the possibility to change rules in the first approach and patterns in the second approach. A hybrid between the two approaches is supported also.

Domain-specific sentence splitters, such as GENIA Sentence Splitter (GeniaSS) [34], which is optimized for biomedical texts, can be used by the MLSAF framework application.

The **part-of-speech (POS) tagging** method involves the task of marking up a word with its corresponding type based on the token and its context. There are three common approaches for POS tagging: rule-based [35], transformation-based [36], and stochastic (probabilistic)-based tagging [37]. For common Latin languages,[18] there are POS taggers based on all three approaches. The MLSAF framework supports the three approaches, with rule-based as the default.

**Phrase chunking** is the task of dividing a sentence into nonoverlapping chunks of tokens called *phrases*. Noun phrase (NP) chunking is a subtask for recognizing chunks of noun-related tokens. Verb phrase (VP) chunking is a subtask for recognizing chunks consisting of verb-related tokens. Chunking is an important task for recognition tasks such entity and relation recognition. There are two common approaches for chunking: rule-based and ML-based [38]. The rule-based approaches make use of POS tags and rules to generate chunks. The ML-based approaches are trained models with manually tagged text [38]. Most of the common NLP toolkits provide domain-independent NP and VP chunkers. Both approaches are support by the MLSAF framework.

**Coreference resolution** is the task of finding all expressions that refer to the same entity in a text. It is an important step for a lot of higher-level NLP tasks that involve natural language understanding, such as document summarization, question answering, and IE.

[18] English, German, French, Spanish, Italian

The approaches of coreference resolution can be categorized into two categories: linguistic approaches and machine learning approaches [39].

**Stemming** is the task of removing inflectional and derivational parts of a word to a common base form called the *stem*. The most common English stemmers are the Porter, Lovins, Paice/Husk, and Snowball [40]. The MLSAF framework has an open interface to support all of those stemmers. The Porter is the default stemmer for English, and Khoja [41] is the default stemmer for Arabic.

The MLSAF framework provides an interface to integrate any of the common NLP toolkits listed above. Additional customization is possible for the domain-specific requirements on those tools.

#### 4.3.3.2 Advance NLP Processing Layer

The advanced NLP tasks are heavyweight tasks that are expensive and not available for all languages. The execution time of those tasks is a bottleneck in any textual information processing application. The advanced NLP module includes tasks such as morphological analysis, parsing, discourse analysis, key phrase extraction, semantic role labeling, and polarity analysis. Some tasks for common Latin languages are provided by most common NLP toolkits, such as Apache OpenNLP, GATE NLP, LingPipe, and Stanford CoreNLP.

### 4.3.4 Document Model Processing

The capabilities and characteristics of annotation systems could be greatly affected by corpus representation. Using graph-based representations of the Web document offers several advantages [42]. First, it permits uniform expression of the arbitrary links between different subgraphs such as grammatical links, coreference links, and HTML formatting–

related links. Moreover, it permits the fast prototyping of new algorithms, provided that all potential features can be stored in the representation and handled.

Document ontology defines a document as a graph with nodes that represent document units (DUs). Part of the ontology is shown in Figure 4.6. It defines types of DUs and the relationships among them. The first type is the content unit (CU), which represents units of raw content in digital format in either discrete or continuous form. Images and text are examples of discrete CUs, and continuous CUs contain media such as audio and video. The second type of DU is the knowledge unit (KU), which represents CUs that aggregate and add navigation among them (e.g., paragraph, section, slide, and table). A paragraph that consists of several text fragments and graphics ordered in a given order is an example of the KU. Moreover, the document ontology defines properties that link annotation entities (i.e., instances, concepts, properties) from domain ontologies to DUs.

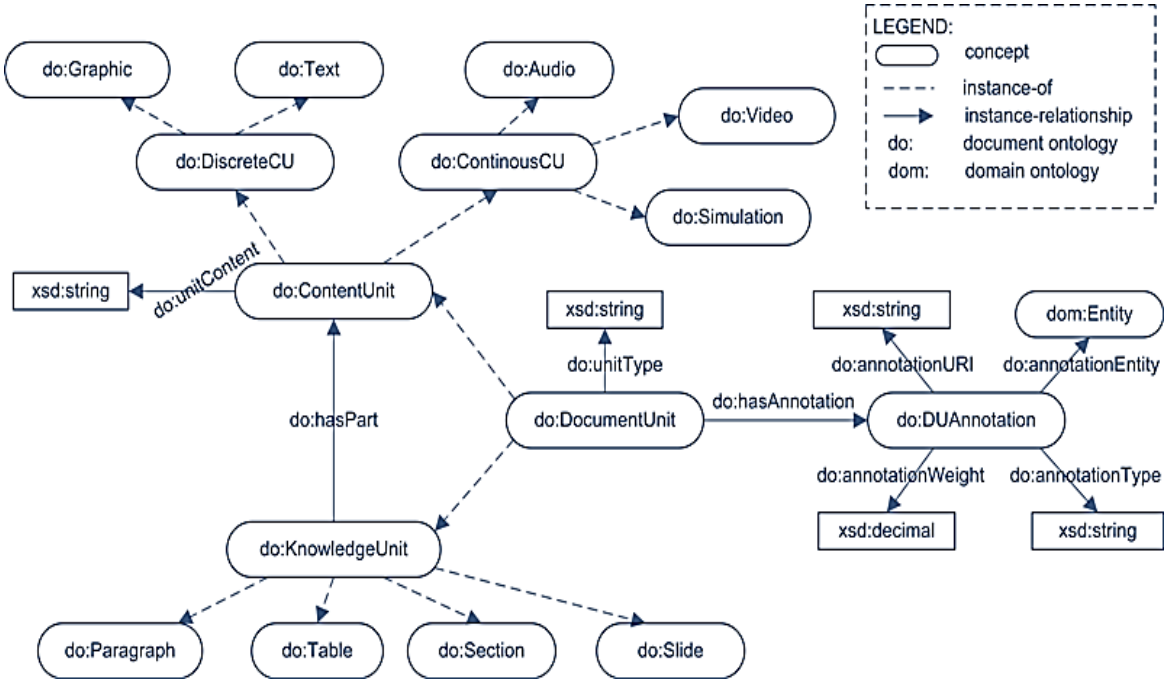



**Figure 4.6 A simplified example of the proposed document representation**

### 4.3.5 Recognition Layer

The recognition layer contains the core components of the semantic annotation process. It links the entities and their semantic relationships discovered in the Web document with domain ontologies. The linking is done in two steps, by starting with entity recognition and then relationship between them. The two tasks will be demonstrated in the next two sections.

#### 4.3.5.1 Domain Entity Recognition

Domain entity recognition is the process of identifying and classifying atomic texts into predefined domain ontological classes. The recognition process starts first with a discovery step followed by a resolution step. The discovery detects which parts of the text refer to ontological concepts and could be performed using different approaches, such as a gazetteer (dictionary-based), rule-based, or machine-learning approach [38]. The resolution matches the detected entity with the most likely ontology concept. The resolution could be handled by a classification method that considers the context of the matched phrase. Different algorithms have been used in this task, as shown in [43]. The focus for the MLSAF framework is to have a language-independent approach to handle both tasks. Next, we will elaborate on the entity recognition using dictionary matching which is a language independent approach.

**Dictionary matching approach:** A simple and efficient approach is to use a dictionary built from all domain ontologies augmented with synonyms. The synonym for each name of the ontology concepts could be generated using publicly available multilingual resources such as WordNet and Wikipedia. The resolution step is performed to select the right concept from multiple options presented by the discovery step. This step could be

resolved by applying a disambiguation, which could be done using one of the existing approaches such as context space vector distance [44] or surface form frequency [45].

The entity recognition task starts with text that may contain a phrase that corresponds to an ontology concept. The general procedure for dictionary-based entity recognition could be broken down into three stages, as shown by Figure 4.7.

1. **Matching**: identification of textual phrase (surface form) that could refer to an ontology resource.
2. **Entity resolution**: the identification of the most likely ontology resource in case multiple candidate entities are returned from matching step.
3. **Filtering**: removing the nonessential information from the two steps according to a configuration provided by the user/application.

After the whole procedure is completed, the output of this process is a set of links between fragment of input text and ontology resources.

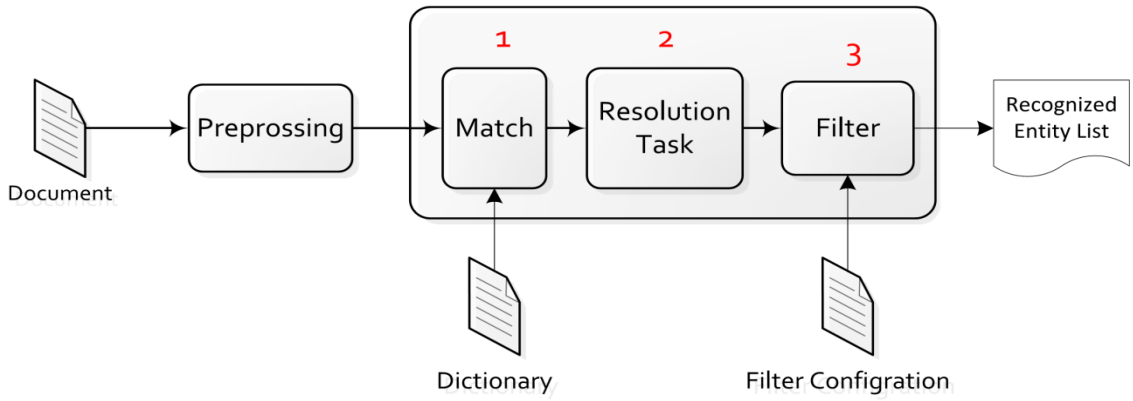



**Figure 4.7 Dictionary-Based Entity Recognition**

The next four algorithms demonstrate the dictionary-based entity recognition. The entity recognition is based on domain ontologies using string matching. The matching is done

using a dictionary of ontology resources with their synonym retrieved from lexical resources such as WordNet. To have efficient access to the dictionary element, finite state automata are usually used as data structure [46].

The first algorithm demonstrates how the dictionary is constructed from a given set of ontologies for a set of languages. A quality check on synonyms is done before inserting them into the dictionary. Such check could validate the entry based on the string length, content and special characters.

***Algorithm*** 1 Dictionary Builder

```
Input:  OntologySet O, AnnotationProperty P, LanguageSet LS, ResourceType RT
Output: Dictionary D

1     For each Resource r ∈ (O ∩ RT) do
2       For each Language l ∈ LS do
3         String lp      = getAnnotationValue(r,P,l);
4         String nr      = normalize (lp);
5         URL Key        = getURL(r);
6         Set synonymSet = GetSynonymSet(nr,l);
7         D.add(Key,nr);
8         For each synonym ∈ synonymSet
9             ns = normalize(synonym);
10            if ns.type = stopWord  or IsEmpty(ns) then
11               skip ns;
12            else
13               D.add(Key,ns);
14            end if;
15        end for;
16      end for;
17    end for;
18    return D;
```

The second algorithm demonstrates the entity discovery, which is performed using the dictionary. To control the precision, chucks with lower scores than the given threshold are filtered out in order to remove any noisy detection.

The third algorithm describes the resolution step, which is performed to select the right concept from multiple candidate entities found in the discovery step. This step could be resolved by using a disambiguation method such as distance between contexts of each

candidate ontology concept and text phrase. The overall entity recognition approach is described in fourth algorithm, which makes use of the dictionary, entity discovery, and entity resolution functions.

***Algorithm*** 2 Entity Discovery

```
Input: Document D, Dictionary C, Threshold T
Output: Set of candidate entity e ∈ C in document D

 1    ChunkSet E     = Chunk(doc,C);
 2    ChunkSet final = [];
 3    foreach entity e   E do
 4    if e.score < T then
 5           skip e;
 6    else
 7      final.add(e);
 8    end if;
 9    end for;
 10   return final;
```

Given a set of documents, a set of domain ontologies, and configuration parameters, the entity recognition task could generate an annotation for the entities discovered and resolved to closest domain concept.

***Algorithm*** 3 Entity Resolution

```
Input: Document D, ChunkSet CS, OntologySet O, Confidence c
Output: disambiguated set entities DCS in document D
 1      ChunkSet processed = [];
 2      ChunkSet final     = [];
 3      foreach chunk e1 ∈ E do
 4           if processed.found(e1) then
 5              continue;
 6           else
 7             processed.add(e1);
 8           ChunkSet os = findOverlapChunk(e);
 9           foreach chunk e2 ∈ os do
10             processed.add(e2);
11             if e1.URI = e2.URL then
12               final.add(longestSpan(e1,e2));
13             else
14               Context c1   = getContext(D,e1);
15               Context c2   = getContext(D,e2);
16               Context oc1  = getContext(O,e1);
17               Context oc2  = getContext(O,e2);
18               Score s1 = similarity (c1,oc1);
19               Score s2 = similarity (c2,oc2);
20               if s1<c AND s2<c then
21                  continue;
22               else if s1 > s2 then
23                       final.add(e1);
24                    else
25                       final.add(e2);
26               end if;
27             end if;
28      end for;
29      return final;
```

The entity recognition task generates a list of the discovered entities with some recognition information such as score and URL. This information is used for relation recognition and other tasks. This task could also add a reference to publicly available LOD[19] resources for each discovered entity. This option could help the semantic annotation application to have access to more information on annotated entities, such as photos and metadata.

[19] For example, DBpedia, freebase, Yago.

***Algorithm*** 4 Entity Recognition

**Input:** Set of documents D, *OntologySet O = {$O_1$..$O_n$}*, *Configuration C*
**Output:** Set of recognized entities $e_{jk} \in O$ for each document $d_j \in D$

```
1     Threshold  T  = C.getThreshold()
2     Confidence CF = C.getConfidence()
3     Filter     FL = C.getCustomFliter(;
4     AnnotationProperty AP  = C.getAnnotationProperty();
5     LanguageSet        LS  = C.getLanguageSet();
6     List Resource rl = [];
7     Dictionary OD;
8     OD = DictionaryBuilder(O,AP,LS,{Classes|Individuals});
9     foreach document dj ∈ D do
10      ChunkSet candidateSet  = EntityDiscovery(dj,OD,T);
11      ChunkSet finalSet = EntityResolution(dj,candidateSet,O,CF);
12      foreach chunk in finalSet
13         Resource rs = chunk2resouce(chunk);
14         if Filter(rs,FL) then
15            continue;
16         else
17            rl.add(rs);
18         end if;
19      end for;
20    end for;
21    return rl;
```

The advantage of using this approach is both domain- and language-independent with an acceptable recognition performance when applied to some domains. This approach is applicable for entities with a finite number of instances, such as names of cities, counties, or fruits.

**Rule-based entity recognition approach:** Another approach for domain entity recognition is based on rules. Those rules might be either manually crafted or use a semiautomatic approach [38]. The MLSAF framework supports this kind of approach by providing a facility to describe those patterns over the output of NLP processing tasks. Some generic rules are predefined, such as subclass and equivalence, as shown in Table 4.2

**Table 4.2 Some Generic Rules for Entity Recognition Task**

| Rules Set | Sample rules (English) | Example |
|---|---|---|
| Subclass | NP <subclass> be NP <superclass><br>NPlist<subclass> be CN NP<superclass><br>NP<superclass> CATV CV? CN? PUNCT? NPlist<subclass> | Apple and orange are kind of fruit. |
| Equivalence | NP<class> be (equal to\|the same as\|like\|similar) NP<class><br>NP<class> (denominate\|call\|as\|designate by\|name) NP<class> | Star Apple is also called Caimito |

For each entity category, users can set a rule based on NLP information, generic rules, and self-developed rules in sequential order. The next step is to use the entity resolution described in the third algorithm, which is performed to select the right concept from multiple candidate entities found in the rule application step.

The workflow steps in the rule-based entity recognition are shown in Figure 4.8. The engine takes the processed document from the NLP task and produces a list of recognized entities.

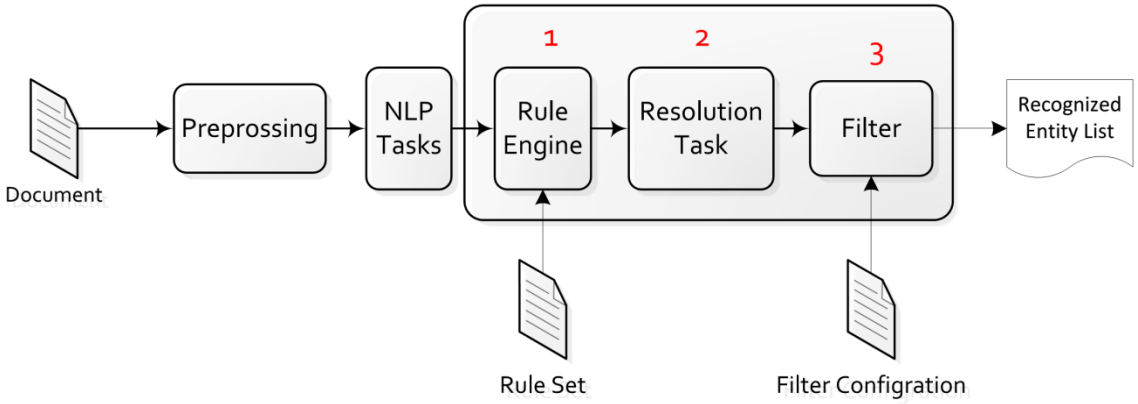



**Figure 4.8 Rule-Based Entity Recognition**

This approach is both domain- and language-dependent. The developer has to provide the rules set for each language. This approach is applicable for entities with names that follow specific sets of patterns, such as Latin people names, job titles, and so on.

**Machine learning approach:** Entity recognition using an ML-based approach makes use of statistical models built on a feature representation of the observed entity data. This ML approach handles some problems of rule- and dictionary-based approaches, such as the recognition of new name entities and spelling variations in name entities. However, ML does not provide URI for ontology resources, which can be solved by using a resolution step in the dictionary approach. Nonetheless, the core limitation of this approach is the need for annotated documents, which may be difficult or costly to have. Therefore, the lack of such resources for a specific entity type may limit the application of the ML approach for entity recognition tasks. The adaptation of the ML-based approach requires two steps, as shown in Figure 4.9: train and recognize. In the training step, the ML model must be presented with annotated documents. The time complexity of this step depends on the size of the model and on the available computational power. After the training model is developed, new documents can be annotated using the model for the general entity category, and entity resolution needs to be applied to get the URI of ontological resource.

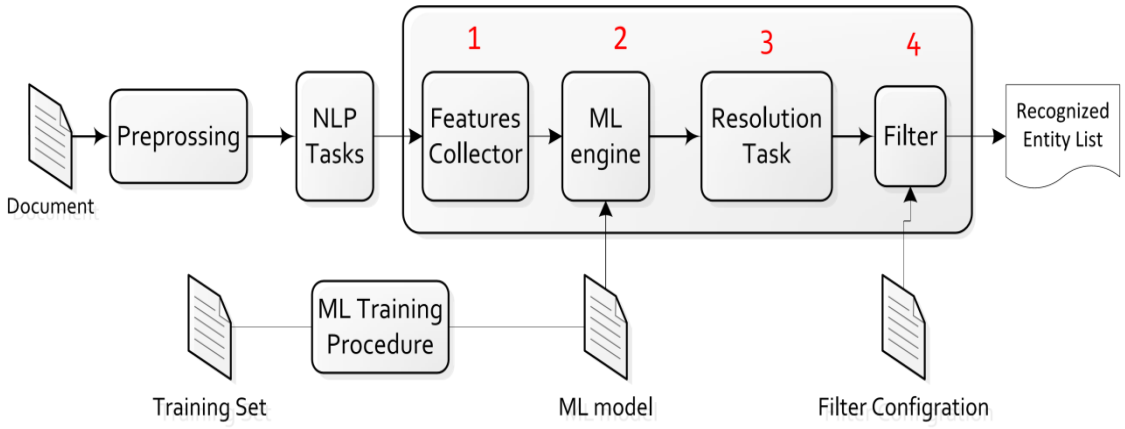



**Figure 4.9 Machine Learning Entity Recognition**

**Entity recognition using ensemble approach:** To achieve the best of all approaches, one possibility is to combine of the results several recognizers, each with different

characteristics. For ML, different models could be generated that reflect different characteristics of the annotated data through the use of different feature sets or parsing directions (forward and backward). Furthermore, different entity recognition approaches can be combined between ML approaches, dictionary-based approaches, and rule-based approaches [47]. MLSAF supports different integration options such as union and intersection. Another combination method is to use an ensemble between ML approaches, based on common ensemble methods such as the majority vote [48] or the memory-based learner [49]. The details of the two ensemble methods will be shown next.

**Ensemble using majority voting:** Because there is a large number of features, it is not practical to put them all into one ML model. Therefore, several ML classifiers can be trained, each with different feature sets, in order to capture as many features as possible. Then, the following simple voting can be used to combine the results of the ML classifiers:

$$S(y,x) = \sum_{i=0}^{T} C_i(y,x) \tag{4.7}$$

where $S(y,x)$ stands for the score of a label *y* and a character *x*; *T* stands for the number of ML models; and the value of $C_i(y,x)$ is 1 if the decision of the result of the $i_{th}$ ML model is *y*; otherwise, it is zero. After processing all different options in *y*, the highest score of *y* is chosen to be the entity category for *x*. Figure 4.10 shows the majoring voting workflow with the input of the entity to be classified.

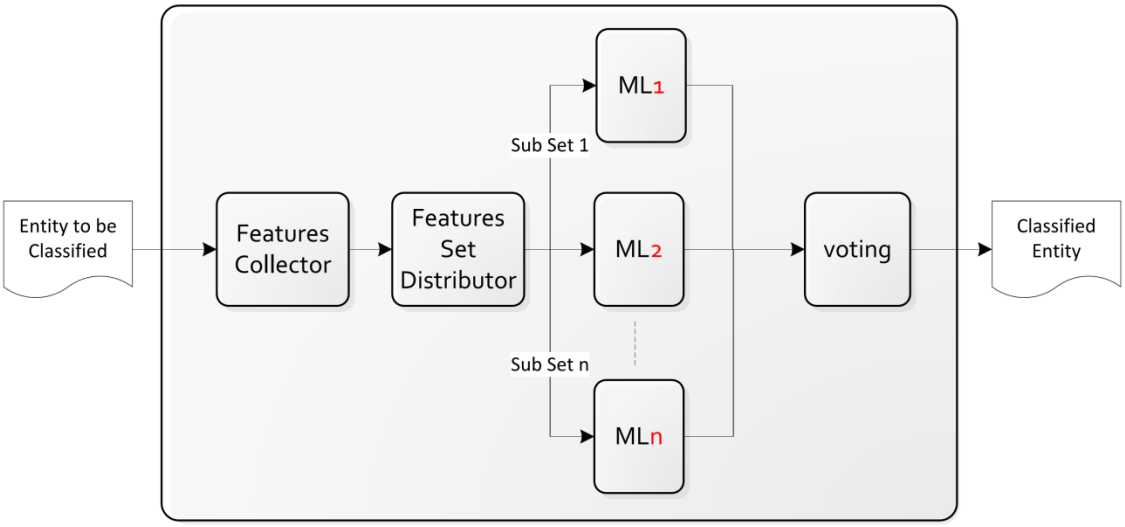



**Figure 4.10 Majority Voting Ensemble of Entity Classification**

**Ensemble using memory-based learner:** The memory-based classifier during the training phase remembers all given examples. If a new word is presented, it searches in the k-nearest neighbors to find the most similar example as the output. In the memory-based combination method, the classifier memorizes all named entities from the results of the various classifiers and then tags the tokens that were originally tagged as "Other." For example, if a word *x* is recognized by one classifier as "0" ("Other" tag), and if the memory-based classifier learns from another classifier that this word is recognized as FOOD, then *x* will be recognized as "B-FOOD" by the memory-based classifier. The obvious drawback of this method is that the precision rate might decrease as the recall rate increases. To overcome this drawback, a set of three rules could be used to filter out samples that are likely to have a high error rate.

1. Not to allow different named entities to be recognized by different classifiers.
2. To filter out examples that have an error rate less than an absolute frequency threshold.

3. To filter out examples with an error rate less than a relative frequency threshold.

The developer can select any of the options available for entity recognition, based on the domain and degree of the accuracy required.

#### 4.3.5.2 Relation Recognition

Relation recognition is the process of discovering relationships between two or more entities based on the types of relationships defined in the ontology. Based on the data source format, different recognition approaches are followed. For structured content such as tables, table headers are mapped to concepts or concept attributes, and rows are mapped as instances of concepts or values of attributes [18]. For unstructured content like paragraph text, rule-based, pattern matching, and ML approaches are commonly followed [50]. An elaboration of MLSAF support for those approaches is demonstrated next.

We classify relations between entities into two main categories based on the complexity of the relationships: simple, and complex [51]. The simple relationship is expected to have a single verb connecting two entities, which may or may not have a modifier. It is not expected to have internal clauses, implicit dependencies, or multiple subjects or objects.

**Relationship recognition using trigger words.** Relationship discovery using trigger words is a naïve approach for relationship detection between name entities in a sentence. It is considered a simple approach that does not require even minimal NLP processing and depends heavily on string matching. The precision of this approach is considered to be low, and recall could be high if enough trigger words are considered. The performance of

this approach mainly depends on the generation mechanism of the trigger words with variations of spellings and synonyms.

The fifth algorithm demonstrates the process relationship recognition using the trigger word. It starts by building a dictionary of object properties in the input ontologies, then uses this dictionary to delete the sentences containing trigger words in the dictionary. It builds up relation tuples if the sentences contain domain and range entities of a relationship. Finally, the validity of the generated tuple is checked with respect to the domain ontologies. Figure 4.11 shows the workflow of the trigger word–based relationship recognition.

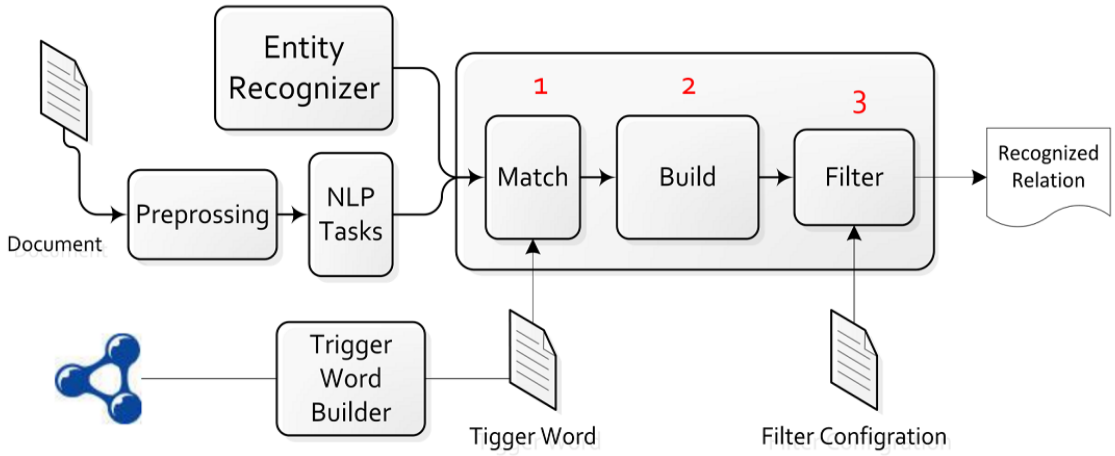



**Figure 4.11 Trigger Word-Based Relation Recognition**

***Algorithm*** 5 Relation Recognition Using Trigger Words

**Input:** Annotated documents D, *OntologySet O = {$O_1$..$O_n$}*, Configuration C
**Output:** Set of relation tuples $r_{jk} \in R$ for each document $d_j \in D$

```
1    Threshold  T  = C.getRelationThreshold()
2    List ObjectProperty OPL = getObjectProperty(O);
3    List PropertyLabels PL  = getLabels (OPL);
4    List TriggerWords   TW  = generateVaraions(PL  );
5    Dictionary OD;
6    RD = DictionaryBuilder(O,AP,LS,{ObjectProperty});
7    foreach document dj ∈ D do
8      NE = ExtractNamedEntites(si);
9      ChunkSet CR = RelationDiscovery(dj,RD,T);
10     if (NE ≠ Θ) && (CR ≠ Θ) then
11          foreach cr ∈ CR do
12             DomainList = ExtractEntites(cr.domain,cr,si);
13             if (DomainList.count)= 0 then
14               skip sentence si;
15           else
16             RangeList  = ExtractEntites(cr.range,cr,si);
17             if (RangeList.count)= 0 then
18                  skip sentence si;
19               else
20                   foreach domain entity ed ∈ (DomainList) do
21                      foreach r entity er ∈ (RangeList) do
22                        E = GenerateTuple(ed; cr; er);
23                        rjk = Aggregate(E, rjk);
24                      end for;
25                   end for;
26               end if;
27        end for;
28     else
29        skip sentence si ;
30     end if;
31   end for;
```

**Relation recognition using extraction patterns:** One common method of relationship extraction uses patterns to extract the relation based on the matches of certain tokens in a given context. In the pattern, the token is variable whereas the surrounding context is constant. The context might contain set of tokens or a linguistic annotation, such as POS or syntax annotation. The sixth algorithm describes the relationship recognition based on patterns that are provided to the system. The pattern could be generated manually or harvested automatically from annotated resources. The algorithm checks for sentences that contain matched entities and patterns. If both are matched then a tuple is generated for the discovered relation.

***Algorithm*** 6 Relation Recognition Using Patterns

**Input:** Set of annotated documents D, *OntologySet O = {$O_1$..$O_n$}*
**Output:** Set of relation tuples $r_{jk} \in R$ for each document $d_j \in D$

```
1     List ObjectProperty OPL = getObjectProperty(O);
2     List Pattern         RP = genarteRelationPatterns(OPL);
3     foreach document dj ∈ D do
4       S = TokenizeToSentences(dj);
5       foreach sentence si ∈ S do
6          NE = ExtractNamedEntites(si);
7          CR = ExtractRelations(si,RP);
8          if (NE ≠ Θ) && (CR ≠ Θ) then
9             foreach cr ∈ CR do
10               DomainList = ExtractEntites(cr.domain,cr,si);
11               if (DomainList.count)= 0 then
12                 skip sentence si;
13               else
14                 RangeList  = ExtractEntites(cr.range,cr,si);
15                 if (RangeList.count)= 0 then
16                    skip sentence si;
17                 else
18                     foreach domain entity ed ∈ (DomainList) do
19                        foreach r entity er ∈ (RangeList) do
20                            E = GenerateTuple(ed; cr; er);
21                            rjk = Aggregate(E, rjk);
22                        end for;
23                     end for;
24                end if;
25            end for;
26         else
27            skip sentence si ;
28         end if;
29       end for;
30    end for;
31    return rjk ;
```

**Relationship recognition based on entity-predicate pair detection:** An advanced relationship recognition method is based on the parsing of a sentence and capturing subject, action, and object from the parse tree. Using dependency parsers to generate the parse tree is the first step. The advantage of using dependency parser tree shows a direct dependency between different components in a given sentence. A subject-action-object extractor is based on a set of rules to be customized for a given parser and language. These rules determine which part of the parser output is considered the subject, verb, and

object [52]. Figure 4.12 shows the workflow of the relationship recognition based on the entities and predicate detection.

For speed efficiency, this relationship recognition should be applied only for special cases because it relies on parsing, which is a time-consuming task, and it will have a huge impact on the extraction speed.

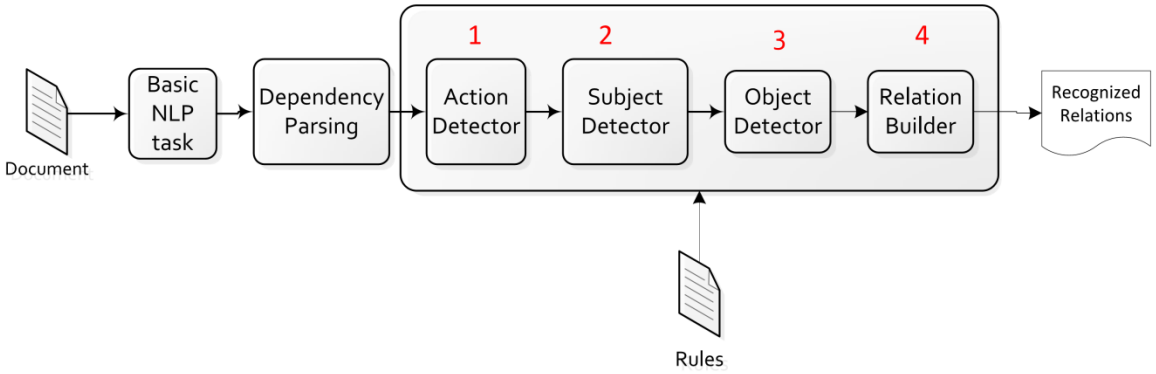



**Figure 4.12 Relation Recognition Using Entity-Predicate Pair Detection**

**Relationship recognition using ML:** Relationship recognition could also be done using ML methods. The problem can be modeled as a classification or sequence-labeling problem [50]. Classification is the most common method in textual analysis; this can be done using Perceptron, Support Vector Machine (SVM), Naive Bayes (NB), K-Nearest Neighbor (KNN), Maximum Entropy Model (MEM), and Decision Tree (DT) [53]. Optimal features need to be defined for the selected classifier, including sentence-level, entity-level, and token-level features.

**Table 4.3 Sample Feature Set for Relation Classifier**

| Category | Features |
|---|---|
| Entity features | types of $entity_1$ and $entity_2$ |
| | Token string of $entity_1$ and $entity_2$ |
| | word bigrams in $entity_1$ and $entity_2$ |
| | POS tags of $entity_1$ and $entity_2$ |
| Context features | words between $entity_1$ and $entity_2$ |
| | word bigrams between $entity_1$ and $entity_2$ |
| | POS tags between $entity_1$ and $entity_2$ |
| | distance between $entity_1$ and $entity_2$ |
| mix | concatenations of above features |

Conditional random Fields (CRF) and hidden Markov models (HMM) are the two common methods used as sequence-labeling approaches for relation recognition [54].

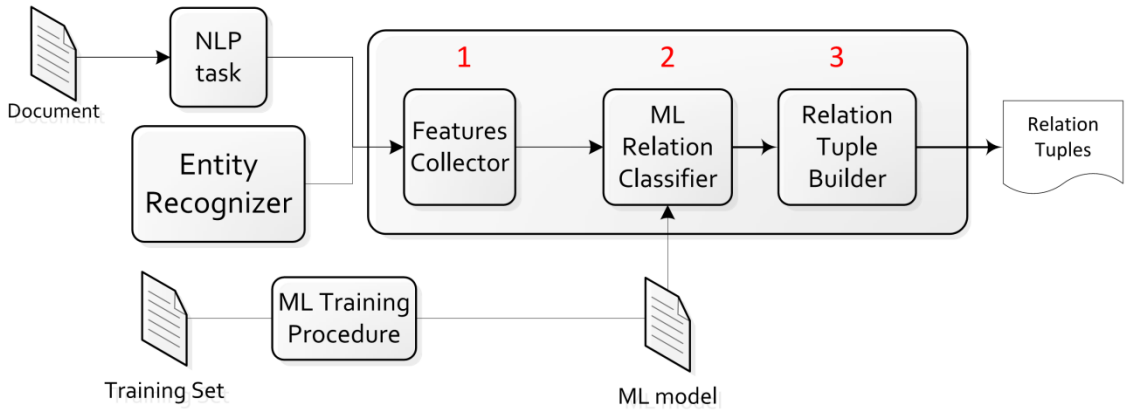



**Figure 4.13 Machine Learning Relation Recognizer**

Figure 4.13 shows the workflow of the relationship recognition using the ML approach. The feature set is collected at token, phrase, sentence, and entity levels for the ML classifier. The classifier will yield a classification for the type of relationship (if any)

between entities. The relation tuple has to combine entities in the form of <subject,relation,object> according to ontologies' object properties.

**Relation recognition using ensemble approach.** To achieve the best of all approaches, one way is to combine of the results several recognizers, each with different characteristics. Furthermore, different relationship recognition approaches can be used to combine the generated annotations between trigger word approach, patterns, entity-predicate predication, and ML approaches [55]. Another combination method is to use an ensemble between ML approaches based on common ensemble methods such as the majority vote [48] or the memory-based learner [49], similar to what was described in the entity recognition section. The details of the two ensemble methods will be shown next. In the ensemble of majority voting for relation recognition, several ML classifiers are used, each with a different feature set. Because the number of features is large, it is not practical to put all feature sets into one ML model. Then, the following simple voting can be used to combine the results of the ML classifiers. The majority voting approach for relationship recognition is shown in Figure 4.14.

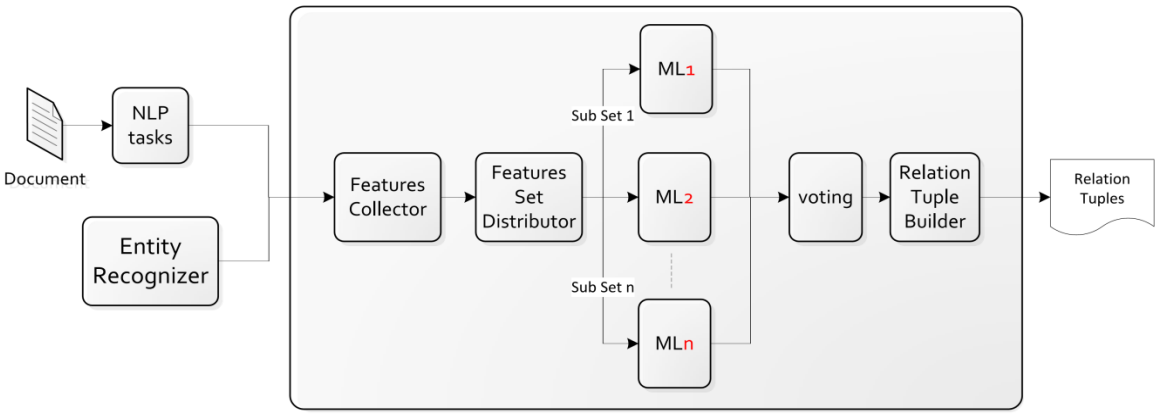



**Figure 4.14 Majority Voting Ensemble of Relation Recognition**

The memory-based classifier during the training phase remembers all given examples. If a new relationship exists in a sentence, it searches in the k-nearest neighbors to find the most similar or a close example as the output. In the combination method of memory-based classifiers, the classifier memorizes all relationships as a result of the various classifiers and then tags the tokens that were originally tagged as "Other." For example, if a relation *x* is recognized by one classifier as "0" ("Other" tag), and if the memory-based classifier learns from another classifier that this relation is recognized as PREVENT, then *x* will be recognized as "B-PREVENT" by the memory-based classifier. The obvious drawback of this method is that the precision rate might decrease as the recall rate increases. To overcome this drawback, a set of three rules could be used to filter out samples that are likely to have a high error rate.

1. Not to allow different relationships to be recognized by different classifiers.
2. To filter out examples with an error rate less than an absolute frequency threshold.
3. To filter out examples with an error rate less than a relative frequency threshold.

### 4.3.6 Post-Processing Layer

The first task of the postprocessing layer is to validate the recognized entities and relationships against the document, ontologies, and knowledge base. The second task is to validate information into persistence storage, either in files or repositories. The details for those two tasks will be elaborated next.

#### 4.3.6.1 Validation

In the validation task, the recognized entities and relations are validated against the domain ontologies and instances. Recognized entities are valid if there are corresponding

instances in the knowledge base to them. Similarly, the recognized relationships are valid if they exist in the domain ontologies and if the entity pair matches the type the domain and range of the object properties defined in the ontologies.

The validation approach of entities and the relationship of a tuple <subject,predicate,object> against a domain ontology with instances could be summarized in the following steps:

1. Search for instances that match subject and object.
2. If a match for both subject and object is found, then search for a match for the predicate in the ontology model.
3. If the three are matched, then the check is done for the class concepts to which the instances for subject and object can be match respectively in domain and range of the property matched.

If all these conditions hold, a new construct (instance for subject, property, and instance for object) is added to the set of validated constructs. The above conditions are captured formally in the validation criteria specified below.

**Validation rule:** For a text fragment *Z* consisting of sentences *{Si}* with word sets *{Wi}*, a set $T_{CTR}$ of candidate constructs extracted by a recognition algorithm, a domain, an ontology *O(R,C)*, a set of instances *I*, a function $h : I \longrightarrow P(C)$, and a mapping *F* from the set *W* to *R U I* that is able to type the words in the sentence to an instance in *I* or a relationship in *R* (whenever such a mapping is intuitive based on the domain of discourse), the validation process must result in a set $\{K = \{s_i,p_i,o_i\}\}$ of three tuples *si,pi,oi* (validated constructs) such that the following holds:

$$\{\exists y_{1i}, y_{2i} \in Y, \exists r_i \in R \quad | \quad (y_1, r_i, y_2) \in K \Leftrightarrow (\exists w_{1i}, w_{2i}, w_{3i} \in W_i, c_{1i}, c_{2i} \in C \,|\{w_{1i}, w_{2i}, w_{3i}\} \in T_{CRT} \cap (w_{1i}, y_{1i}) \in F \cap (w_{2i}, y_{2i}) \in F \cap (w_{3i}, y_{3i}) \in F \cap c_{1i} \in h(y_{1i}) \cap c_{2i} \in h(y_{2i}) \cap c_{1i} \in Domain(r_i) \cap c_{2i} \in Range(r_i)\} \tag{4.8}$$

#### 4.3.6.2 Annotation Writer

In the source and knowledge base enrichment task, the recognized entities of the Web documents are matched to the concepts and instances in the ontology and knowledge base. Then, the recognized set of relations between the concepts mentions are mapped to the object properties in the ontology. The matching of entities and relationships are annotated within the source document and external annotation files. The external annotation files could be used directly or stored in the semantic repository to be used by the semantic application.

The seventh algorithm demonstrates the standalone RDF generation of an annotated document by entity and relationship recognizer. It considers documents and sentences that contain related information to the domain ontologies.

***Algorithm 7*** RDF Generation

```
Input:  Annotated document C with named entitles NE and relation tuples RT,
        Ontologies O
Output: RDF format for D according domain Ontologies O

 1    D_RDF = NameSpaces(O);
 2    D_RDF = D_RDF  UImports(O);
 3    D_RDF = D_RDF  UGenerateTuple(D.id,rdf:type,O:Document);
 4    D_RDF = D_RDF  UGenerateTuple(D.id,O:hasURL,D.URL);
 5    foreach relation tuples rt ∈ RT do
 6      s = findSentence(D,rt);
 7      D_RDF = D_RDF  U GenerateTuple(rt,rdf:type,O:Relation);
 8      D_RDF = D_RDF  U GenerateTuple(s.id,rdf:type,O:Sentence);
 9      D_RDF = D_RDF  U GenerateTuple(s.id,O:hasContent,s.content);
10      D_RDF = D_RDF  U GenerateTuple(s.id,O:appearsIn,D.URL);
11      D_RDF = D_RDF  U GenerateTuple(rt,O:appearsIn,s);
12      D_RDF = D_RDF  U GenerateTuple(rt.domain,O:rt.relationType,rt.range);
13   end for;
14   S = TokenizeToSentences(C);
15   foreach sentence si ∈ S do
16     NE = FindEntites(si);
17     foreach entity ei ∈ NE do
18       D_RDF = D_RDF  U GenerateTuple(s,rdf:type,O:Sentence);
19       D_RDF = D_RDF  U GenerateTuple(s.id,O:hasContent,s.content);
20       D_RDF = D_RDF  U GenerateTuple(s,O:appearsIn,D.URL);
21       D_RDF = D_RDF  U GenerateTuple(ei,O:appearsIn,s);
22     end for;
23   end for;
```

#### 4.3.6.3 Knowledge Base Enrichment

The role of knowledge base enrichment tasks is to insert the recognized entities and relations between them as well as the source information into the knowledge base repository. This task needs to be configured to connect to one of the common tuple repositories, such as Oracle 11g20 and Sesame.21 The MLSAF framework provides an adapter to those two, and the developer could easily customize a new adapter for others.

## 4.4 MLSAF Characteristics

The proposed framework supports most of the requirements described in section 4.1. The first requirement, which is related to support for a new language, could be achieved easily by providing NLP processing resources using one of the common NLP toolkits, such as

[20] http://www.oracle.com
[21] http://www.openrdf.org

GATE22 or UIMA. Addition of the NLP language-specific processor to MLSAF using the provided interfaces will not harm the multilingual nature of the framework.

For supporting new domains, new ontologies are required; and in addition, customizations also need to be made in relationship recognition only. MLSAF provides an open interface to the developers and researchers to plug in new document formats related to specific domains. MLSAF provides a format parser for common Web documents (i.e., HTML, XML, PDF, MS office, etc.). MLSAF provides support analysis for semistructured and unstructured document styles. The developers could easily extend exiting analyzers to support new styles.

MLSAF provides different annotation writers to generate different embedded annotation languages such as RDFa, Microformat, and Microdata in addition to standalone annotation languages such as RDF, N3, and Turtle. Developers could also easily extend those provided writers to generate different annotation languages.

MLSAF provides adapters to three common triple-store repositories, namely, sesame, Oracle, and ReSQL. The developer could also customize new adapters to other triple stores. MLSAF provides robust components that handle the acquisition of Web sources using focus crawlers as well as documents stored in local networks or folders.

[22] http://gate.ac.uk

# CHAPTER 5

# ONTOLOGY LEARNING

Ontology learning is the process of building the ontology. There are several approaches to doing so. A new dimension of ontology learning is to build a multilingual ontology. Such ontology is needed for many applications, such as cross-lingual information access, and multilingual information extraction for domains such as food and health.

The availability of open Web-based information sources, such as Wikipedia, in semistructured formats makes it possible to automatically build or extend multilingual domain ontologies. In this chapter, we show an ongoing work in how to build such an ontology utilizing Wikipedia and other multilingual online data sources. The constructed ontology is built to capture the culture for each language based on the available concepts for each language.

## 5.1 Existing Ontology Learning Approaches

The ontology construction process is not an easy job and requires a lot of time and effort to build. To help reduce the time and effort in this process, several ontology learning systems have been developed that allow extracting concepts and relations between concepts from unstructured data source such as OntoLearn [56] and OntoMiner [15]. Those systems, along with many other systems, are discussed in detail in [57]. Most of these systems use NLP techniques in a shallow manner, extracting mainly taxonomic

concepts. Biomedical fields have been primarily addressed in the taxonomic relationship discovery approaches due to the availability of large text collections readily available in sources such as PubMed.

Automated ontology construction systems such as Text-2-Onto [58] have the possibility to extract nontaxonomic (hyponymic) relationships between concepts using crafted rules and regular expressions. The issues with those tools are their limitations in the effectiveness of extracting domain-specific concepts, because they identify semantic relations based on POS tags only.

An effective system for extracting concepts in general domains that mainly include person, organization, and location named entities was described by Cimiano and Staab in [59]. They used taxonomic and nontaxonomic rules and patterns for semantic relationship discovery between concepts as a preliminary step for entity classification.

Weber and Buitelaar [60] purposed an ontology learning system that uses Wikipedia and other online data sources to create domain ontology. The system was developed to create ontology in the German language only. The system uses an unsupervised NER system to extract entities, then uses Wikipedia to further classify new entities and place them in the right position on the ontology. It starts with base ontology and applies ontology-based NER on a given corpus, and then it applies syntactic pattern analysis on the context of the extracted entities to extract recognized new entities. Finally, it uses online information sources like Wikipedia to extract additional information about the newly recognized entities and places them in the right positions on the new domain ontology.

Tamagawa et al. [61] presented an approach to automatically build a general-purpose ontology from the Japanese Wikipedia. Five types of relationships were extracted: Is-a, class-instance, data property, property definition domain, and synonym relationships.

Yago [62] is lightweight and extensible ontology. Yago contains a huge set of more than a million entities and more than five million facts. The relationships between those entities include taxonomic and nontaxonomic relationships. The facts and concepts have been automatically extracted from Wikipedia and unified with WordNet. A combination of rule-based and heuristic methods were used to build the ontology in the English language. The method used to build the ontology cannot be applied and needs to be modified in order to build a similar ontology in other languages due to the fact that the techniques depend on English grammar.

DBpedia [8] is a large ontology and knowledge base extracted from Wikipedia. Properties and concepts of DBpedia's ontology are built manually, and it consists of 320 concepts. It was built using information resources such categories of the article it belongs to, infobox, and external links.

Hecht and Gergle [63] conducted a study on knowledge diversity across Wikipedia for different languages. Two types of diversity were discussed, the first related to concepts that are available in some languages and not others and the second related to subconcepts that are available in some languages and not others. Reasons and effects of that diversity were discussed. Most of the foreseen ontology learning approaches lack the multilingual ontology construction and utilize the Wikipedia editions. Our approach overcomes this limitation by construction ontologies connected by language-agnostic ontology acting as a

bridge between them. In addition, each monolingual ontology captures the cultural aspect of each language represented as Wikipedia's edition.

## 5.2 Open Data Sources

There are several open data sources available today. Some of those data sources cover many domains, and some are domain-specific. Those data sources present data in different formats: structured, semistructured, and unstructured. In this chapter, we make use of two open domain data sources: Wikipedia and WordNet. Wikipedia itself is represented in a semistructured form whereas WordNet is represented in a structured format. Both of these data sources come with multiple language editions.

### 5.2.1 Wikipedia

Wikipedia [64] contains more than 22 million articles from 271 languages, according to May 2012 Wikipedia statistical figures. Figure 5.1 shows the article counts for some of languages, with the article count for English around four million articles. There are common concepts between languages, and there are concepts that are specific to each language. Wikipedia uses interlanguage links (ILLs) to identify and group articles about the same concept in different language editions. ILLs are connections between articles in distinct language editions entered by humans and propagated. They are supposed to indicate near conceptual equivalence between pages in different languages. According to a study done by [63], only around 25% of concepts overlap between Wikipedia language editions, and the remaining large percentage of concepts are described only in one language. Reasons for this diversity are culture and missing links between the concepts not done by users who update the articles.

### 5.2.2 Global WordNet

WordNet is an English-language lexical database [65]. Words are grouped into sets of synsets. Synsets provide general definitions and record the various semantic relationships between the synonym sets. Table 5.1 shows the WordNet 3.0 database statistics. There is a WordNet edition for most languages. To link different WordNet language editions together, a concept called Base Concept (BC) was introduced to act as a medium between those editions. It acts as fundamental building blocks for establishing the relationships in WordNet and gives information about the dominant lexicalization patterns in languages. There are three types of BC that have been distinguished: Common Base Concepts (CBC) that act as BC in at least two languages, Local Base Concepts (LBC) that act as BC in only a single language, and Global Base Concepts (GBC) that act as BC in all languages of the world. For example, there are 1,024 CBCs in EuroWordNet [66], which is a multilingual lexical database of WordNet for different European languages (German, French, Dutch, Italian, Spanish, Czech, and Estonian) in which WordNet editions are linked to each other through an interlingual index.

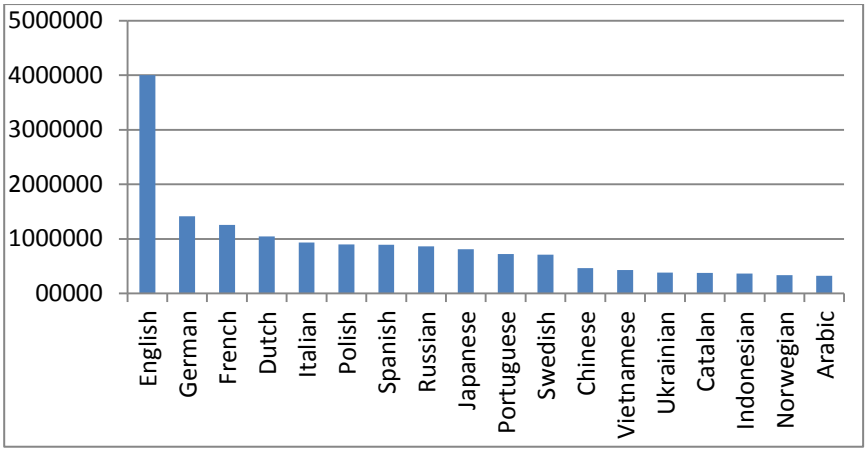



**Figure 5.1 Wikipedia Article Statistics by Language**

Comparing the number concepts defined in WordNet to the number of articles found in Wikipedia, there is a significant difference between the two. Each article in Wikipedia is considered a name entity, which is a concept or an instance of a concept. Wikipedia is considered to be a large knowledge base that consists of concepts and instances of concepts.

**Table 5.1 WordNet 3.0 Database Statistics**

| ***POS*** | ***Synsets*** | ***Unique Strings*** | ***Total Word-Sense Pairs*** |
|---|---|---|---|
| Noun | 82115 | 117798 | 146312 |
| Verb | 13767 | 11529 | 25047 |
| Adverb | 3621 | 4481 | 5580 |
| Adjective | 18156 | 21479 | 30002 |
| **Totals** | **117659** | **155287** | **206941** |

## 5.3 The Proposed Ontology Learning Approach

To build such multilingual ontology, we start by manually developing an initial upper ontology in English for the domain in question. Then we extend the initial ontology using

the YAGO ontology construction approach that was described in [62]. For each concept in the new ontology, we start to create a multilingual concept group using conceptualign algorithm [63] with the English concept and following ILLs between articles in different Wikipedia language editions. For each concept group, we search for missing links between existing and matching concepts that were not linked by ILL using missing link finder developed in [63]. In each concept group, we create a new node as a hyperlingual or universal concept and connect to all concepts in the group with it using hasAgnosticConcept object property. To validate each concept in each multilingual concept group, we use Explicit Semantic Analysis (ESA) (6) to ensure the correct relatedness of each concept to the group (and remove from the group if not valid). At this stage, we have scattered concepts for each language connected in groups. To build monolingual and agnostic ontologies, we need to connect concepts for each language by following the link between concepts in the English ontology. The summary for the construction algorithm is shown below. Figure 5.2 shows the ontology constructions steps for clarification where step (a) shows the initial ontology; (b) the extended English ontology; (c) one of the concept group consisting of four languages, with one language having two concepts in the same group; and (d) multilingual ontology where the agnostic ontology acts as a bridge between monolingual ontologies.

### 5.3.1 Manual Ontology Construction

The initial ontology for food and health domains was manually constructed using several lists of food retrieved from publicly available domain-specific data sources mentioned in the data source section before: USDA, AGROVOC, EuroFIR, and LanguaL. After manual

merging and deduplication of the above-mentioned food lists, we have more than 150 concepts in the initial English ontology. The initial ontology consists of concepts that cover food, ingredients, nutrition, and health condition–related concepts. Part of the manually constructed ontology is shown in Figure 5.2.

### 5.3.2 Automated Multilingual Ontology Construction

We started with initial monolingual ontology that was created manually. The process of construction starts by using Wikipedia and WordNet as sources for enrichment of monolingual ontology and construction of the multilingual ontology.

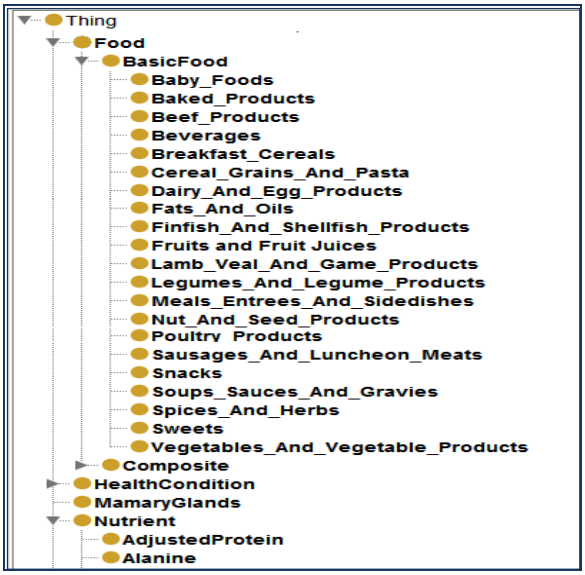



**Figure 5.2 Initial Food Ontology**

The process is a complex process, and its summary will be shown next.

1. Manually build an initial upper-level ontology in the English language with a link to a related Wikipedia article for each concept.
2. Extend the upper ontology using YAGO ontology construction approach [62].

3. For each concept in the extended ontology, use conceptualign algorithm [63] to construct multilingual concept group.
4. Add a hyperlingual concept in each concept group, connect it to all nodes in the concept group, and then link it using the hasAgnosticConcept object property.
5. For each concept group, search for missing links using missing link finder [63] to find a missing link between all language concepts.
6. Validate each concept in each multilingual concept group using Explicit Semantic Analysis (ESA) [67] to ensure the correct relatedness of each concept to the group.
7. Connect all concepts of each language using link creates in the Ontology English edition of the extended ontology created in step 2 and concept group links in the subconcept diversity in each language.
8. Output each monolingual ontology and agnostic ontology in separate OWL-formatted files.

## 5.4 Preliminarily Evaluation

In order to evaluate the quality and effectiveness of the proposed system, we have implemented a simple experimentation by using JADE API.3 and conducted two evaluations. The first is based on the similarity measure of the produced ontology with respect to reference ontology. The second evaluation targets the behavior and system, which could be measured based on the expected sequence of events.

For the first evaluation, precision and recall measures are used because they are good indicators of the correctness of the system in generating the target results. We extracted from YAGO ontology a set of subontologies related to food, nutrition, and disease for this evaluation. A number of concepts and depth levels of the extracted and learned ontologies are shown in Table 5.2. We have used global lexical and taxonomic measures defined in [68] to evaluate the learned ontologies. Lexical precision (LP) is the measure of the ratio of correctly extracted constructs over all automatically extracted constructs. Lexical recall (LR) quantifies the number of relevant lexical constructs that are extracted from the analyzed dataset compared to all constructs to be extracted from all datasets.

$$LP(O_C, O_R) = \frac{|C_C \cap C_R|}{|C_C|} \qquad (5.1)$$

$$LR(O_C, O_R) = \frac{|C_C \cap C_R|}{|C_R|} \qquad (5.2)$$

where $O_C$ stand for computed ontology, $O_R$ stands for reference ontology, $C_C$ stands for concepts in $O_C$, and $O_R$ stands for concepts in $O_R$.

Taxonomic precision (TP) is a measure of the ratio between the correctly placed concepts over all place concepts. Taxonomic recall (TR) quantifies the number of relevant concept hierarchy over the total concepts hierarchies.

$$TP(O_C, O_R) \coloneqq \frac{1}{|C_C|} \sum_{c \in O_C} \frac{tp(c, c, O_C, O_R) \quad if\ c \in C_R}{max_{c' \notin C_R}\ tp(c, c, O_C, O_R)\ if\ c \notin C_R} \quad (5.3)$$

$$TR(O_C, O_R) \coloneqq TR(O_R, O_C) \quad (5.4)$$

A summary of the evaluation using the five measures is shown in Table 5.3.

In the second evaluation, we were able to evaluate the speed, and the path system will follow for each type of change in the environment. To do this, we updated the Wikipedia articles from different perspectives and monitored the effects on the system's behavior and outcome. In this evaluation we selected several Wikipedia pages. The two evaluations show a promising performance in terms of the common measure and effectiveness of the ontology learning as well as the capability to cope with constructive changes in the data sources, as shown in Table 5.4, where P and R stand for precision and recall, respectively.

**Table 5.2 Reference and Learned Ontologies Description**

| Ontology | # of Concepts | Depth Levels |
|---|---|---|
| Yago-Food | 1130 | 6 |
| Yago-Nutrition | 234 | 3 |
| Yago-Disease | 3454 | 7 |
| Learned-Food | 2152 | 7 |
| Learned- Nutrition | 423 | 4 |
| Learned- Disease | 4554 | 9 |

**Table 5.3 Evaluation of learned Ontologies**

| Ontology | LP | LR | TP | TR | TF |
|---|---|---|---|---|---|
| Food | 95.21% | 71.32% | 51.02% | 67.57% | 58.14% |
| Nutrition | 90.64% | 69.13% | 56.32% | 65.34% | 60.50% |
| Disease | 98.52% | 89.35% | 59.44% | 71.51% | 64.92% |

**Table 5.4 Evaluation of Wikipedia Changes**

| Wikipedia page | Change | Impact |
|---|---|---|
| http://en.wikipedia.org/wiki/Apple | Move different category | Restructure change; Lower P |
| http://en.wikipedia.org/wiki/Apple | Move different deeper level | Restructure change; no impact on P, R |
| http://en.wikipedia.org/wiki/Apple | Delete page | Concept get discovered from other Wikipedia language edition; no impact on P, R |
| http://en.wikipedia.org/wiki/Apple | Add new category | Additional relational links; improve R |
| http://en.wikipedia.org/wiki/Apple + http://en.wikipedia.org/wiki/Orange | Merge | Restructure change; impact on P, R |

## 5.5 Chapter Summary

In this chapter, we have presented how to build an ontology using multilingual online structured and semistructured data sources, such as Wikipedia and WordNet. The constructed ontology consists of parallel monolingual ontologies and an agnostic ontology connecting them together. The agnostic ontology acts as a bridge between the monolingual ontologies. Each monolingual ontology consists of the concepts that exist in that language. The number of concepts in a monolingual ontology could be different from

the other ontologies due to the diversity in each language. It also reflects the culture for each language based on the available concepts for each language.

Our future direction is to build a system that generates this multilingual ontology automatically using all methods and tools discussed in this thesis. Also, we will be working on augmenting the ontology with metadata that eases the information extraction and cross-lingual information access.

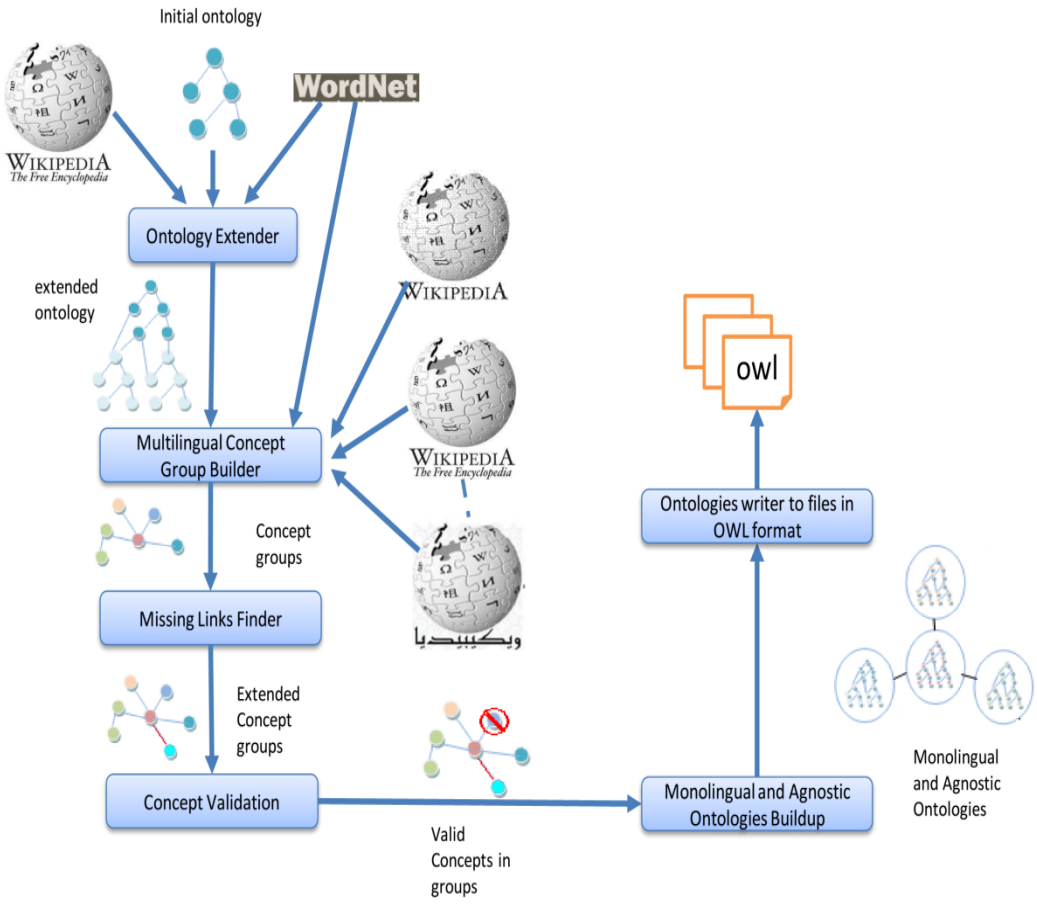



**Figure 5.3 Architecture of The Proposed Ontolog Learning System**

# CHAPTER 6

# ONTOLOGY AND KNOWLEDGE-BASES

This chapter introduces the processes used to develop domain ontologies. We explain the development cycle for the multilingual cross-domain ontologies for food, nutrition, and health. First, we define the requirements for each domain. Second, we investigate the existing related ontologies and summarize their limitations with respect to the requirements. Third, we explain how we use the introduced processes to fulfill the requirements. Finally, we describe the developed ontologies for food, nutrition and health domains.

## 6.1 Introduction

Ontology is a representation of knowledge in formal form. It involves a network of concepts within a certain domain using a shared terminology for the types, properties, and relationships between the domain's concepts. Different ontologies are developed for different domains by the domain experts to fulfill certain objectives.

Ontology serves a single domain, and some applications need to use ontologies from different domains to integrate different information sources. Moreover, there could be several ontologies developed for the same domain due to different languages, cultures, types of expertise, and purposes. Therefore, there is a need to integrate existing ontologies in order to capture cross-domain knowledge.

As mentioned in the framework chapter, we need to plug domain ontologies into the framework. Ontologies help annotation in having standard reference for the acquired knowledge. So, Web sources can be structured in knowledge-bases based on the domain ontologies. These knowledge-bases are used in the semantic manipulation of the user's queries returning more relevant results.

Our objective is to apply the proposed framework to build a multilingual semantic Web search application for the food, nutrition and health domains as they are critical domains. This will help the community in providing food recommendation based on the user's health conditions. In order to provide such capabilities, we need integrated ontologies between different domains such as food, nutrition, and health. In addition, we want to utilize the knowledge discovered in one language to be used for people in a different language. Ontologies that satisfy these requirements do not exist. Therefore, we were challenged to develop these ontologies by creating, integrating, and reusing some of the existing ontologies to meet our requirements. Next, we present the processes we have followed in developing these ontologies.

## 6.2 Ontology Development Processes

We introduced four processes below that use some of the existing methodologies. We used these processes to develop multilingual cross-domain ontologies for the food, nutrition, and health domains. The processes are described in the following tables with the inputs, outputs, and possible methodologies that can be followed in each.

**Table 6.1 Domain Ontology Development Process**

| | |
|---|---|
| Process No | 1 |
| Process Name | Domain Ontology Development |
| Description | To develop or reuse certain domain ontology that satisfies the application requirements. |
| Input | Application requirements |
| Output | Domain Ontology |
| Methodologies | - Reuse a single existing domain ontology as is.<br>- Reuse multiple heterogeneous domain ontologies as they are.<br>- Build domain ontology from scratch. |

**Table 6.2 Cross Domain Ontologies Development Process**

| | |
|---|---|
| Process No | 2 |
| Process Name | Cross Domain Ontologies Development |
| Description | To have integrated cross domain ontologies |
| Input | Different domains ontologies |
| Output | Integrated cross domain ontologies |
| Methodologies | - Reuse an existing integration between different domain ontologies as is.<br>- Extend an existing integration between different domain ontologies, i.e. add additional integration points.<br>- Build an integration between different domain ontologies from scratch (merge ontologies into one ontology, create an integration ontology and linking the ontologies with relationship). |

**Table 6.3 Multilingual Ontologies from Multiple Mono-lingual Ontologies Process**

| | |
|---|---|
| Process No | 3 |
| Process Name | Multilingual Ontologies Development from Multiple Mono-lingual Ontologies |
| Description | To have integrated multilingual ontologies based on multiple mono-lingual ontologies |
| Input | Multiple mono-lingual domain ontologies |
| Output | Integrated multilingual domain ontologies using either one-to-one mapping or agnostic ontology acting as a bridge between the existing ontologies. |
| Methodologies | - Automatic alignment between the mono-lingual ontologies (e.g. using translation service, mediator like Wikipedia)<br>- Manual alignment between the mono-lingual ontologies<br>- Semi-automatic alignment between the mono-lingual ontologies (human guided) i.e. partially automatic and partially manual. |

**Table 6.4 Multilingual Ontologies from Single Mono-lingual Ontology Process**

| | |
|---|---|
| Process No | 4 |
| Process Name | Multilingual Ontologies Development from Single Mono-lingual Ontology |
| Description | To have integrated multilingual ontologies starting from a single mono-lingual ontology |
| Input | Single mono-lingual domain ontology |
| Output | Integrated multilingual domain ontologies using either one-to-one mapping or agnostic ontology acting as a bridge between the existing ontologies. |

| | |
|---|---|
| Methodologies | - Option-1: (create different ontology for each culture)<br>  o Use “Domain Ontology Development” process to create another mono-lingual domain ontology<br>  o Use “Multilingual Ontologies Development from Multiple Mono-lingual Ontologies” process to align the two mono-lingual domain ontologies.<br>- Option-2: (enrich the existing ontology or replicate it)<br>  o Automatic Translation of the input mono-lingual domain ontology into a new language.<br>  o Manual translation for the input mono-lingual domain ontology<br>  o Semi-automatic translation for the input mono-lingual domain ontology (human guided) |

## 6.3 Ontology Development Cycle

In order to develop the domain ontologies, the requirements need to be captured from the objective and intended use. Then, related existing ontologies are surveyed and assessed to see if they meet the requirements. Finally, we explain how we followed the introduced processes.

### 6.3.1 Requirements

We aim to provide answers to questions related to food, nutrition, and health domains. Some examples of these questions include: “Are apples good for people with heart diseases?” “How much honey can be taken by a diabetes patient?” “What are the health benefits of eating pineapple?” and “What are the fruits that contain the daily needed quantity of calcium?” In order to answer such questions, it is necessary to have integrated ontologies for different domains: food, nutrients, health (diseases, body parts, body functions), and recipes. Moreover, in order to answer queries in different languages, the

systems and ontologies should support multilingual access. In addition, in order to answer queries that require aggregation of information, we need to have multilevel ontologies. Also, in order to achieve high relevancy and coverage, we need to use ontologies that have rich, comprehensive vocabularies. Moreover, in order to make effective use of the annotation, ontologies' concept names should be unique and self-contained.

One of the most-used and richest knowledge bases for food and nutrition is the U.S. Department of Agriculture (USDA) database. To develop the food and nutrition ontologies, the USDA (18) food database was used as a guide. In the USDA database, foods are clustered under 25 classes, and for each food there are a maximum of 146 nutritional values. The link between a food item and its nutrients is based on 100g of that food.

### 6.3.2 Related Ontologies

Based on the criteria discussed above, we have considered related ontologies for food, nutrition, and health. In the next sections, we will present a short description about each one, highlighted with respect to the requirements given before.

#### 6.3.2.1 Semantic Diet Ontologies

Evan Patton developed a project called Semantic Diet23 for the purpose of helping people eat healthier. He provided a set of ontologies related to food and nutrition based on the USDA database. We have used those ontologies as a base to build food and nutrition ontologies. The main ontology has one concept related to nutrition and two concepts related to food, based on two USDA food tables: food items and food groups. In addition,

[23] http://semanticdiet.com/

several ontologies related to recipes, units of measurement, common measures for foods, and USDA nutritional guidelines. The limitations of semantic diet ontologies are that they provide shallow alignment with USDA, flat lists with no grouping, and support a single language only.

**Table 6.5 Semantic Diet Ontologies**

| Ontology | Number of triples |
|---|---|
| Recipe | 124 triples |
| Food Groups | 100 triples |
| Units for measurements | 65 triples |
| Common measures for foods | 118,791 triples |
| Nutrient | 2,847,367 triples |
| Nutritional Guidelines | 136 triples |

#### 6.3.2.2 International Classification of Diseases (ICD-10) ontology

The ICD-10[24] ontology is formalized in OWL-DL of the tenth edition of the International Classification of Diseases (ICD), which was published by the World Health Organization (WHO) in 2004. It consists of 14,502 concepts consisting of diseases and healthcare procedures. The original language of this ontology is English, and there is partial translation of this ontology into other languages. This classification was developed in order to have standard categories for diseases and other health problems that include death certificates and health records. It has been used in the medical information system. The limitations of ICD-10 are as follows:

- Limited support for Arabic

[24] http://www.who.int/classifications/icd/en

- Uses technical names for disease, no synonyms
- Mixed up between body parts with diseases (class names) and depend on the path
- Not useful for annotation; more into human use than machine use .

#### 6.3.2.3 Human Disease Ontology

The Disease Ontology (DO)25 is a public ontology of human diseases that support integration of biomedical data. DO ontology contains well-defined terms using standard references. These terms are associated with well-adopted and well-established terminologies such as SNOMED, UMLS, ICD-9, ICD-10, and MeSH. DO ontology contains a broad knowledge base of 8,043 human diseases. The strengths of DO are the link to other disease ontologies and richness with synonyms for each disease, and useful for annotation. The short coming is only the single language support.

#### 6.3.2.4 AGROVOC Ontologies

AGROVOC26 is a controlled vocabulary developed by the Food and Agriculture Organization (FAO) of the United Nations. It covers FAO areas of interest such as agriculture, food, nutrition, fisheries, forestry, and environment. AGROVOC contains over 32,000 concepts organized in a hierarchy; each concept may have labels in up to 22 languages, including Arabic, and additional languages versions are under development.

AGROVOC is represented by using the RDF/SKOS-XL format and is made accessible through using a SPARQL endpoint. Other formats are available for download also. AGROVOC can be used by application developers through provided Web services.

[25] http://disease-ontology.org/
[26] http://aims.fao.org/standards/agrovoc

#### 6.3.2.5 FOODS Ontology

FOODS [69] is food ontology grouped into nine main concepts: regional cuisine, dishes, ingredients, availability, nutrients, nutrition-based diseases, preparation methods, utensils, and price. The limitation of FOODS ontologies is that they only support English and have no alignment with the USDA and DO.

#### 6.3.2.6 PIPS Ontologies

The PIPS food ontology [70] has 261 concepts with only two object properties. Foods are grouped into 13 groups: Vegetables, Grain_products, Special_nutrition_products, Beverages, Sea_Food, Egg_products, Oils_fats_products, Meat, Soups_Sauces_Misc, Fruits, Sugar_products, Nuts_seeds, and Milk_products. The problems with this ontology is that it is English only and is not aligned with the USDA.

## 6.4 Domain Ontologies Description

Domain ontologies consist of food, nutrition, health, integration, and other ontologies related to culture and user profiling. In this section we will elaborate on the four listed ontologies only y.

### 6.4.1 Food Ontology

The food ontology, as shown in Figure 6.1, contains all food items that also belong to certain food groups represented by other ontologies. All food items in the food ontology are collected from the USDA [71]. The food group ontology is for the categorization of the food ontology instances. The food groups described by this ontology are taken from the USDA database.

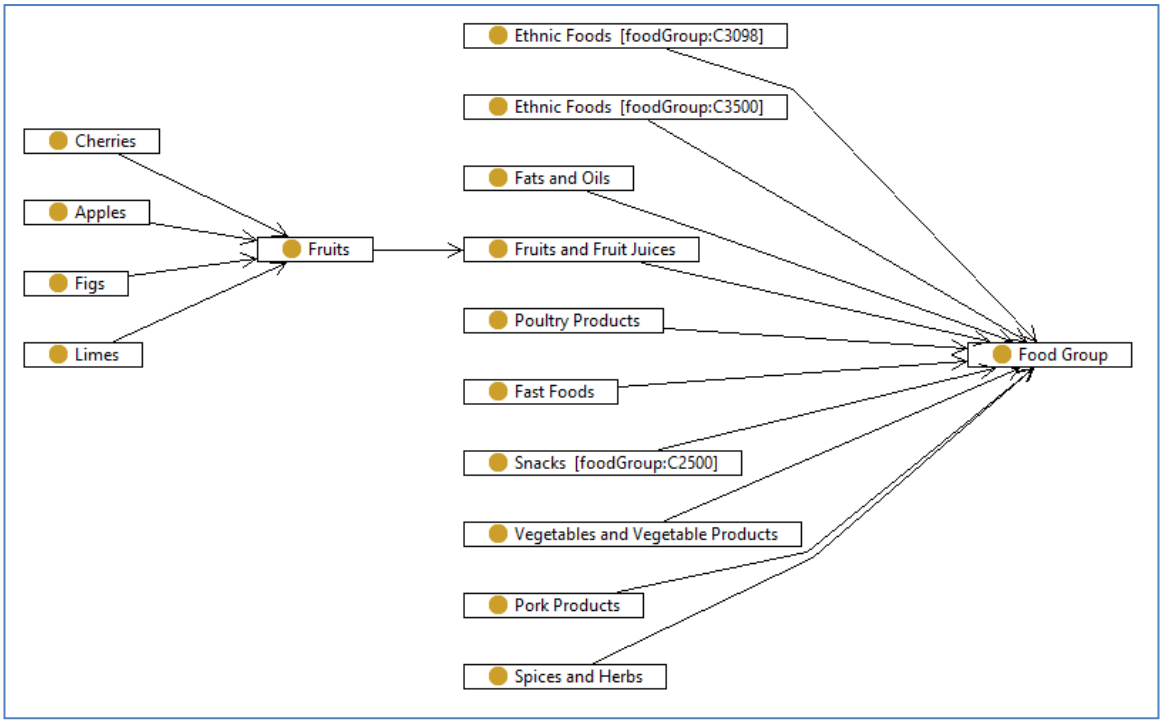



**Figure 6.1 Food Ontology**

### 6.4.2 Nutrition Ontology

The first-level proposed nutrition ontology consists of aggregation layers of different types of nutrients, such as vitamins, minerals, and fats. Under each concept, deeper-level concepts are enumerated. The nutrition ontology contains at most 5 layers at this stage. Part of the nutrition ontology is shown in Figure 6.2. The nutrition ontology contains all nutrients information that could be available in a given food with all the details available for each food based on the USDA database [71].

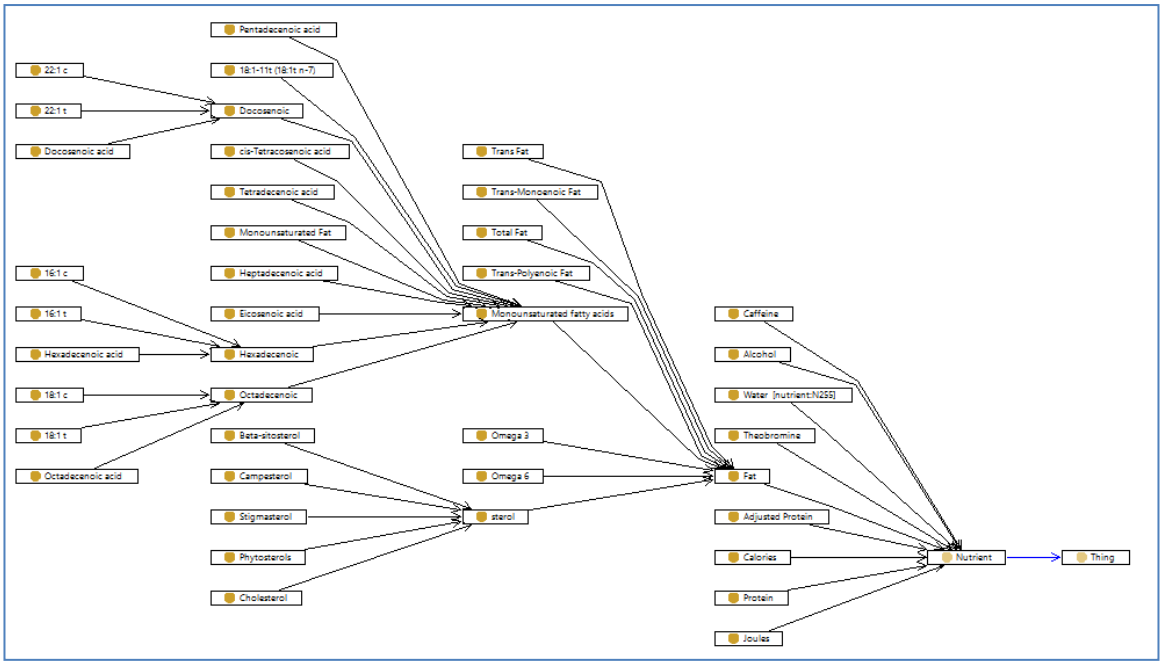



**Figure 6.2 Nutrition Ontology**

### 6.4.3 Health Ontology

The proposed health ontology consists of three main concepts to model human diseases, body parts, and body function. This modeling came to fulfill the purpose of creating this ontology and to be able to answer questions that might be asked by the user with respect to the relationship between food and nutrition and the different aspects of the human. Under disease concept, we can plug in any disease ontology from the ones discussed previously. The body part concept models different parts of the human body like head, arm, eye, or skin. The body function concept models human body functions and processes like vision, absorption of food, or generation of blood. The three upper concepts are generic enough to capture different human relation health conditions. Part of the proposed health ontology can be seen in Figure 6.3.

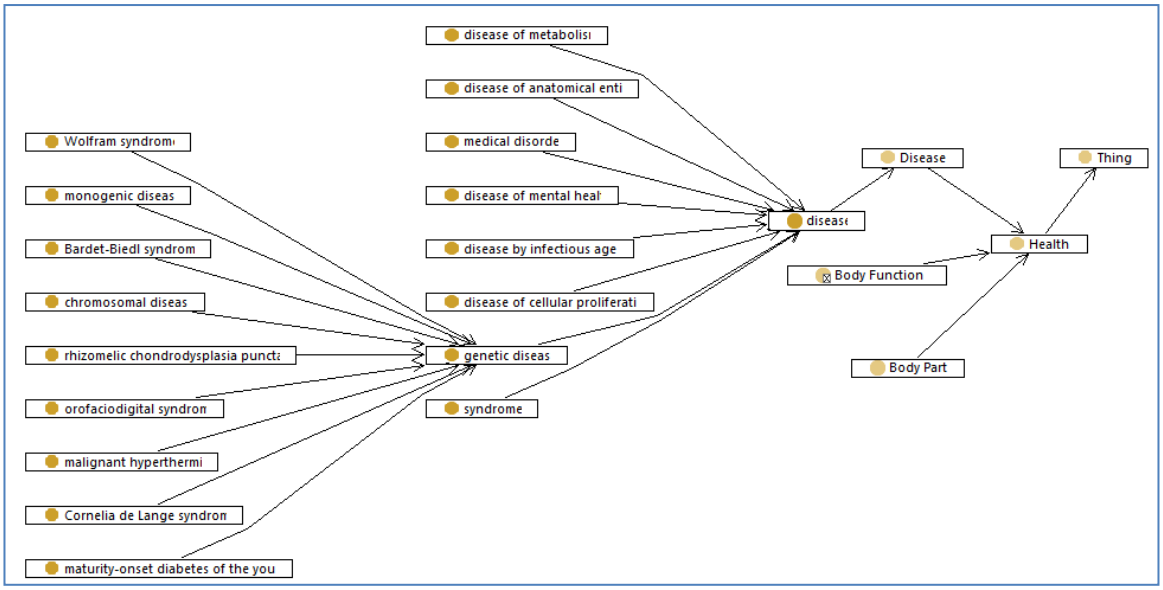



**Figure 6.3 Health Ontology**

### 6.4.4 Integration Ontology

In order to capture the relationship between domain ontologies concepts and source of the information, separate integration ontology was created. It contains concepts for document, sentence, entities, and relationships. The document concept captures the metadata of the Web document processing in annotation. The sentence concept represents the text segment in the document where entities and relationships of interest were recognized. The entity concept is stored in the metadata for named entities found in the document texts and references to ontological resources. A single sentence could contain several entities and relations between them. Figure 6.4 shows the core part of the integrated ontology.

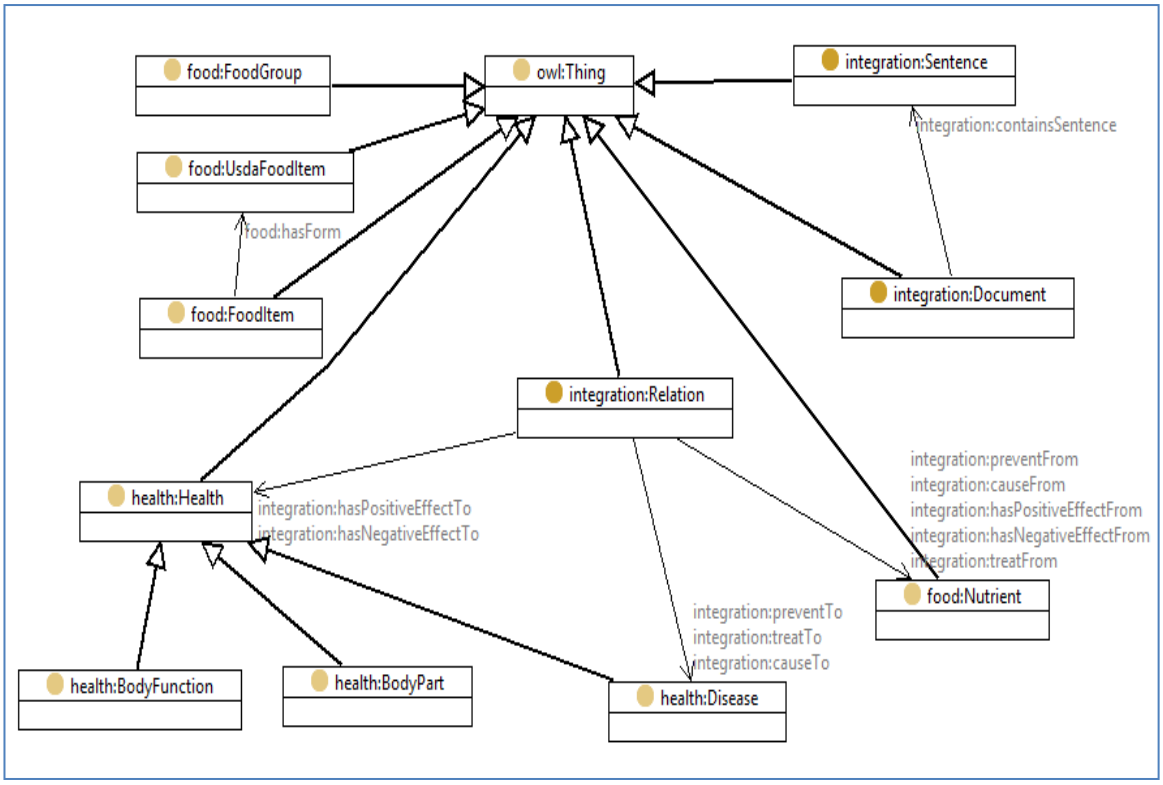



**Figure 6.4 Integrated Ontology**

The relation concept is the heart of the ontology integration, as it makes it possible to relate all these separate ontologies to each other for extracting the data as well as reasoning on the data. The relation concept contains the mapping between food, nutrition, and health ontologies to allow us to capture the relationships between various Web resources.

The relation ontology holds attributes like *hasPositiveEffectTo*, prevent, *hasPositiveEffectFrom*, causes, and so on, which map to foods, nutrients, diseases, body functions, and body part instances.

The sentence concept is used to maintain the reference of the extracted information from the document level of a given Web resource. It has three data properties describing the

position of the sentence in the document and textual content. It has one object property that links the sentence instance to the source document.

The document concept is used to maintain the meta-data about the source Web document. It holds information like URL, title, and type of the document. It is a simplified version of *foaf:Document* concept.

## 6.5 Chapter Summary

In this chapter, we introduced the processes that we used to develop the domain ontologies. We explained the development cycle for the multi-lingual cross-domain ontologies for food, nutrition and health. Starting with the definition of the requirements for each domain, we investigated the existing related ontologies and summarized their limitations with respect to the requirements. Then, we explained how we use the introduced processes to fulfill the requirements. Finally, we describe the developed ontologies for food, nutrition, and health domains.

# CHAPTER 7

# FRAMEWORK IMPLEMENTATION

In this chapter, we explore the implementation details of the proposed framework and challenges faced. We show how things were done with some walk-through examples and cases. We show some screenshots of different components and how to use them.

## 7.1 Used Toolkits and Tools

In the implementation of MLSAF, we used some open-source toolkits with many customizations to suit the framework objectives and domains' needs. In the next subsections, we will give an overview of those toolkits and where and why they are going to be used.

### 7.1.1 Apache Lucene

Apache Lucene27 is an open-source implementation of a high-performance and fully equipped engine, written in pure Java. Author basic version is a developer Doug Cutting. Search Lucene has been successfully translated into other environments such as Delphi, Perl, C#, C++, Python, Ruby, and PHP. A further development organization is supported by the Apache Software Foundation. The library is available under an open license Apache software license.

[27] http://lucene.apache.org

Lucene is a technology suitable for any application that requires indexing and full text search. It is particularly popular among implementations of Lucene search engines, spiders, or local search engines within individual sites. The Lucene index conceptually consists of a set of documents composed of a plurality of fields. The Lucene allows great flexibility because it is independent of specific file formats. Texts in formats TXT, HTML, PDF, MS Word, and so on are easily indexed, if only we have access to the written text. Lucene provides access to the functionality of the indexing and searching through a simple Java application-programming interface. It is fast and scalable. As it has low performance requirements, it can also be used on weaker configurations with limited memory or limited space on your hard drive. Its programming interface provides access to advanced functionality such as sophisticated query ranking results, search by selected attributes, edit and search of date intervals, and the like. Among other features, it's the incremental indexing and parallel reading and writing in the index.

The main characteristic by which Lucene has earned its reputation is the valuation and ranking results (scoring). Lucene ranking of results is extremely fast, but it also hides a lot of complexity under the hood index implementation. For ranking, vector space model (VSM) is used together with Boolean models. The idea behind the VSM is that the following documents are presented as vectors of terms, where each dimension represents the frequency of occurrences of the term in the document. If the frequency of occurrences of the term is higher compared to all occurrences of the term in any document, then the document is more relevant to the goal of higher rank, namely vice versa. This algorithm is called a TF-IDF algorithm, and it is considered one of the most efficient algorithms for ranking results. Alternatively, it is possible to use our own implementations' ranking. We

can do this upon finding the first use of the Boolean model, which is limited to the set of valid documents from the set of all documents that match the logical part of the query. Several optimizations and additives can be added (e.g., fuzzy search), but it basically remains a pure implementation of VSM [72].

### 7.1.2 GATE Tool for Natural Language Processing

Briefly, GATE [73] is an open-source Java tool for word processing and IE. It is used by scientists, large companies, teachers, and students around the world for the purpose of processing texts in different languages. Basically, GATE consists of three elements:

- Architecture, which consists of the basic components of the systems for processing the text
- Java frameworks and software libraries
- Graphical development environment that represents a user-friendly graphical interface for application frameworks

GATE is an open-source project available under the LGPL license. It is written in pure Java and verified to run on Linux, Windows, and Mac OS X. It is a mature and active project with about 20 developers. GATE is a comprehensive tool that provides or enables hand labeling, environmental performance check for word processing, elimination of information, (semi)automatic semantic markup, and much more. It is available to more than 50 different plugins that are already included in the standard distribution. GATE has no problems using various document formats. With the supplied input, parsers can read text in XML, HTML, PDF, MS Word, e-mail, or plain text. Data during the operation of staying unified memory repository that is designed to document the corps and labeling.

For persistent data has support for XML and databases Oracle and PostgreSQL. Data can be stored in the form of Java serialization.

GATE already includes well-known standard algorithms for common word processing tasks such as chunking in words, labeling the sentence elements (POS tagging), divide sentences and phrases (called entity identification), collation pronouns, and ML. GATE is also deeply integrated with other open-source projects. GATE is integrated with ML tools such as Weko, MAXENT, and SVMLight. To work with the ontology, GATE is integrated with Sesame and OWLIM.

The main features of GATE are as follows:

- Component-oriented development, which relieves the burden of directing integration and development in research projects
- Automatic measurement of word processing, which encourages comparative evaluation
- Separation between low-level (data storage, data visualization, downloading components) and high-level text processing
- A clear separation between data structures and algorithms for processing text
- Consistent use of standard components mechanisms
- The use of open standards (Unicode, XML)
- Basic processing pipeline, consisting of individual components; the user can arbitrarily set or replace their implementations

Text analysis is the process that accepts information in the form of natural language, and the results are returned in fixed format that is clear of unambiguous information. Such data can be used for direct display to users. Analysis of the text covers a family of

applications, which include the identification of named entities recognition of relationships and recognition events. GATE has been successfully applied in the domains of bioinformatics, health care, and the processing of historical court records.

### 7.1.3 Ontology API

We have used two types of API to process ontologies and write annotation files: Jena[28] and OWLAPI.[29] Jena is used for generation of standalone annotation files using RDF language. It is an open-source and well-documented library for manipulation of ontologies and knowledge bases. It has been used in generating graphical representation of ontologies and annotation files. OWLAPI is a Java API for creating, manipulating, and serializing Ontologies in OWL format. It has been used in some components to access OWL files and interfaced with external reasoners.

## 7.2 Web Acquisition Package

The acquisition package contains the components that collect the Web sources from Websites related to the domains of interest. The components interact with each other to achieve this goal. The focus crawler sends any detected RSS feed to the feed receiver. We provided implementation of those components with a focus on food, nutrition, and health domains. Minimal customization could be required for new domains.

### 7.2.1 Focused Crawler

We used the WebSphinx [74] Web crawler and reimplemented it to run in an ontology-focused crawling mode similar to the method used in [27]. The use of a topical or focused

[28] http://jena.sourceforge.net
[29] http://owlapi.sourceforge.net

crawling mode can keep it limited to relevant topics of food, nutrition, and health. It takes in the basic crawling parameters being specified as a root Web page (or several root Web pages), depth level, regular expression link visit pattern, maximum number of pages, breadth or depth of first crawl, and so on. Additionally, for running in an ontology-focused mode and computing the relevance based on it, it also takes in the background ontology and the entities we are interested in. The documents are preprocessed with a GATE [73] pipeline containing a morphological analyzer to get the word roots, while relevant gazetteers and JAPE grammars are used to find semantic entities contained in the ontology. The page relevance is scored by counting the number of entities of interest, entities that are linked to these by the taxonomy or by relations in the ontology graph multiplied by different weight measures. By limiting the crawl only to a specific Web site, we can discover all the pages of interest in that Web site. Figure 7.1 shows a snapshot of the crawler implementation.

Crawler configuration:

- Seed list
    - Manually collected from a trusted Web site
- Page score-based ontology
    - Percentage of terms matched to the total number in the document after normalization
    - A score of 10% matched is selected as a threshold for page filter.
- Trust configuration
    - Factor
        - Web site category: Hub or Authority
        - Number of links coming from trusted Web site to this site

- Authorship (.gov*, .edu*, .org*)
- Trust certification from organization such as Health on the Net Foundation (HON[30])

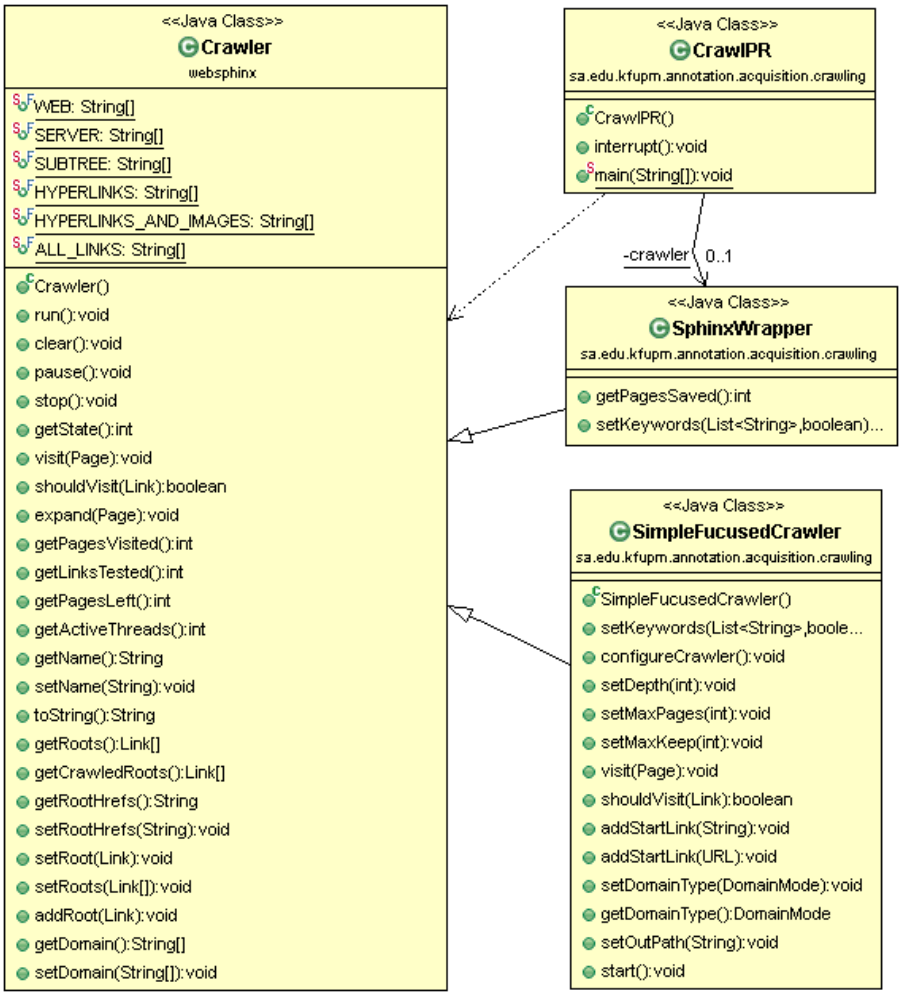



**Figure 7.1 Focused Crawler Class Diagram**

[30] http://www.hon.ch

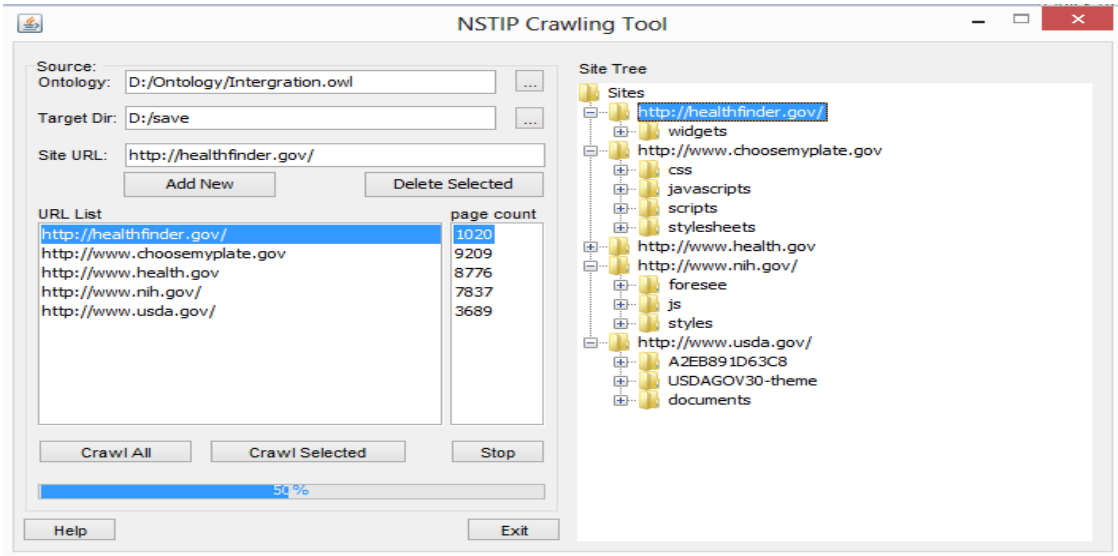



**Figure 7.2 Crawler Screen Snapshot**

**Table 7.1 Sample URL Seed**

| Website | URL |
|---|---|
| Food and Nutrition Information Center | http://fnic.nal.usda.gov |
| Dietary Guidelines for Americans | http://health.gov/dietaryguidelines/ |
| Heart UK: the Nation:  Cholesterol Charity | http://heartuk.org.uk |
| Age UK. | http://www.ageuk.org.uk/ |
| Allied Health Professions Federation. | http://www.ahpf.org.uk/ |
| Alzheimer's Research Forum. | http://www.alzforum.org/ |
| Association for the Study of Obesity (ASO) | http://www.aso.org.uk |
| Association for Nutrition. | http://www.associationfornutrition.org |
| British Association of Perinatal Medicine. | http://www.bapm.org |
| British Geriatrics Society | http://www.bgs.org.uk/ |
| British Heart Foundation. | http://www.bhf.org.uk/ |
| British Society for Allergy | http://www.bsaci.org/ |
| Centers for Disease Control and Prevention | http://www.cdc.gov/ |
| Coeliac UK website | http://www.coeliac.org.uk/ |
| Cood Food Planet Web site | http://www.coolfoodplanet.org |
| CORE: fighting gut and liver disease | http://www.corecharity.org.uk |
| Care Quality Commission | http://www.cqc.org.uk/ |

### 7.2.2 RSS Feed Reader

The RSS feed reader has two main classes derived from the original **Informa** library classes—they wrap **FeedInfo** and **ItemInfo** functionality. It also has a main class feeder that is used to start the application, holds static objects, implements the initialization of the application, and runs the main application cycle. And finally, there is a set of **extractors** and a set of **commands**. Each extractor corresponds to a source to be used when extracting HTML content. And each command is responsible for executing specific function. These commands are not executed one by one; rather, there are thread pools for each type of command, and they are executed in a batch manner. More details on commands and thread pools will be given in below.

When loading the application, the tracker log files are being read to cache items that were previously downloaded. This is necessary to be able to detect when a feed item needs updating. After the cache is initialized, the configuration of the feeds is loaded, and then each feed URL is queued for feeding in the applicant command thread pool.

Once a feed item is received, it is parsed, scheduled for refresh, and distributed further to decide whether it needs update, or it needs to have content separately downloaded. If the item passes these checks, it finally comes to the extract command where a feature map of metadata and the content of the item are finally created. All the commands and their corresponding thread pools to this step are mandatory. After the extraction, there must be at least one optional command for exporting the data to some type of storage.

### 7.2.2.1 Configuration Files

The configuration files for RSS Feed Reader can be found in the configuration folder of the project. In this paragraph each of them will be described separately.

```
############################################################
## OPTIONAL THREADPOOL CONFIGURATION
# list them space separated
ncf.optional.threadpools=persisters exporters2kim
# Persisters (although the name is not quite adequate) produce KIM "features"
# object (in the form of XML file), store original RSS to disk, get only the
# content of RSS items and store them to disk (with optional stripping of all
# HTML tags beforehand).
fr.threadpool.persisters.count.core=3
fr.threadpool.persisters.count.max=3
fr.threadpool.persisters.queue.limit=2048
fr.threadpool.persisters.queue.fair=false
fr.threadpool.persisters.keepalive.time=60000
fr.threadpool.persisters.command.class=com.ontotext.nc.feed.commands.StoreCommand
```

**Figure 7.3 Example: configuring optional thread pools**

**Table 7.2 Main configuration (feedreader.properties)**

| | |
|---|---|
| fr.base.store.dir | Sets the base storage directory (used by Store command) |
| fr.feeds.config | Locates an XML file with feed configuration |
| fr.log.tracker | Sets a file to log track info to |
| fr.log.tracker.cache.size.days | Determines the how old items to be cached |
| fr.config.httpclient | Locates a config for the httpclient |
| fr.ie.<feed machine name> | Extractor class must be present for each feed defined in the **fr.feeds.config** to use with corresponding page |
| fr.pg.<feed machine name> | Page retriever class is available only for some of the extractors defined in **fr.feeds.config** |
| fr.threadpool.<threadpool>.count.core | Number of threads to start with |
| fr.threadpool.<threadpool>.count.max | Number of max threads |
| fr.threadpool.<threadpool>.queue.limit | Limit of the pool queue |
| fr.threadpool.<threadpool>.queue.fair | Determines if the queue is ordered |
| fr.threadpool.<threadpool>.keepalive.time | Time to keep threads alive |
| fr.threadpool.<threadpool>.command.class | A class for a command to be run by the pool |
| fr.optional.threadpools | A set of optional pool names space separated |

**Table 7.3 Feeds configuration (feeds.xml)**

| | |
|---|---|
| type | RSS or Atom (Atom does not work well) |
| agency | Name of the agency |
| name | Name of the feed/channel |
| machineName | Machine name of the feed/channel |
| category | Category for feeding |
| url | URL of the feed/channel |

**Table 7.4 Alternative configuration for Web service usage (again in feeds.xml)**

| | |
|---|---|
| type | Web-search |
| searcher | URL of a search engine |
| query | The query you want to type in the search engine |
| number | Get the first N results (unfortunately max is 100) |
| type | Google |
| query | URL from the first page of an archive search on google news. Make sure that you append & num=100 to the URL |
| number | This is the number of pages to be downloaded, use the number of the last real page |

```
<feed>
      <type>rss</type>
      <agency> Mayo Clinic </agency>
      <name> MayoClinic </name>
      <machineName>mayoclinic </machineName>
      <category>Health</category>
      <url> http://www.mayoclinic.org/rss/all-news </url>
</feed>
<feed>
      <type>rss</type>
      <agency>Health Canada </agency>
      <name> Health Canada </name>
      <machineName>healthcanada</machineName>
      <category> Food and Nutrition </category>
      <url> http://www.hc-sc.gc.ca/rss/fn-an/fn-an-eng.xml </url>
</feed>
```

**Figure 7.4 Example: configuring feeds**

### 7.2.3 File Reader

The role of the file reader is to load the files stored in the local network into the document index in order to process them in the annotation pipeline. The user is asked to give the URL for the folder and a source URL for all the files. The file reader will store the source URL as metadata for the files collected from the folder. The reader will trigger the annotation pipeline tasks to process this file. Figure 7.6 shows a screenshot of the file reader where the user has to assign the folder and source URL for a Web document before starting the annotation process for all files.

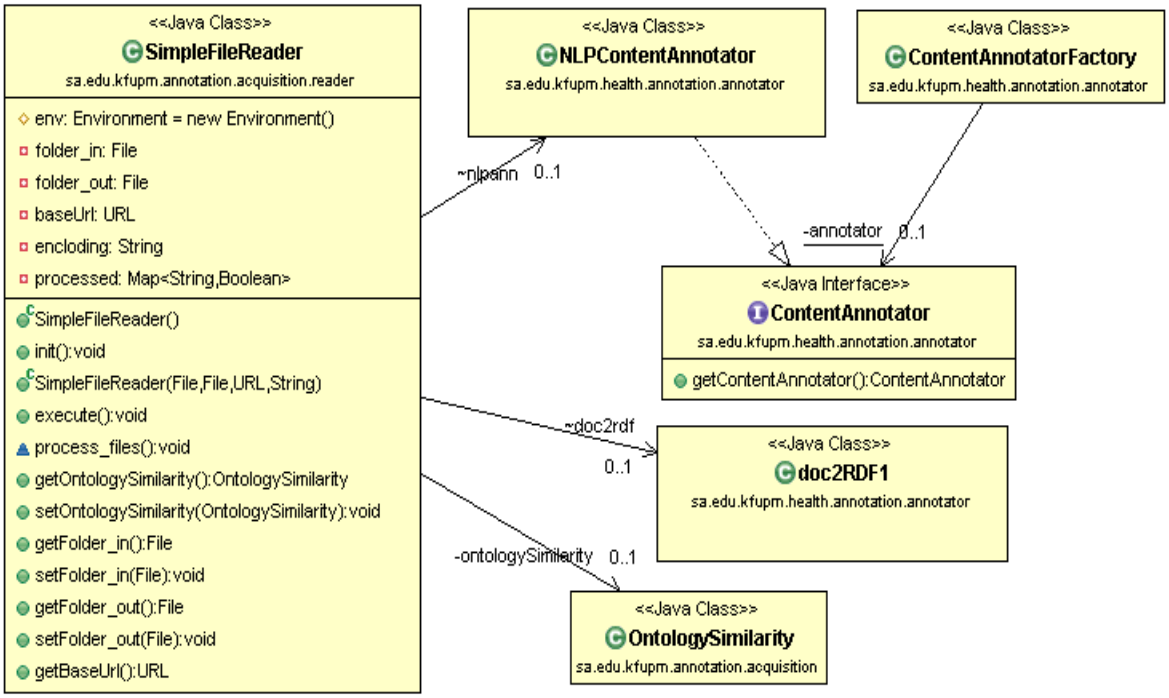



**Figure 7.5 File Reader Class Diagram**

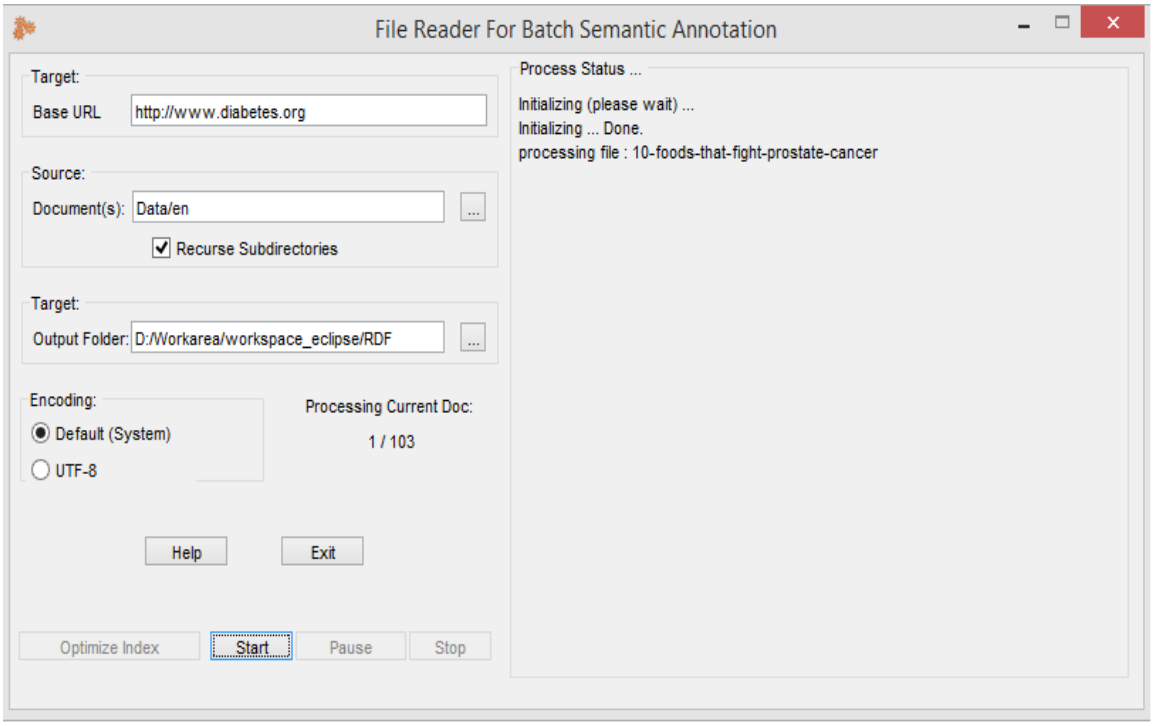



**Figure 7.6 File Reader Screen Snapshot**

## 7.3 Preprocessing Layer

The preprocessing layer's aim is to prepare the collected Web sources for processing by annotation pipeline. The Web document will be parsed to extract textual data and then analyzed to extract useful data that will be annotated by the system.

Figure 7.7 shows the class diagram for the preprocessing package. The *IDocument* interface is to be implemented by one more document based on the format of the document, and it is linked to the document model mentioned in the framework chapter.

### 7.3.1 Format Parser

The first step performed by the annotator is to transform the Web document into text by parsing the document. One implantation of the *IDocument* interface is released with *DefaultDocument* class, which is intended for textual documents. A different document format implementation could be added easily by providing an extension to *DefaultDocument* or creating new class. *IDocumentFormatParser* is in an interface for format parser. We adapted Apache Tike document parsers in the class of ApacheTikeParser in the current framework implantation. Apache Tika library for autodetection of the document format and then parses the document and converts it to textual representation. Apache Tika provides support for PDF documents and a number of the document formats from both Microsoft Office and OpenOffice. Apache Tika converts the document structure into HTML by adding a new adapter for special formats, which can be easily done by providing an implantation for the interface.

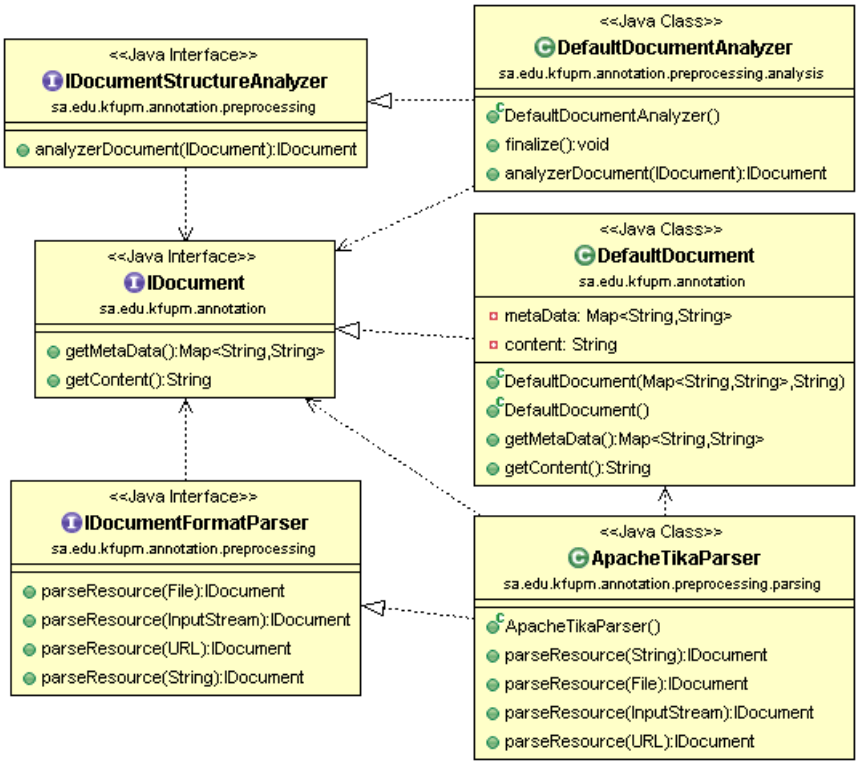



**Figure 7.7 Preprocessing Package Class Diagram**

### 7.3.2 Structure Analyzer

The role of the structure analyzer is to filter out nonrelevant data in the source documents and to separate the page into useful segments for processing. Web documents come with different structural styles; therefore, segmentation of those documents needs to be performed for each content type in order to be able to annotate them.

In the current implementation, three types of analysis are done in the Web document: paragraph extractor, table extractor, and list extractor. The paragraph extractor is responsible for extracting the textual context, which consists of a set of sentences with entities and relationships between them that follow a grammatical structure. NLP processing is required in order to recognize information presented in these sentences.

Detection of paragraphs is done using regular expression with sentences with a split between them. Table detection is based on common styles of table representation in Web pages.

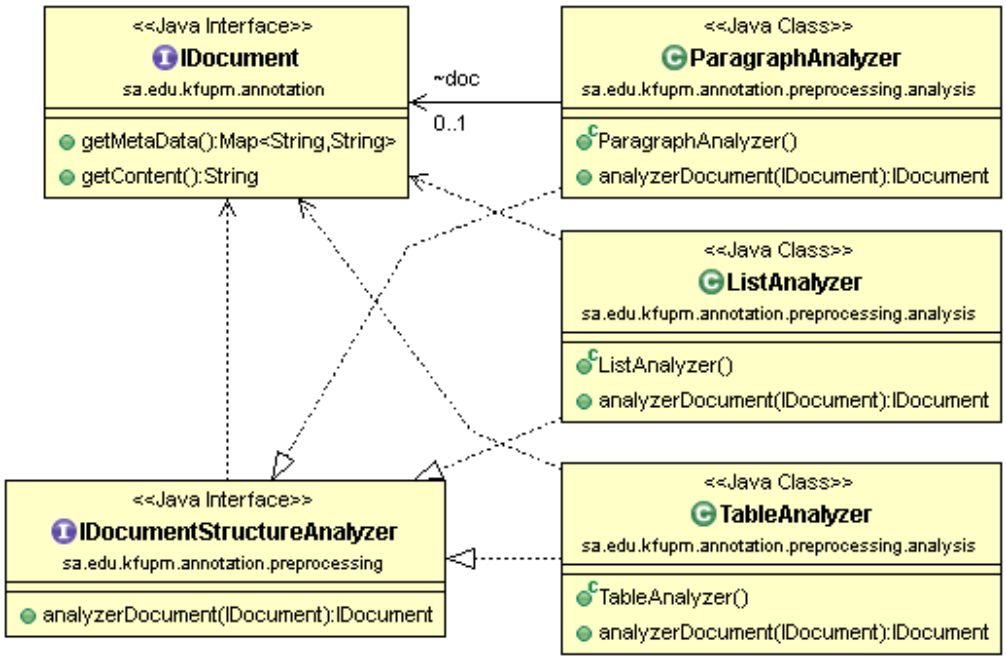



**Figure 7.8 Structure Analyzer Class Diagram**

### 7.3.3 Dominate Language Identifier

Language identification is implemented using a set of classes with a configuration file indicating the location of *n*-gram frequencies for every supported language. Frequency files can be built for specific languages and domains by providing a set of textual documents to the *FingerprintGenerator* class. The frequency files are then used to identify whether a given text belongs to a particular language. The frequency files could also be used for classifying the domain of the given text into one language. Figure 7.9 shows the class diagram for the language identification package.

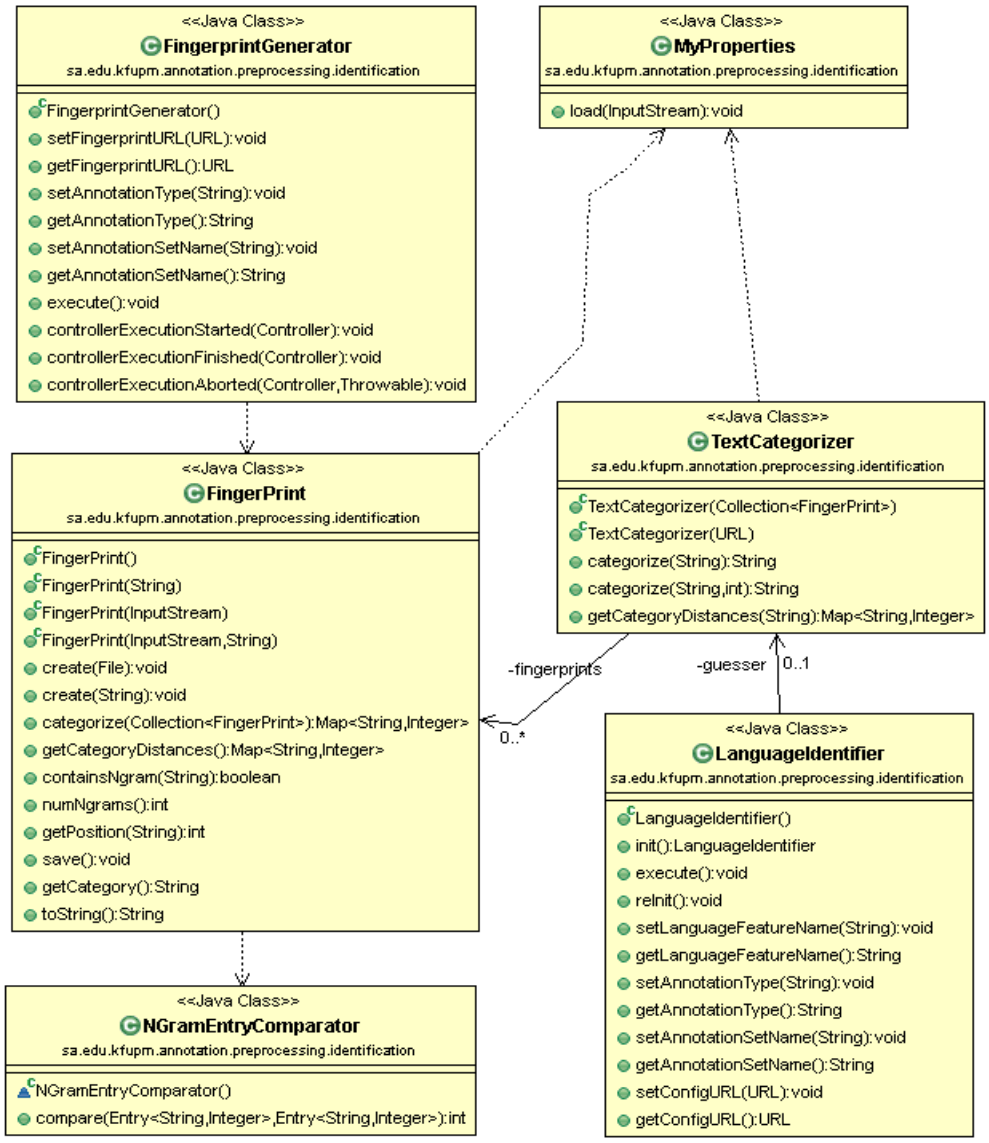



**Figure 7.9 Class Diagram for language identification**

## 7.4 Recognition Layer

The recognition layer contains the core components of the semantic annotation process. It links the entities and their semantic relationships, discovered in the Web document, with domain ontologies. The linking is done in two steps, starting with entity recognition and then examining the relationship between them. The two tasks will be demonstrated in the

next subsections. Figure 7.10 shows the abstract classes for entity and relationship recognition. The *Recognizer* class is an abstract class that represents objects that can be used to annotate documents. In general, the defining property of a *Recognizer* is the tag that is assigned to the annotations produced in the document. The method that must be used for a *Recognizer* to annotate a document is the *enrich* method. The *EntityRecognizer* is an abstract entity recognition class that is an extension of *Recognizer*. Instances of this class are able to annotate entities that are present in the text. The *RelationshipRecognizer* is an abstract relationship recognition class that is an extension of *Recognizer*. Instances of this class are able to annotate relationship between entities that are present in the text.

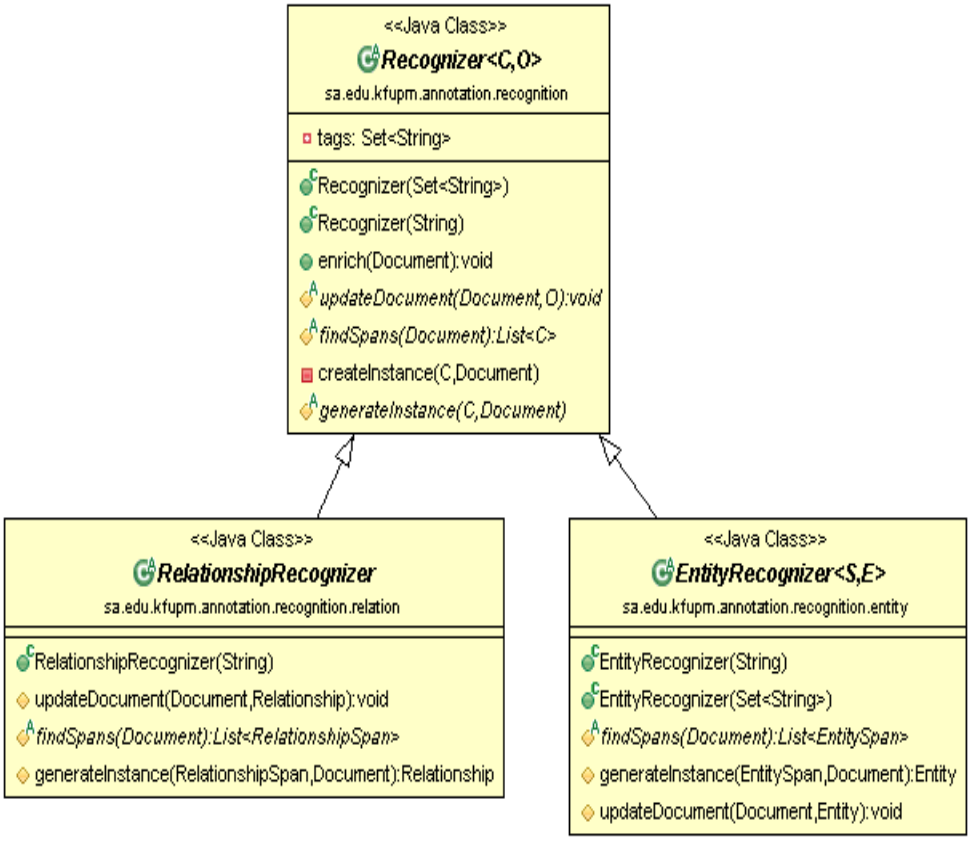



**Figure 7.10 Recognition Classes**

### 7.4.1 Domain Entity Recognition

Domain entity recognition is the process of identifying and classifying atomic texts into predefined domain ontological classes. The recognition process starts with the discovery step, followed by the resolution step. The discovery step detects which part of the text refers to ontological concepts and could be performed using different approaches such as the gazetteer (dictionary-based), rule-based, or machine-learning approach. The resolution is to match the detected entity with the most likely ontology concept. The resolution could be handled by classification methods that consider the context of the matched phrase. Different algorithms have been used in this task. The focus for this framework is to have a language-independent approach to handle both tasks.

#### 7.4.1.1 Dictionary Matching

Domain entity recognition using dictionary is implemented using a set of classes. The main class is the DictionaryBasedEntityRecognizer which makes use of LingPipe's31 ExactDictionaryChunker to build a dictionary and match for a given text with dictionary entries. The DictionaryBasedEntityRecognizer extends EntityRecognizer and corresponds to an implementation of the Aho-Corasick dictionary matching algorithm described in [75]. For each language, we build a separate dictionary containing all domain concept names with their synonyms in a single dictionary. The key of each dictionary entry is a combination of concept name and concept unique ID. To recover the full URL from the key, a prepopulated list of concept name and complete URL is built from the given set of ontologies.

[31] http://alias-i.com/lingpipe

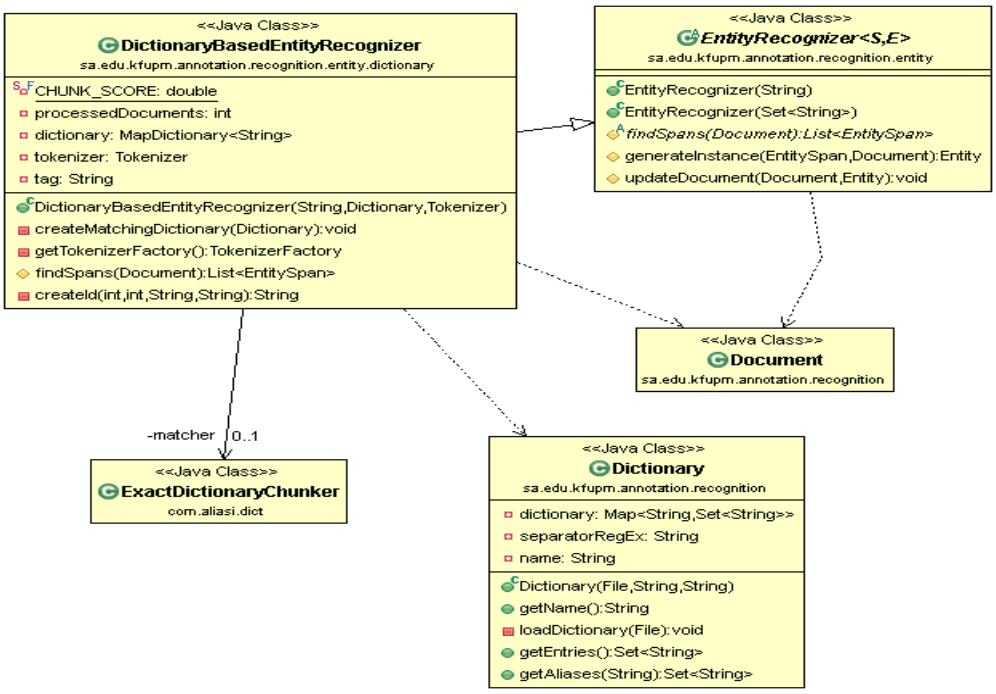



**Figure 7.11 Class Diagram of Dictionary Entity Recognizer**

### 7.4.1.2 Rule Based Domain Entities Recognition

We adapted GATE JAPE engine which provides a rule language over NLP annotation and regular expression over the text. Rules are either created by hand or semi-automatically created using sample data. We have created a set of rules for each entity class: Food, Nutrition, Disease, Body Part, Body Function. We have used JAPE rules to represent rule patterns. In order to extract domain-named entities using the rule base, we have rules for each entity class: food, nutrition, disease, body part, and body function. Figure 7.13 shows the class diagram of entity recognition using rule-based recognition. The main class, RuleBasedEntityRecognizer, uses a set of JAPE rules for each entity type. The left-hand side of the rule is the pattern to search for; if found, the action in the right-hand side is executed. The left-hand side is expressed in regular expression format over annotation done in the input text, including NLP processing annotation such as POS and morphological processing. A sample JAPE rule for disease entity is shown in Figure 7.12.

```
Rule: Disease1
Priority: 50
(
  ({Token.category == "NN"}|{Token.category == "NNP"}):disease_name
  {Token.category == "disease"}
):desease_ann
-->
: desease_ann.Entity = {string = disease_name, rule = Disease1}
```

**Figure 7.12 Sample Disease Rule used by Rule Based Entity Recognizer**

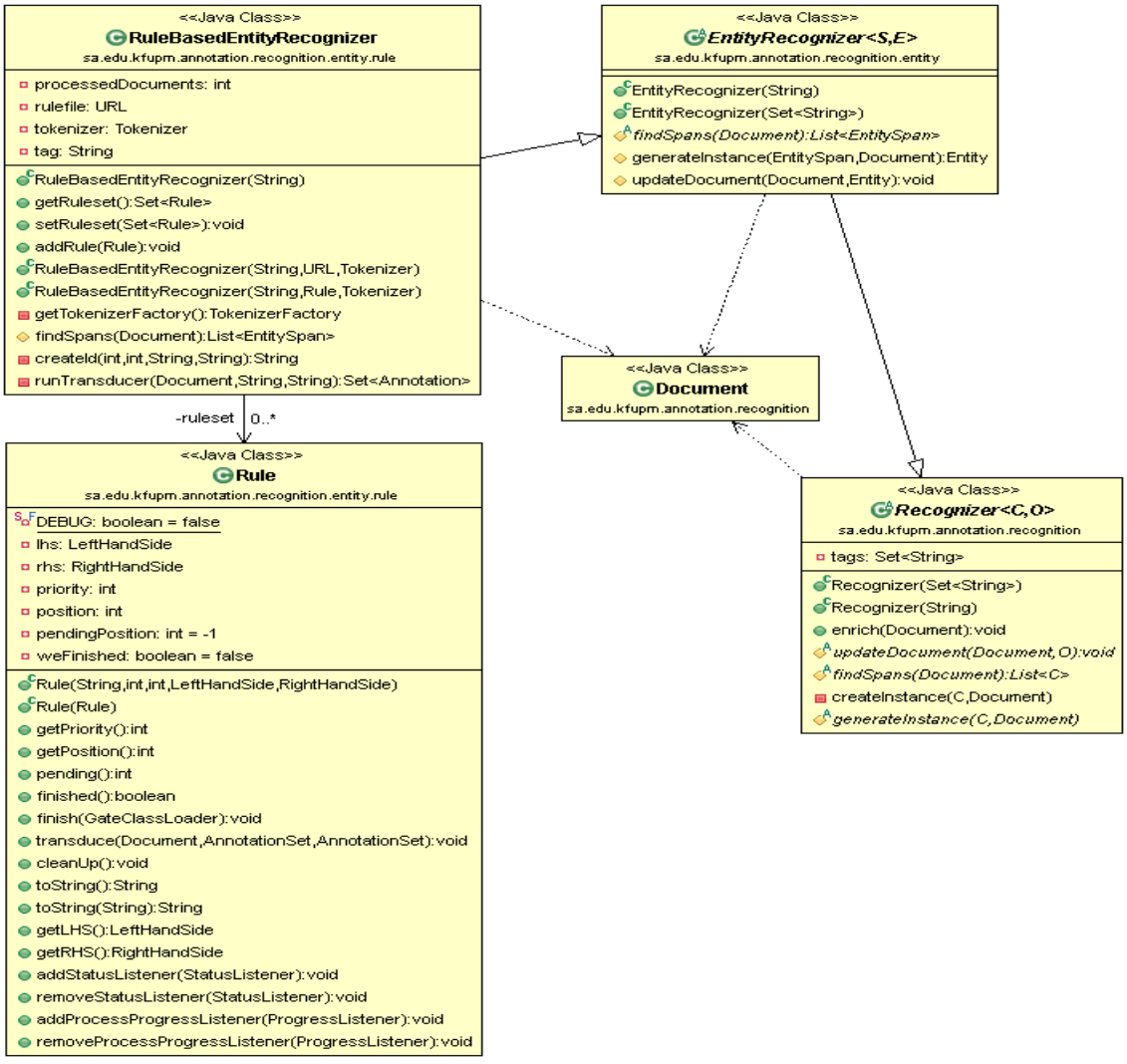



**Figure 7.13 Class Diagram for Rule Based Entity Recognizer**

### 7.4.1.3 Machine Learning Based Domain Entities Recognition

The current implementation of the ML recognition is using support vector machines (SVM) as shown in Figure 7.14. The MachineLearningEntityRecognizer is the main class for entity recognition and it is based on Learning API provided by GATE. To use the class, the user has to supply a configuration file and train the model. The configuration file contains the parameters for SVM engine and feature set. The trained model contains a binary file of the training conducted using offline mode based on preannotated sample documents. A sample configuration file for the SVM entity recognizer is shown in Figure 7.15. The same file is used for both training and recognition. The feature set configure in the file needs to be provided for the model to work as trained.

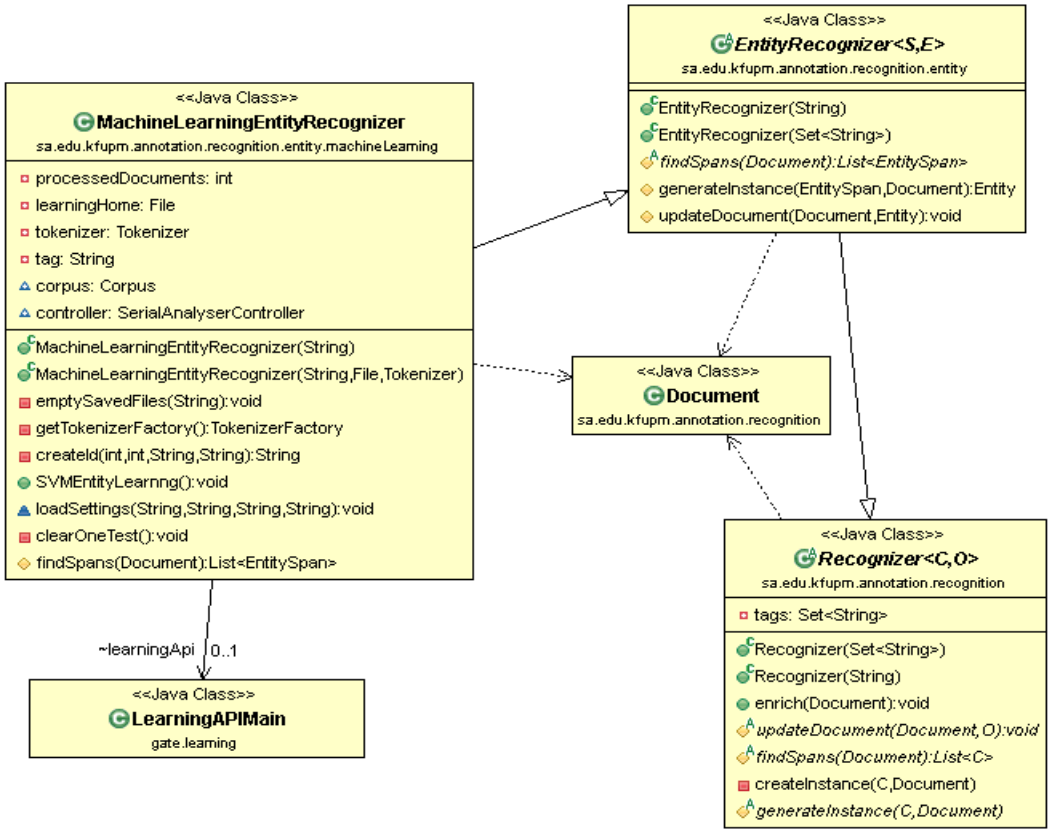



**Figure 7.14 Class Diagram of Machine Learning Entity Recognizer**

```
<?xml version="1.0"?>
- <ML-CONFIG>
    <VERBOSITY level="0"/>
    <FILTERING dis="far" ratio="0.1"/>
    <SURROUND value="true"/>
    <PARAMETER value="0.3" name="thresholdProbabilityEntity"/>
    <PARAMETER value="0.50" name="thresholdProbabilityBoundary"/>
    <PARAMETER value="0.5" name="thresholdProbabilityClassification"/>
    <multiClassification2Binary method="one-vs-others"/>
    <EVALUATION ratio="0.66" method="split" runs="2"/>
    <ENGINE options=" -c 0.7 -t 0 -m 100 -tau 0.6 " implementationName="SVMLibSvmJava" nickname="SVM"/>
  - <DATASET>
      <INSTANCE-TYPE>Token</INSTANCE-TYPE>
      <WINDOWSIZE windowSizeRight="5" windowSizeLeft="5"/>
    - <ATTRIBUTELIST>
        <NAME>Form</NAME>
        <SEMTYPE>NOMINAL</SEMTYPE>
        <TYPE>Token</TYPE>
        <FEATURE>string</FEATURE>
        <RANGE to="5" from="-5"/>
      </ATTRIBUTELIST>
    + <ATTRIBUTELIST>
    - <ATTRIBUTELIST>
        <NAME>Tokenkind</NAME>
        <SEMTYPE>NOMINAL</SEMTYPE>
        <TYPE>Token</TYPE>
        <FEATURE>kind</FEATURE>
        <RANGE to="5" from="-5"/>
      </ATTRIBUTELIST>
    - <ATTRIBUTELIST>
        <NAME>Lemma</NAME>
        <SEMTYPE>NOMINAL</SEMTYPE>
        <TYPE>Token</TYPE>
        <FEATURE>root</FEATURE>
        <RANGE to="5" from="-5"/>
      </ATTRIBUTELIST>
    + <ATTRIBUTELIST>
    - <ATTRIBUTELIST>
        <NAME>EntityType</NAME>
        <SEMTYPE>NOMINAL</SEMTYPE>
        <TYPE>Entity</TYPE>
        <FEATURE>type</FEATURE>
        <RANGE to="5" from="-5"/>
      </ATTRIBUTELIST>
    - <ATTRIBUTE>
        <NAME>Class</NAME>
        <SEMTYPE>NOMINAL</SEMTYPE>
        <TYPE>Mention</TYPE>
        <FEATURE>class</FEATURE>
        <POSITION>0</POSITION>
        <CLASS/>
      </ATTRIBUTE>
    </DATASET>
</ML-CONFIG>
```

**Figure 7.15 Sample SVM Entity Recognition Configuration file**

### 7.4.2 Relation Recognition

As mentioned earlier, relationship recognition is the process of discovering a relationship between two or more entities based on the types of relationship defined in the ontology. As there are different approaches for relationship recognition, there are different implementation options for each approach. The current implementations of the relationship recognitions approaches are based on GATE platform in addition to other toolkits. Figure 7.16 shows the base class diagram for the relationship recognition. The *RelationshipRecognizer* is an abstract class. The class *Relationship* represents

relationships between several entities present in a document. A relationship is defined by its type which is an instance of a *RelationshipType*. Additionally, a relationship contains a set of entities that may be related or not, and each of these entities fulfills a role in the relationship. The class *RelationshipType* represents a type of relationship that we are interested in, and it contains more than one label indicating the semantics of the relationship. That is, a relationship type may also contain some constraints imposed over rules, the relationship itself, and the entities that can be matched. These constraints are used to check whether a given relationship is valid.

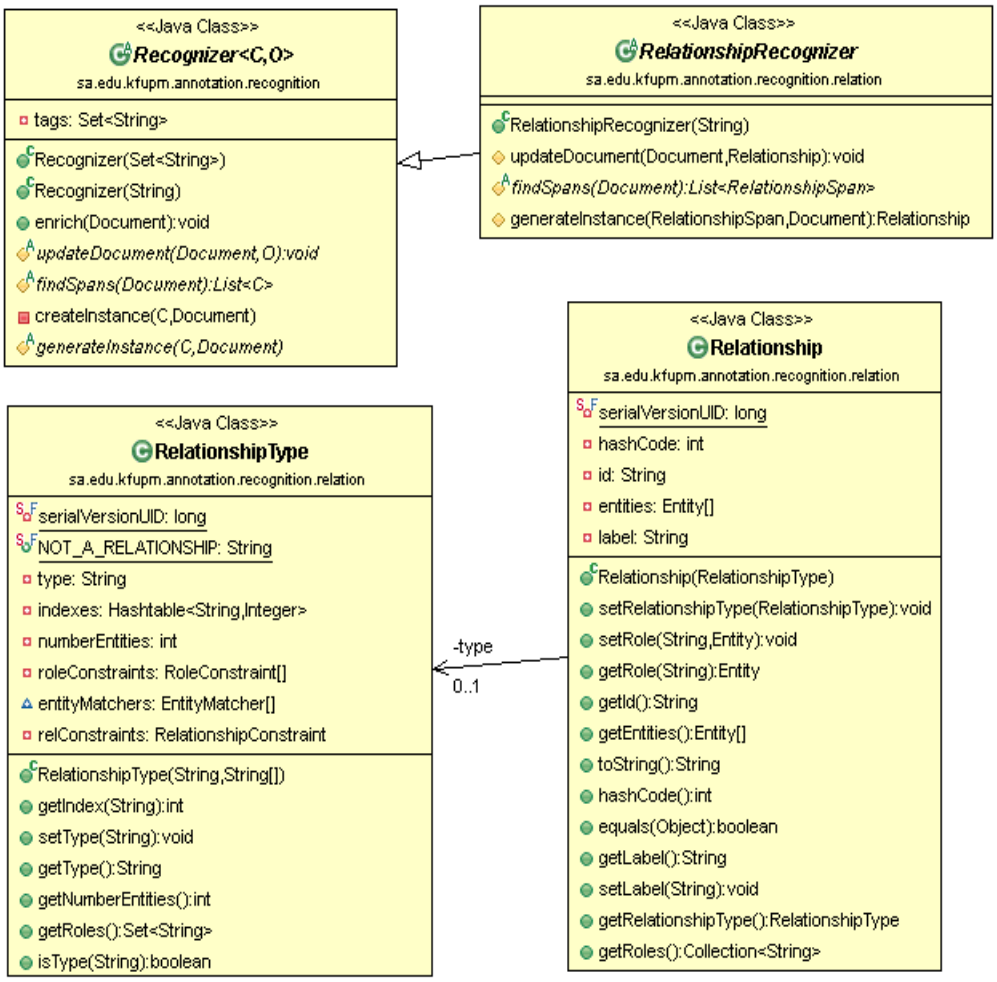



**Figure 7.16 Relation Recognition Base Class Diagram**

#### 7.4.2.1 Relation Recognition Using Trigger Words

Relation discovery using trigger words is a naïve approach for relationship detection between name entities in a sentence. It is considered as a simple approach that does not require even minimal NLP processing but depends heavily on string matching. Figure 7.17 shows the class diagram for relationship recognition using trigger words. The class TriggerWordsRelationshipRecognizer is the main class for this type of recognition. The user has to supply a set of trigger words for each type of relationship. This class assumes that a given document is already processed by an entity recognizer. This relationship recognizer searches for these trigger words in the document text. If one is found, then it will check for recognized entities in the text around this trigger word (i.e., in the same sentence) and build a relationship between these entities. The validity of the newly discovered relationship is left for the validation task.

To generate trigger words, the set of supplied ontologies is traversed for all object properties. For each property, different annotations are used to capture the name of the relationship between entities by applying some normalization of the textual value of the properties.

.

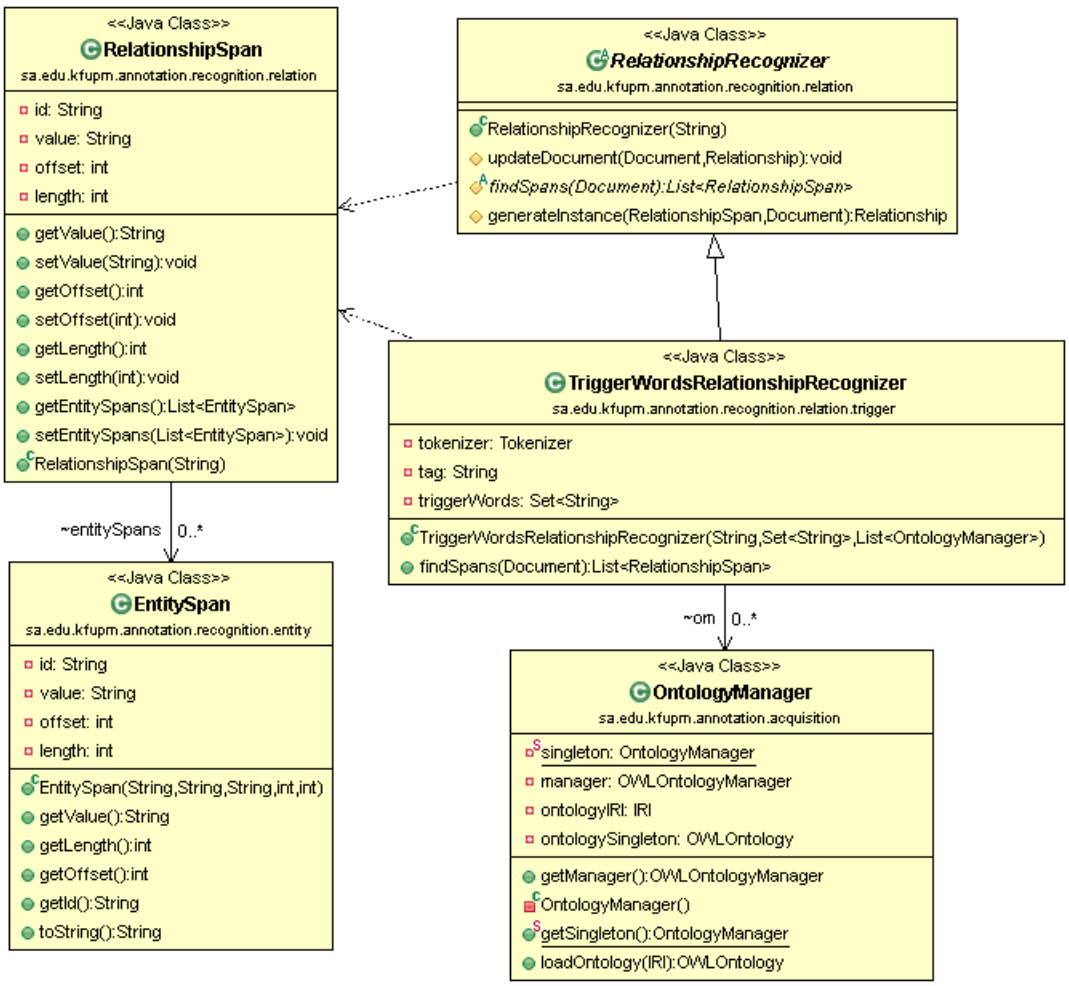



**Figure 7.17 Class Diagram for Relation Recognition Using Trigger Word**

### 7.4.2.2 Rule Based Domain Relation Recognition

We adapted the GATE JAPE engine, which provides a rule language over NLP annotation and regular expression over the text. Rules are either created by hand or semiautomatically created using sample data. We have created a set of rules for each relation type: prevent, treat, cause, good for, and bad for. We have used JAPE rules to represent rule patterns. In order to recognize the relationship between entities using rule base, we created set of JAPE rules for each relation.

Figure 7.19 shows the class diagram of entity recognition using rule-based recognition. The main class *RuleBasedEntityRecognizer* uses a set of JAPE rules for each entity type. The left-hand side of the rule is the pattern to search for; if it is found, the action on the right-hand side is executed. The left-hand side is expressed in regular expression format over the annotation done in the input text, including NLP processing annotation such as POS and morphological processing. Two sample JAPE rules for prevent relation are shown in Figure 7.18. The first rule is to detect the relationship in the text, and the second rule is to build the relationship between different entities around the relationship phrase.

```
Rule: Prevent_Relation_1
Input: Token
(
 (({Token.string ==~ "(can|could|may|might)"}) ({Token})[0,1])?
 (({Token.string ==~ "^(help|helps)"}) ({Token})[0,1] )?
 (({Token.string ==~ "(in|to|)"})      ({Token})[0,1] )?
 {Token.string ==~ "^(prevent|protect)"}
 (({Token.string ==~ "(a|the)"})
  ({Token})[0,1]  ({Token.string ==~ "^(grow|develop)"}))?
) →
{
   FeatureMap fm = Factory.newFeatureMap();
   fm.put("propertyName", "prevent");
   fm.put("rule", "Relation_Rule_Prevent");
   outputAS.add(sentAS.firstNode(),sentAS.lastNode(),"Relation", fm);
}

Rule: Single_Relation_1
Input: foodItem nutrient disease bodyPart bodyFunction Relation Split
 (
  ({foodItem}|{nutrient})+:domain ({Relation}):rel1
  ({disease}|{bodyPart}|{bodyFunction})+:range {Split}
) →
{
  for(Annotation d : domain) {
   for(Annotation r : range) {
     FeatureMap fm = Factory.newFeatureMap();
     fm.put("propertyName", rel1);
     fm.put("domain-id",d.getId().toString());
     fm.put("range-id" ,r.getId().toString());
     outputAS.add(d.getStartNode(),r.getEndNode(),"RelInstance",fm);
    }
}
```

**Figure 7.18 Sample Prevent Rule used by Rule Based Relation Recognizer**

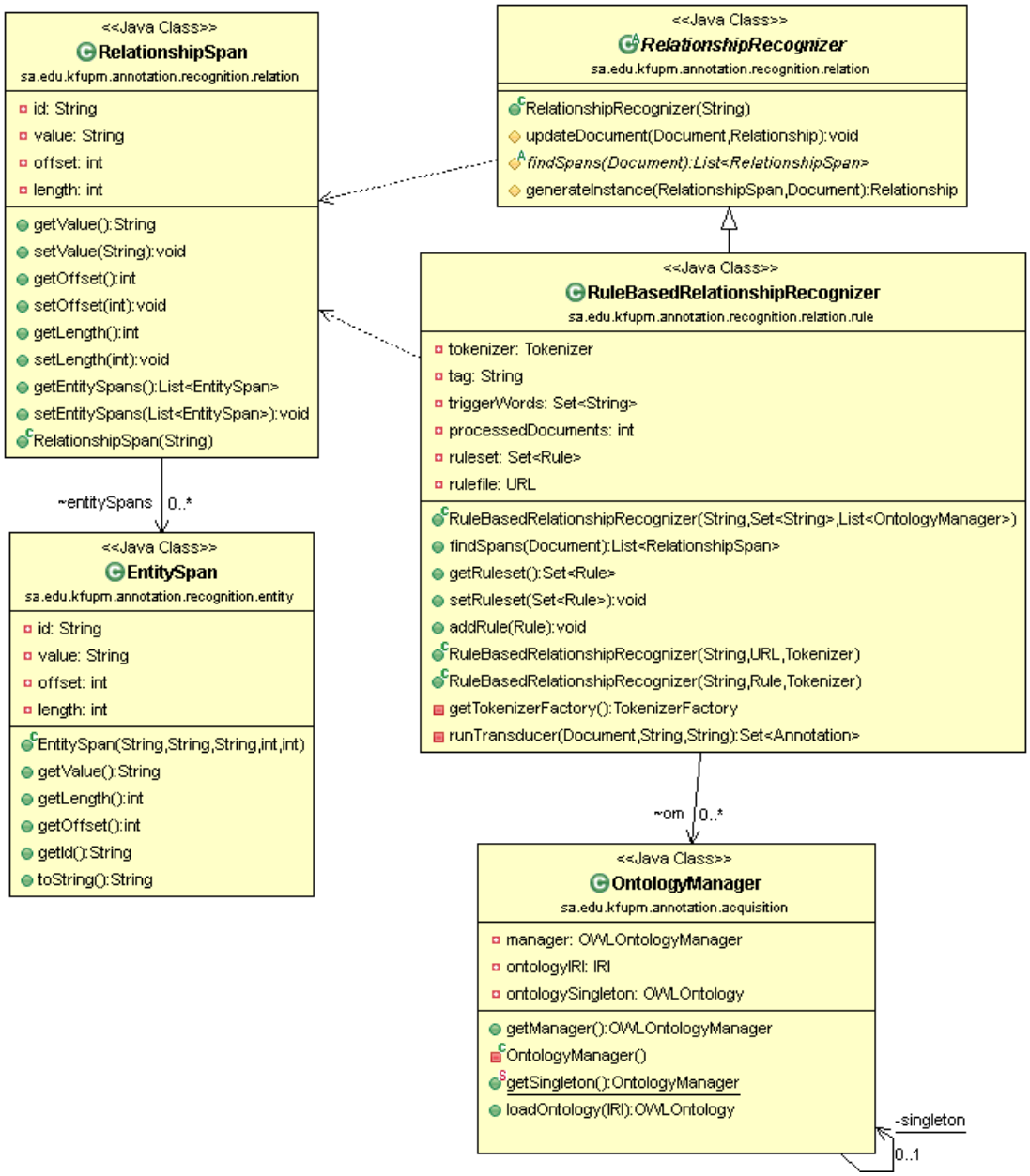



**Figure 7.19 Class Diagram for Rule Based Entity Recognizer**

### 7.4.2.3 Entity-Predicate Based Relation Recognition

For advanced relationship recognition, the entity-predicate pair (EPP) method is used which parses the sentence to capture subject, action, and object from the parse tree. The first step, dependency parser, is used to generate the parse tree. Then, a subject-action-object extractor is based on a set of rules to be customized for a given parser and

language. These rules determine which part of the parser output is considered the subject, verb, and object.

In the current implementation of this recognizer, we have used the Stanford parser to generate dependency parse tree and set of rules are customized to access the subject, verb, and object in a given sentence.

Figure 7.20 shows the class diagram for the EPP-based relationship recognizer approach. The class *EntityPredicateRelationshipRecognizer* processes the document sentences and generates tuples of <subject, predicate, object> where the subject and object belong to ontology concepts and instances. For each sentence, the parser generates a dependency parse tree using the *ParseTreeRuleEngine* class. A set of rules is used based on the complexity of the parse tree to generate the subject, object, and predicate. The process starts by checking the reification flag. if it is true, then it breaks the parse-tree about the main predicate and finds the main subject of the sentence. Then, it checks whether the graph has clausal (*conj_and*) structure or not. The complete logic of the generated tuples out of textual sentence is coded in the *ParseTreeRuleEngine.*

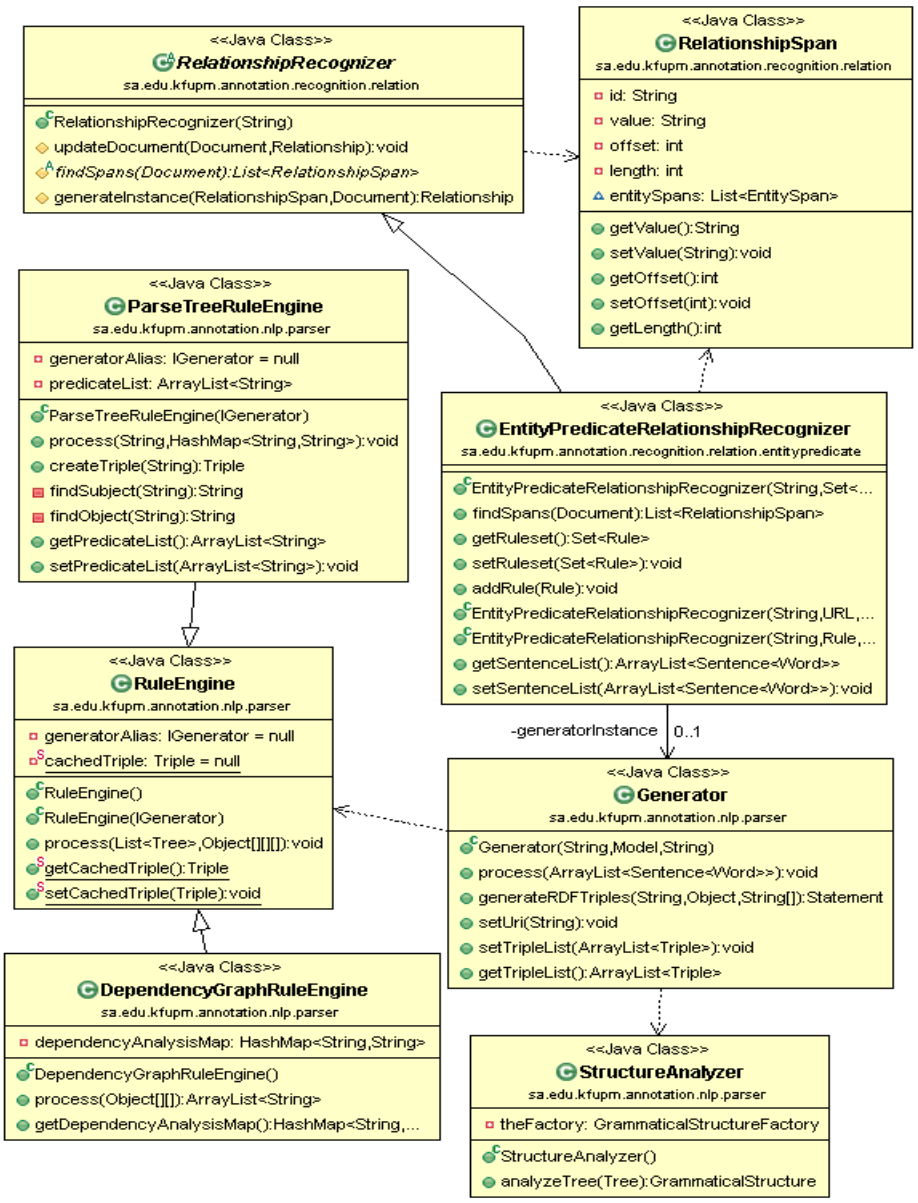



**Figure 7.20 Class Diagram for Entity-Predicate Based Relation Recognizer**

### 7.4.2.4 Machine Learning Based Relation Recognition

The current implementation of the ML recognition is done using SVM as shown in Figure 7.22. The main class for entity recognition is MachineLearningRelationRecognizer, and it is based on Learning API provided by

GATE. To use the class, the user has to supply a configuration file and train the model. The configuration file contains the parameters for SVM engine and feature set. The trained model contains a binary file of the training conducted using offline mode based on pre-annotated sample documents. A sample configuration file for SVM relationship recognizer is shown in Figure 7.21 for the relationship with two arguments. The same file is used for both training and recognition. The feature set configuration in the file needs to be provided for the model to work for training and recognition tasks.

```
<?xml version="1.0"?>
<ML-CONFIG>
   <VERBOSITY level="0"/>
   <SURROUND value="false"/>
   <FILTERING dis="near" ratio="0.0"/>
   <PARAMETER value="0.2" name="thresholdProbabilityEntity"/>
   <PARAMETER value="0.42" name="thresholdProbabilityBoundary"/>
   <PARAMETER value="0.5" name="thresholdProbabilityClassification"/>
   <multiClassification2Binary method="one-vs-another"/>
      <!-- Evaluation : how to split the corpus into test and learn? -->
   <EVALUATION ratio="0.66" method="split" runs="1"/>
   <ENGINE options=" -c 0.7 -t 0 -m 100 -tau 0.6"
           implementationName="SVMLibSvmJava" nickname="SVM"/>
 - <DATASET>
      <INSTANCE-TYPE>RE_INS</INSTANCE-TYPE>
      <INSTANCE-ARG1>arg1</INSTANCE-ARG1>
      <INSTANCE-ARG2>arg2</INSTANCE-ARG2>
    - <FEATURES-ARG1>
       - <ARG>
            <NAME>ARG1</NAME>
            <SEMTYPE>NOMINAL</SEMTYPE>
            <TYPE>foodItem</TYPE>
            <FEATURE>MENTION_ID</FEATURE>
         </ARG>
       - <ATTRIBUTE>
            <NAME>Form</NAME>
            <SEMTYPE>NOMINAL</SEMTYPE>
            <TYPE>Token</TYPE>
            <FEATURE>string</FEATURE>
            <POSITION>0</POSITION>
         </ATTRIBUTE>
       - <ATTRIBUTE>
            <NAME>POS</NAME>
            <SEMTYPE>NOMINAL</SEMTYPE>
            <TYPE>Token</TYPE>
            <FEATURE>category</FEATURE>
            <POSITION>0</POSITION>
         </ATTRIBUTE>
      </FEATURES-ARG1>
    - <FEATURES-ARG2>
       - <ARG>
            <NAME>ARG2</NAME>
            <SEMTYPE>NOMINAL</SEMTYPE>
            <TYPE>disease</TYPE>
            <FEATURE>MENTION_ID</FEATURE>
         </ARG>
       - <ATTRIBUTE>
            <NAME>Form</NAME>
            <SEMTYPE>NOMINAL</SEMTYPE>
            <TYPE>Token</TYPE>
            <FEATURE>string</FEATURE>
            <POSITION>0</POSITION>
         </ATTRIBUTE>
       - <ATTRIBUTE>
            <NAME>POS</NAME>
            <SEMTYPE>NOMINAL</SEMTYPE>
            <TYPE>Token</TYPE>
            <FEATURE>category</FEATURE>
            <POSITION>0</POSITION>
         </ATTRIBUTE>
      </FEATURES-ARG2>
    - <ATTRIBUTE_REL>
         <NAME>EntityCom1</NAME>
         <SEMTYPE>NOMINAL</SEMTYPE>
         <TYPE>RE_INS</TYPE>
         <ARG1>arg1</ARG1>
         <ARG2>arg2</ARG2>
         <FEATURE>t12</FEATURE>
         <POSITION>0</POSITION>
      </ATTRIBUTE_REL>
    - <ATTRIBUTE_REL>
         <NAME>EntityCom2</NAME>
         <SEMTYPE>NOMINAL</SEMTYPE>
         <TYPE>RE_INS</TYPE>
         <ARG1>arg1</ARG1>
         <ARG2>arg2</ARG2>
         <FEATURE>s12</FEATURE>
         <POSITION>0</POSITION>
      </ATTRIBUTE_REL>
    - <ATTRIBUTE_REL>
         <NAME>Class</NAME>
         <SEMTYPE>NOMINAL</SEMTYPE>
         <TYPE>prevent</TYPE>
         <ARG1>MENTION_ARG1</ARG1>
         <ARG2>MENTION_ARG2</ARG2>
         <FEATURE>Relation_type</FEATURE>
         <POSITION>0</POSITION>
         <CLASS/>
      </ATTRIBUTE_REL>
   </DATASET>
</ML-CONFIG>
```

**Figure 7.21 Sample SVM Relation Recognition Configuration file**

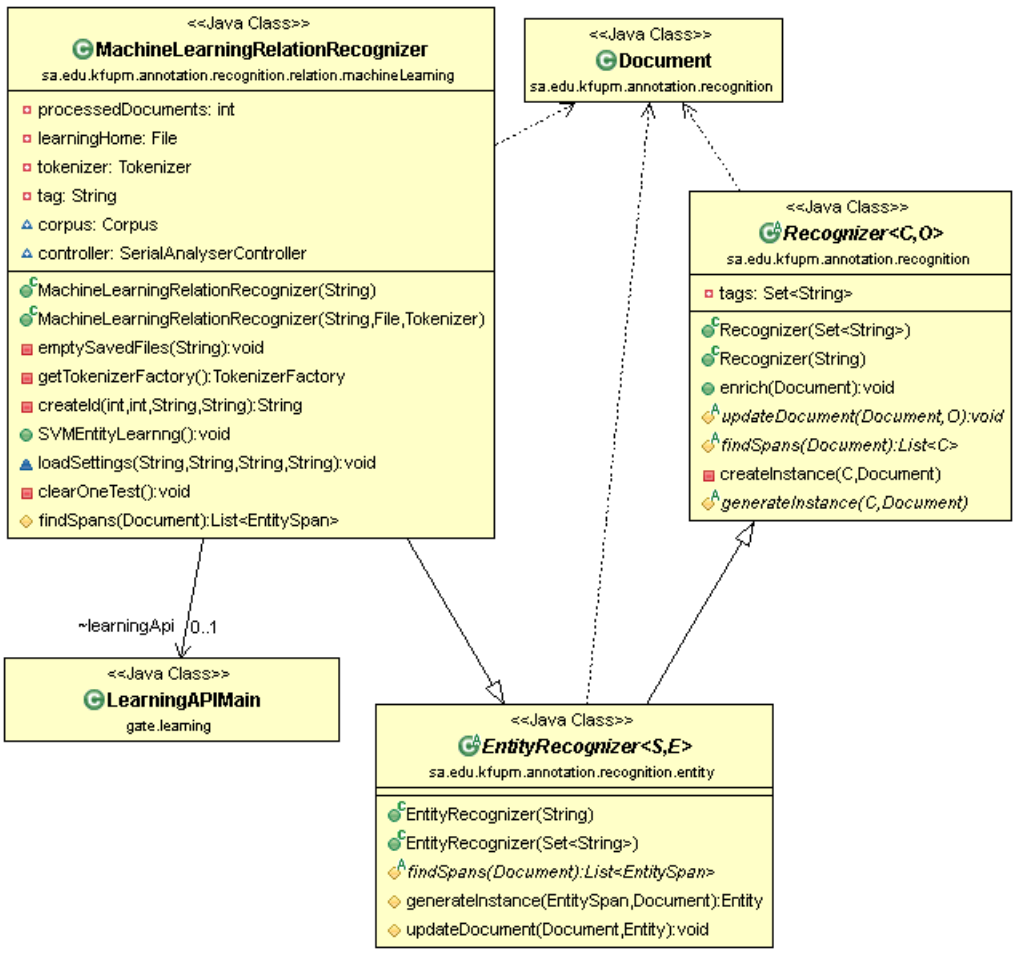



**Figure 7.22 Class Diagram of Machine Learning Relation Recognizer**

## 7.5 Post-Processing Layer

The main role of the postprocessing layer is to finalize the annotation task by validating the annotated information and storing the annotation into persistence storage. In this prototype, we implemented three processes: validation, file storage, and repository storage. In the next sections, we will examine more details about those processes.

### 7.5.1 Validation

In the validation task, the recognized entities and relationships are validated against the domain ontologies and instances. Recognized entities are valid if they have corresponding instances in the knowledge base. Similarly, the recognized relationships are valid if the relationship exists in the domain ontologies and the entity pair matches the type of the domain and range of the object properties defined in the ontologies.

The validation approach of entities and relation of a tuple <subject,predicate,object> against a domain ontology with instances could be summarized in the following steps:

1. Get the superset of concepts for subject and object instances from ontologies through the is-a relation.
2. Get all possible object properties between subject superset concepts and object superset concepts. Search on the collected properties for the predicate.
3. If they are matched, then check if there is any restriction on the domain and range for matched properties on the superset concepts of subject and object.

If all these conditions hold, create a new RDF statement for the relationship between subject and object, and then add it to the set of validated RDF statement in the knowledge base.

Figure 7.23 shows the class diagram for the validation of recognized item. The class *recognitionValidation* is initialized with domain ontologies to validate against. The validation is performed by comparing ontologies' concepts and instances with recognized entities and then comparing the allowable relationship between them if it covers the recognized relationship or not. The *recognitionValidation* class returns a list of valid relationships for a given document by calling the *validate* function.

The *recognitionValidation* class can be used to aggregate the recognized resources by calling the *aggregate* function. It will return a distinct contract from the given document. This aggregation is helpful for many applications, and it removes all redundant information that is not required for semantic Web application as compared to traditional Web retrieval.

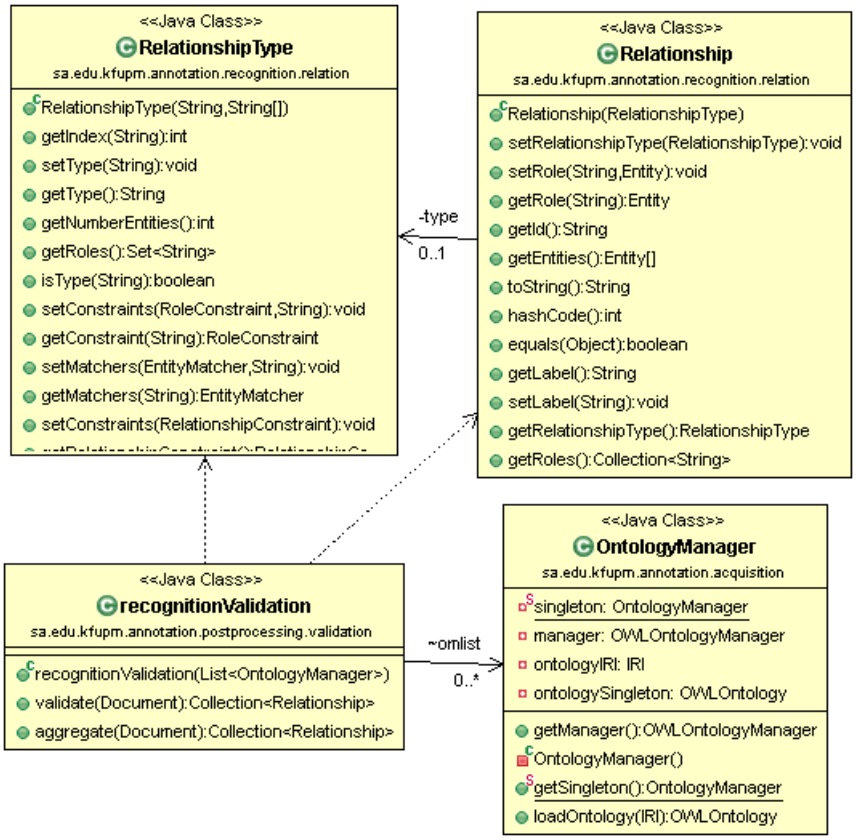



**Figure 7.23 Class Diagram for Validation**

### 7.5.2 Annotation Writer

The recognized entities and relationships of Web documents are stored in external annotation files using preferred standard languages such as RDFa, RDF, N3, or Turtle. Figure 7.24 shows the class diagram of the *fileStorage* annotation writer. To generate the

target annotation, the class accesses different software libraries as well as domain ontologies.

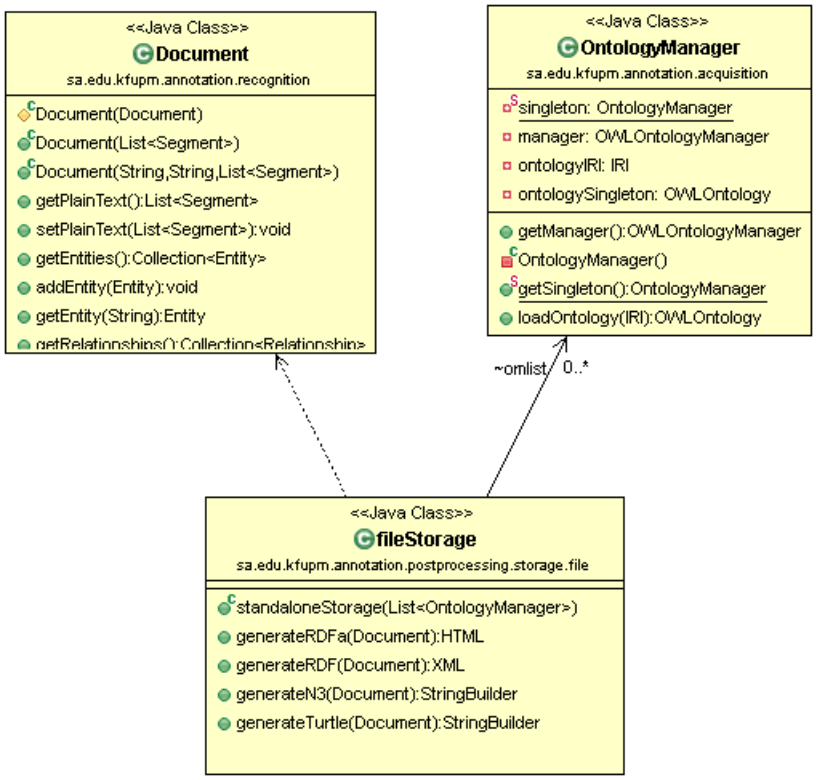



**Figure 7.24 Class Diagram for Standalone Annotation Storage**

### 7.5.3 Knowledge Base Enrichment

The role of the knowledge base enrichment task is to insert the recognized entities and relationships between them as well as the source information into the knowledge base repository. This task is configured to connect to one of the common tuple repositories, such as Sesame.[32] The MLSAF framework provides an adapter to such a repository, and the developer could easily customize a new adapter for others. Figure 7.25 shows the class diagram for generic and sesame repository adapters

[32] http://www.openrdf.org

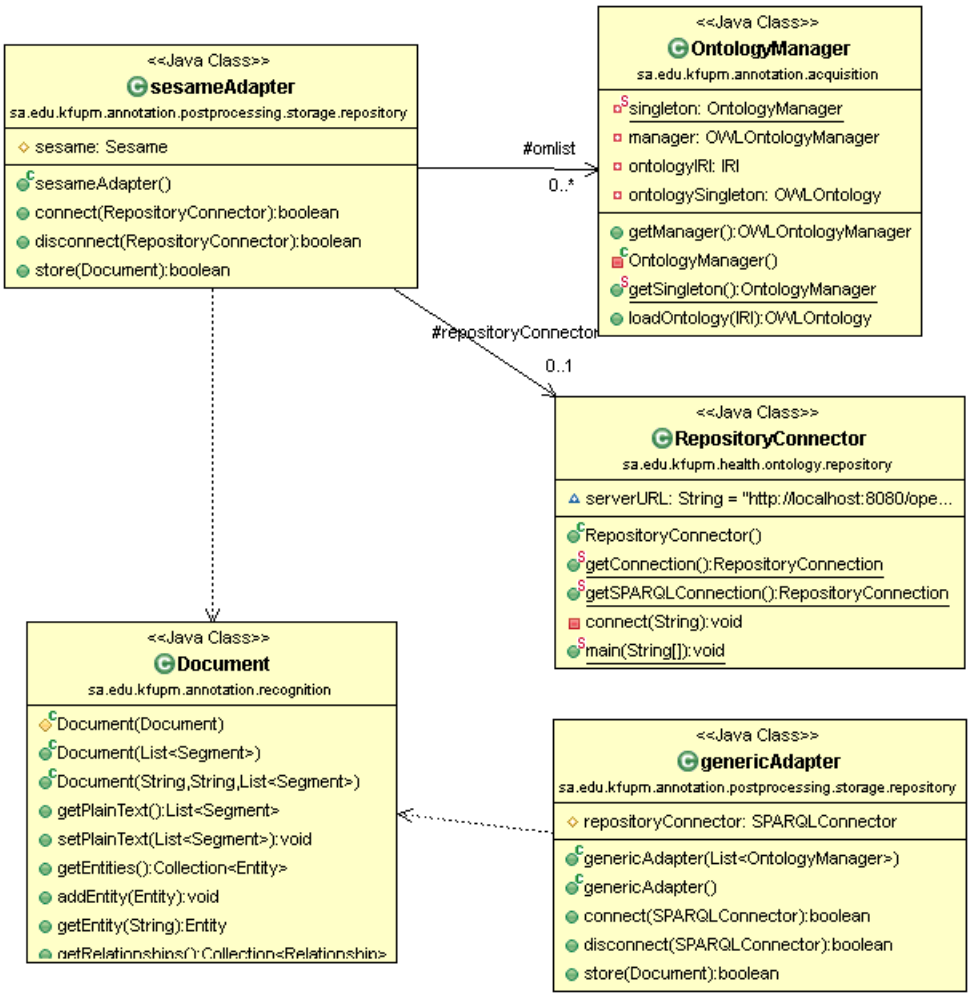



**Figure 7.25 Class Diagram for Repository Adapter**

## 7.6 Annotation Interface

The annotation process starts with the acquisition process and ends with post-processing process. This workflow is model as information pipeline where the input of each task is the output of the pervious task.

In the annotator implementation, GATE [73] is used for NLP processing, and Jena [76] is used to generate RDF annotation of the extracted entities and relationships according to the domain ontologies.

The next step is to analyze the document in order to identify the useful segments of the document. Unstructured text segments are then sent to the NLP pipeline, and structured segments are sent to data frame pipeline. In the current implementation of the annotator, we only implemented the NLP pipeline. Figure 7.26 shows a snapshot of the annotation tool that takes source folders containing the Web documents collected by the crawler. The tool stores the RDF files containing as external annotation for each document in the target folder.

Figure 7.27 shows an example of an HTML document containing unstructured text about the health benefits of brown rice. The facts of interest are the relationships between the brown rice and bone health and between manganese and bone health. The system was able to discover both relationships. Brown rice is an instance of a food item concept with an ID of I20036,[33] and bone is an instance of a body part with an ID of BP004.[34] Manganese is a nutrient with ID 315.[35] Part of the RDF file generated by the system is shown in Figure 7.28.

[33] URI: http://www.kfupm.edu.sa/ontology/food/foodItem/I20036
[34] URI: http://www.kfupm.edu.sa/ontology/health/bodyPart/BP004
[35] URI: http://www.kfupm.edu.sa/ontology/food/nutrients/315

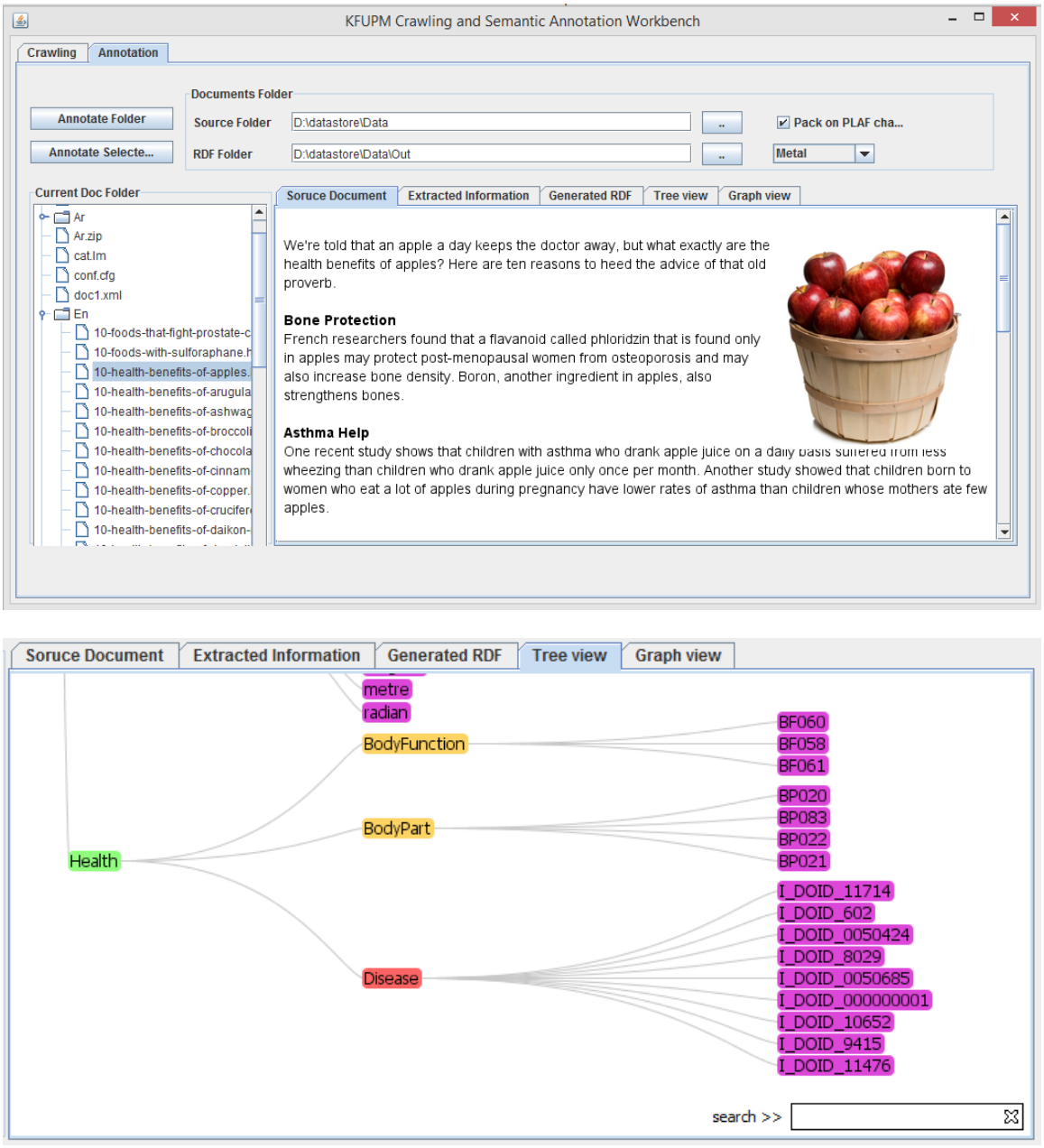


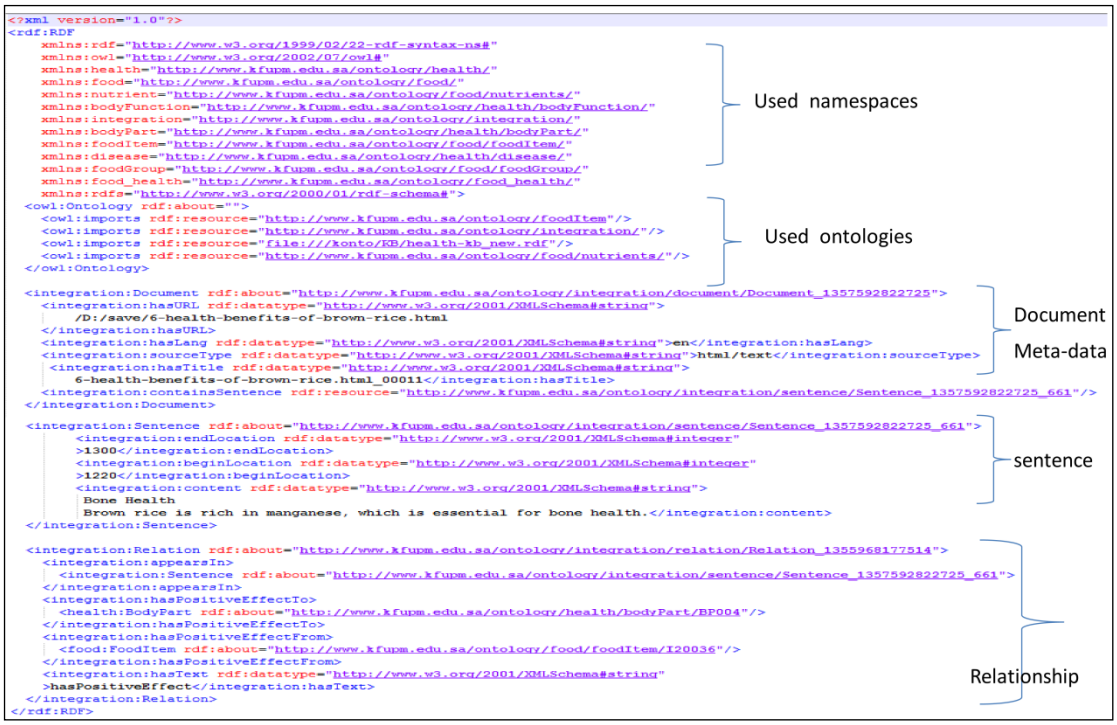

**Figure 7.26 Annotation Tool Screen Snapshot**

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "
http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta name="description" content="Discover 6 health benefits of brown rice,
from cancer prevention to heart health." />
<meta name="keywords" content="health benefits of brown rice" />
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />

<title>6 Health Benefits of Brown Rice</title>
<h1>6 Health Benefits of Brown Rice</h1>
</head>
<body>
<p>
Brown rice has more nutrition than white rice, which is lacking in nutrition
because it has been processed to remove the endosperm, hull, and pretty much
everything that's good for you.
</p>
<p><strong>Bone Health</strong><br />
Brown rice is rich in manganese, which is essential for bone health.</p>
</body>
</html>
```

**Figure 7.27 Part of HTML page of Brown Rice**




```
<?xml version="1.0"?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:health="http://www.kfupm.edu.sa/ontology/health/"
    xmlns:food="http://www.kfupm.edu.sa/ontology/food/"
    xmlns:nutrient="http://www.kfupm.edu.sa/ontology/food/nutrients/"
    xmlns:bodyFunction="http://www.kfupm.edu.sa/ontology/health/bodyFunction/"
    xmlns:integration="http://www.kfupm.edu.sa/ontology/integration/"
    xmlns:bodyPart="http://www.kfupm.edu.sa/ontology/health/bodyPart/"
    xmlns:foodItem="http://www.kfupm.edu.sa/ontology/food/foodItem/"
    xmlns:disease="http://www.kfupm.edu.sa/ontology/health/disease/"
    xmlns:foodGroup="http://www.kfupm.edu.sa/ontology/food/foodGroup/"
    xmlns:food_health="http://www.kfupm.edu.sa/ontology/food_health/"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
  <owl:Ontology rdf:about="">
    <owl:imports rdf:resource="http://www.kfupm.edu.sa/ontology/foodItem"/>
    <owl:imports rdf:resource="http://www.kfupm.edu.sa/ontology/integration/"/>
    <owl:imports rdf:resource="file:///konto/KB/health-kb_new.rdf"/>
    <owl:imports rdf:resource="http://www.kfupm.edu.sa/ontology/food/nutrients/"/>
  </owl:Ontology>

  <integration:Document rdf:about="http://www.kfupm.edu.sa/ontology/integration/document/Document_1357592822725">
    <integration:hasURL rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
        /D:/save/6-health-benefits-of-brown-rice.html
    </integration:hasURL>
    <integration:hasLang rdf:datatype="http://www.w3.org/2001/XMLSchema#string">en</integration:hasLang>
    <integration:sourceType rdf:datatype="http://www.w3.org/2001/XMLSchema#string">html/text</integration:sourceType>
     <integration:hasTitle rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
        6-health-benefits-of-brown-rice.html_00011</integration:hasTitle>
    <integration:containsSentence rdf:resource="http://www.kfupm.edu.sa/ontology/integration/sentence/Sentence_1357592822725_661"/>
  </integration:Document>

  <integration:Sentence rdf:about="http://www.kfupm.edu.sa/ontology/integration/sentence/Sentence_1357592822725_661">
        <integration:endLocation rdf:datatype="http://www.w3.org/2001/XMLSchema#integer"
        >1300</integration:endLocation>
        <integration:beginLocation rdf:datatype="http://www.w3.org/2001/XMLSchema#integer"
        >1220</integration:beginLocation>
        <integration:content rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
         Bone Health
         Brown rice is rich in manganese, which is essential for bone health.</integration:content>
  </integration:Sentence>

  <integration:Relation rdf:about="http://www.kfupm.edu.sa/ontology/integration/relation/Relation_1355968177514">
    <integration:appearsIn>
      <integration:Sentence rdf:about="http://www.kfupm.edu.sa/ontology/integration/sentence/Sentence_1357592822725_661">
    </integration:appearsIn>
    <integration:hasPositiveEffectTo>
      <health:BodyPart rdf:about="http://www.kfupm.edu.sa/ontology/health/bodyPart/BP004"/>
    </integration:hasPositiveEffectTo>
    <integration:hasPositiveEffectFrom>
      <food:FoodItem rdf:about="http://www.kfupm.edu.sa/ontology/food/foodItem/I20036"/>
    </integration:hasPositiveEffectFrom>
    <integration:hasText rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >hasPositiveEffect</integration:hasText>
  </integration:Relation>
</rdf:RDF>
```

**Figure 7.28 Generated RDF Annotation file of Brown Rice HTML page**

# CHAPTER 8

# EVALUATION AN DISCUSSION

In this chapter, we discuss the experimental results and evaluate the performance of the MLSAF at different setting. This chapter has several objectives:

1. Describe and discuss the test environment to annotate and retrieve of information by the developed prototype.
2. Benchmark the improvement in information retrieval with and without annotation.
3. Present the test results and performance of different algorithms for entity and relationship recognition on the domains of food, nutrition, and health.
4. Finally, analyze the results of the annotation in quantitative and qualitative perspectives.

## 8.1 Preparation of the Experimentation

In order to evaluate the potential information recall on semantically annotated documents using the MLSAF prototype, we ran a set of tests based on selected Web pages related to the domain of interest, which will be described in this section. In these experiments, we used a total of more than 10k documents of full text, mostly in HTML format. These

documents were obtained from several publicly accessible Web sites in addition to structured data collected from several agencies such as the USDA[36] and ACNUT.[37].

### 8.1.1 Data Set

We have crawled 102K documents scattered across 96 trusted Web sites. Out of these documents, we have selected a set of rich documents that contain information about all three domains of food, nutrition, and health. Those documents were selected to have at least two semantic relations between concepts belonging to the three domains. We have found 9,852 documents that have at least two semantic relations, which is equivalent to 9.60% of the crawled documents. That is because most of the documents only have concepts related to the three domains but no relationships. Table 8.2 shows the statistics of the top 10 crawled Web sites, followed by a table and a chart, which show the distribution of the selected documents with at least two semantic relations

**Table 8.1 Top 10 crawled Websites**

| SN | Website | URL | # of Pages |
|---|---|---|---|
| 1 | U.S food and drug administration | www.fda.gov | 523 |
| 2 | Central of disease control and prevention | www.cdc.gov | 319 |
| 3 | Saudi medical journal | www.smj.org.sa | 1,253 |
| 4 | Service of the national library medicine | www.pubmed.gov | 729 |
| 5 | New England journal of medical | www.nejm.org | 582 |
| 6 | Medscape Continuing Medical Education | www.medscape.com | 356 |
| 7 | American Medical Association | www.ama-assn.org | 3,682 |
| 8 | American Society of Health System | www.ashp.com | 4,253 |
| 9 | US National Institutes of Health | www.nih.gov | 259 |
| 10 | Arab Center of Nutrition | www.acnut.com | 853 |

[36] http://www.usda.gov
[37] http://www.acnut.com

**Table 8.2 Distribution of selected documents based on number of relations**

| Number of relations | Number of documents |
|---|---|
| > 5 | 157 |
| 5 | 316 |
| 4 | 586 |
| 3 | 4,075 |
| 2 | 4,718 |

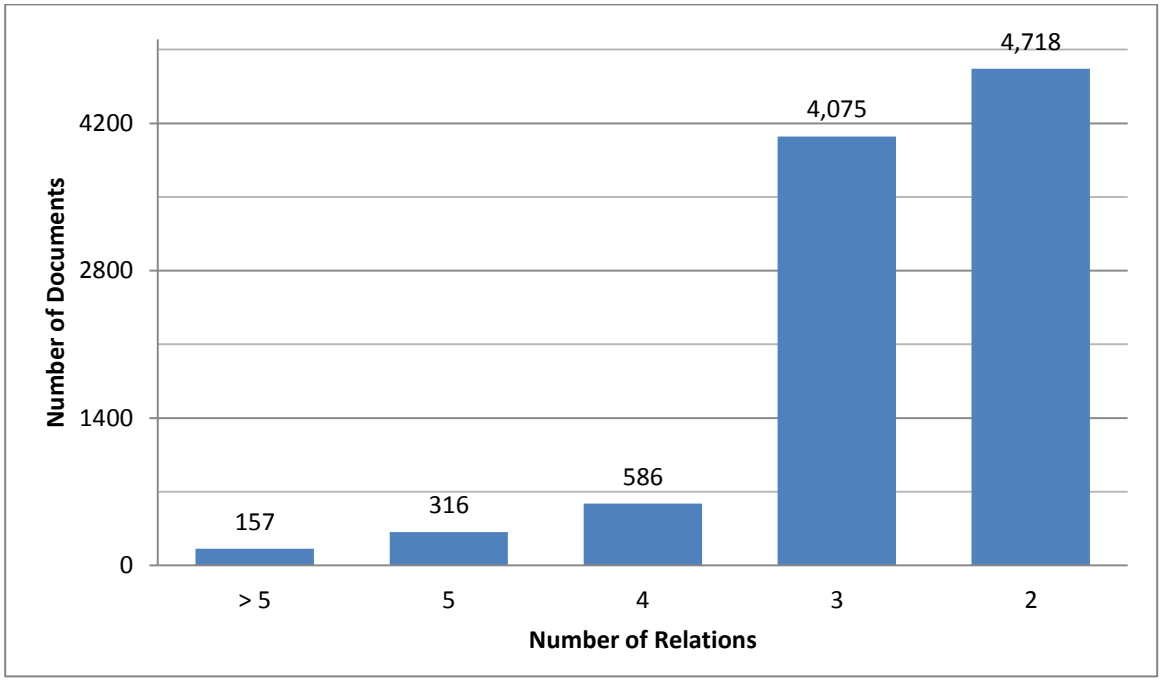



**Figure 8.1 Distribution of selected documents based on number of relations**

We have randomly selected 10% of the filtered documents for manual annotations in order to evaluate the performance of the prototype. We have specifically selected 985 documents that have 2,325 semantic relations.

**Table 8.3 Information Distribution of 985 Documents**

| Category | English | | Arabic | |
|---|---|---|---|---|
| | Count | % | Count | % |
| Documents | 546 | 55% | 439 | 45% |
| Tokens | 513786 | 0.106% | 395978 | 0.111% |
| NP | 48470 | 9% | 46044 | 12% |
| Food | 2482 | 5.12% | 1934 | 4.20% |
| Nutrient | 543 | 1.12% | 967 | 2.10% |
| Disease | 1556 | 3.21% | 1133 | 2.46% |
| Body Part | 475 | 0.98% | 396 | 0.86% |
| Body Function | 368 | 0.76% | 401 | 0.87% |

### 8.1.2 Hardware Configuration

Noting the significant amount of documents to be worked by semantic annotation system, we have used a powerful machine with Intel® Core™ i7-3820QM processor (8M Cache, up to 3.70 GHz), 12 GB memory and 1TB hard drive, and a Windows 7 operating system. In the production system, hardware configuration could be made in cluster of machine where each component could be deployed in a dedicated server.

### 8.1.3 Software Configuration

The semantic annotation system was used in batch mode with the aid of our own developed domain ontologies of food, nutrition, and health. The system is executed with input as a set of Web documents and a set of ontologies. Several customizations were made for each type of experiment in order to collect additional data that could be used to measure the performance of the annotation.

## 8.2 First Experiment: Improvement over IR

The main goal of this experiment is to investigate the gains in information retrieval from semantic annotation by comparing the precision and recall obtained before and after semantic annotation. Traditional information retrieval is based on keyword indexing mechanisms.

### 8.2.1 Query Set

We have collected 453 queries from different sources. Table 8.4 shows the source and the distributions of the collected queries.

**Table 8.4 Query Source Distribution**

| **Source** | **Number of** |
|---|---|
| Survey sent to users interested in the system | 98 |
| Domain experts | 53 |
| Yahoo! Answers | 103 |
| Google Answers | 86 |
| Various Health Consumer Websites | 113 |

We categorized the 453 queries based on the concepts related to the health and food domain concepts. Figure 8.6 shows the distribution of the queries on the categories.

**Table 8.5 Query Categories**

| Query Category | Query Type | | | Total Queries per Category |
|---|---|---|---|---|
| | Yes/No | List | Quantities | |
| Food-centric questions | 32 | 57 | 16 | 105 |
| Nutrition-centric questions | 29 | 27 | 19 | 75 |
| Disease-centric questions | 24 | 34 | 16 | 74 |
| Body Part-centric questions | 18 | 12 | 7 | 37 |
| Body Function-centric questions | 23 | 16 | 6 | 45 |
| Profile-centric questions | 11 | 8 | 7 | 26 |
| Culture-centric questions | 14 | 10 | 9 | 33 |
| Recipe-centric questions | 21 | 24 | 13 | 58 |
| Total | 172 | 188 | 93 | 453 |

### 8.2.2 Information Retrieval Environment

The technology used in information retrieval (IR) environments was developed using the Java platform based on the Lucene[38] library, which lets us index and search documents with the aim of achieving results similar to traditional search tools.

### 8.2.3 Experimental Method

A set of documents related to food, nutrition, and health domains is collected from Websites using focus crawling component customized for this purpose. The documents were converted from their source format to textual documents by the semantic annotation prototype system.

[38] http://lucene.apache.org

After the conversion step, the next step is to perform the semantic annotation of these documents using the semantic annotation system, with the help of domain ontologies related to food, nutrition, and health. In order to evaluate the semantic annotation prototype system, we set up a scenario in which it is possible to compare the efficiency of information retrieval with and without the aid of the semantic annotation. In this scenario, we used the same set of documents in the Lucene indexing tool and our semantic annotation system.

Below are the steps we followed in this experiment:

1. We randomly selected 45 queries out of the 453 identified queries in order to run the experiment.
2. For each query, we identified entities and relationships between them using annotation pipeline.
3. For each query, we identified a reference set of related documents based on the existence of the entities and their relationship, found in step 2. The mapping is done using entity matching between document set and query, then filtered the return set with the relationship found in the query.
4. We then indexed the selected 985 documents using the Lucene engine.
5. We submitted the 45 queries to Lucene through QueryParser.
6. For each query, we performed calculations for Precision, Recall, and F-measure from the results of the Lucene engine against the reference set of the query.
7. The same set of documents was then annotated by our system.
8. We manually converted the same selected queries to SPARQL queries and executed them over the annotated documents.

9. Distinct list of documents were collected for each query results from executing SPARQL queries.
10. For each query, we performed calculations for Precision, Recall, and F-measure from the results of our system against the reference set of the queries.
11. Finally, we compared the performance of the query results from Lucene and our system on the Precision, Recall and F-measure.

### 8.2.4 Experiment Result and Discussion

The results of the IR before and after annotation are shown in Table 8.6 and Table 8.7, with aggregated values for all queries.

**Table 8.6 The Performance of IR without Annotation**

| | Precision | Recall | F-measure |
|---|---|---|---|
| Minimum | 17.30% | 29.97% | 22.26% |
| Maximum | 94.92% | 100% | 97.21% |
| Average | 45.93% | 60.90% | 50.09% |
| Standard Deviation | 18.86% | 19.89% | 18.15% |

**Table 8.7 The Performance of IR with Annotation**

| | Precision | Recall | F-measure |
|---|---|---|---|
| Minimum | 75.28% | 50.21% | 60.44% |
| Maximum | 100% | 100% | 100.24% |
| Average | 90.52% | 87.01% | 88.67% |
| Standard Deviation | 7.42% | 11.47% | 7.52% |

IR-without annotation achieves 100% Recall for 5 queries, and annotation done by our system achieves 100% Recall for 13 queries. On the other hand, our system achieves 14 queries with 100% Precision, whereas Lucene did not reach 100% Precision for all queries. From the results, we conclude that semantic annotation provides relevant and precise results to user queries. The performance of our system is heavily dependent on the translation of user query to SPARQL syntax, where is this case done manually.

## 8.3 Second Experiment: Entity Recognition

The aim of this experiment is to test the recognition capability of the initial implementation of the entity recognition algorithms on the food, nutrition, and health entities. In the experiment, we collected documents from Arabic and English Web sources to test the entity recognition performance of each language.

### 8.3.1 Experiment Setup

In order to evaluate the effectiveness of the annotation subframework, a set of 200 Web documents were selected for Arabic and English and were manually annotated by two independent annotators for each language. Manual annotation was limited to concepts and relationships between food, nutrition, and health according to the domain ontologies and integration ontology in hand. Table 8.8 shows the analysis of the 200 documents with respect to the number of entities discovered by the annotators.

**Table 8.8 Entity Analysis of the Selected 200 Documents**

| Item | English Count | Arabic Count |
|---|---|---|
| Documents | 100 | 100 |

| | | |
|---|---|---|
| Tokens | 94100 | 90200 |
| Food | 420 | 410 |
| Nutrient | 100 | 199 |
| Disease | 200 | 228 |
| Body Part | 90 | 80 |
| Body Function | 65 | 79 |

Table 8.9 shows the contingency figures for manual annotation done on the English document in order to calculate the Kappa [77] agreement matrix for the two annotators.

**Table 8.9 Contingency Table for Two Annotator on English Documents**

| **English Documents** | | **Annotator 2** | | | | | |
|---|---|---|---|---|---|---|---|
| | | **Food** | **Nutrient** | **Disease** | **Body Part** | **Body Function** | **Marginal Sum** |
| **Annotator 1** | **Food** | 420 | 3 | 0 | 0 | 0 | 423 |
| | **Nutrient** | 5 | 100 | 9 | 0 | 0 | 114 |
| | **Disease** | 0 | 10 | 200 | 0 | 0 | 210 |
| | **Body Part** | 0 | 0 | 15 | 90 | 5 | 110 |
| | **Body Function** | 0 | 0 | 5 | 8 | 65 | 78 |
| | **Marginal Sum** | 425 | 113 | 229 | 98 | 70 | 935 |
| | **Agreement** | 420 | 100 | 200 | 90 | 65 | 875 |
| | **By chance** | 192 | 14 | 51 | 12 | 6 | 275 |

The kappa value for English document annotation is shown next and indicates a good agreement between the two annotators:

$$kappa = \frac{875-275}{935-275} = 90.91\%$$

**Table 8.10 Contingency Table for Two Annotator on Arabic Documents**

| **Arabic Documents** | | **Annotator 2** | | | | | |
|---|---|---|---|---|---|---|---|
| | | **Food** | **Nutrient** | **Disease** | **Body Part** | **Body Function** | **Marginal Sum** |
| **Annotator 1** | **Food** | 410 | 2 | 0 | 0 | 0 | 412 |
| | **Nutrient** | 8 | 199 | 3 | 0 | 0 | 210 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | **Disease** | 0 | 8 | 228 | 0 | 1 | 237 |
| | **Body Part** | 0 | 0 | 8 | 80 | 3 | 91 |
| | **Body Function** | 0 | 0 | 2 | 7 | 79 | 88 |
| | **Marginal Sum** | 418 | 209 | 241 | 87 | 83 | 1038 |
| | **Agreement** | 410 | 199 | 228 | 80 | 79 | 996 |
| | **By chance** | 166 | 42 | 55 | 8 | 7 | 278 |

The kappa value for Arabic document annotation is shown next and indicates a good agreement between the two annotators, better than the English annotation agreement:

$$kappa = \frac{996-278}{1038-278} = 94.47\%$$

### 8.3.2 Dictionary-Based Entity Method

The dictionary was built from ontology and knowledge-based resources with synonyms from the Arabic and English WordNet. The size of dictionary entries for each entity type including synonyms is shown in Table 8.11.

**Table 8.11 Dictionary Size per Entity Type**

| Category | English | Arabic |
|---|---|---|
| Food | 13548 | 9032 |
| Nutrient | 520 | 423 |
| Disease | 5890 | 2114 |
| Body Part | 815 | 652 |
| Body Function | 183 | 122 |

The semantic annotation of Arabic entities using the dictionary was performed on 100 Arabic documents. The dictionary was supplied with Arabic Stanford Tokenizer. The result of the annotation with respect to manually annotated documents is shown in the next two tables. Table 8.12 shows the confusion matrix of the entity annotation, and it shows the misrecognition between the different entities. The accuracy is calculated based

on the ration of correctly recognized over the total entities. Table 8.13 shows the Precision, Recall, and F-measure performance of the Arabic entity recognition. The lowest performance was found in the Body Parts entity recognition, which could be due to the number of entities in the dictionary and in the Arabic documents.

**Table 8.12 Confusion Matrix for Dictionary-Based Arabic Entities Recognition**

| | **Food** | **Nutrient** | **Disease** | **Body Part** | **Body Function** | **Others** |
|---|---|---|---|---|---|---|
| **Food** | 351 | 30 | 2 | 0 | 0 | 1 |
| **Nutrient** | 10 | 170 | 5 | 0 | 0 | 0 |
| **Disease** | 0 | 8 | 194 | 2 | 1 | 2 |
| **Body Part** | 0 | 0 | 3 | 65 | 3 | 3 |
| **Body Function** | 0 | 0 | 2 | 5 | 70 | 1 |

**Table 8.13 Dictionary-Based Arabic Entities Recognition Performance**

| **Entity** | **Precisio** | **Recall** | **F-Measure** |
|---|---|---|---|
| Food | 91.41% | 85.61% | 88.41% |
| Nutrient | 91.89% | 85.43% | 88.54% |
| Disease | 93.72% | 85.09% | 89.20% |
| Body Parts | 87.84% | 81.25% | 84.42% |
| Body | 89.74% | 88.61% | 89.17% |

Similarly, Table 8.14 and Table 8.15 summarize the results obtained by the annotation system for English entities. The lowest performance was found in the Nutrient concepts.

**Table 8.14 Confusion Matrix for Dictionary-Based English Entities Recognition**

| | **Food** | **Nutrient** | **Disease** | **Body Part** | **Body Function** | **Others** |
|---|---|---|---|---|---|---|

| | | | | | | |
|---|---|---|---|---|---|---|
| **Food** | 370 | 30 | 2 | 0 | 0 | 1 |
| **Nutrient** | 10 | 80 | 5 | 0 | 0 | 0 |
| **Disease** | 0 | 8 | 170 | 2 | 1 | 2 |
| **Body Part** | 0 | 0 | 3 | 77 | 3 | 3 |
| **Body Function** | 0 | 0 | 2 | 5 | 49 | 1 |

**Table 8.15 Dictionary-Based English Entities Recognition Performance**

| **Entity** | **Precisio** | **Recall** | **F-Measure** |
|---|---|---|---|
| Food | 91.81% | **88.10%** | **89.91%** |
| Nutrient | 84.21% | 80.00% | 82.05% |
| Disease | **92.90%** | 85.00% | 88.77% |
| Body Parts | 89.53% | 85.56% | 87.50% |
| Body Functions | 85.96% | 75.38% | 80.33% |

### 8.3.3 Rule-Based Entity Recognition

Custom rules were built for each entities-based name phrases and features of context of each word. Rules contain patterns of features such as POS tags of each word and surrounding words features. Table 8.16 shows the number of rules built for each entity category for each language. Those rules were built by hand using a trial-and-error approach with the intent to increase the recognition performance of the 100 documents for each language.

**Table 8.16 Rule Count for English and Arabic languages**

| **Category** | **English** | **Arabic** |
|---|---|---|

| | | |
|---|---|---|
| Food | 25 | 21 |
| Nutrient | 23 | 19 |
| Disease | 18 | 20 |
| Body Part | 15 | 13 |
| Body Function | 21 | 10 |

Table 8.17 shows the confusion matrix of the entity annotation and the misrecognition between the different entities. The accuracy is calculated based on the ratio of correctly recognized entities to the total number of entities..

**Table 8.17 Confusion Matrix for Rule-Based Arabic Entities Recognition**

| | **Food** | **Nutrient** | **Disease** | **Body Part** | **Body Function** | **Others** |
|---|---|---|---|---|---|---|
| **Food** | 337 | 33 | 20 | 10 | 10 | 13 |
| **Nutrient** | 20 | 177 | 2 | 0 | 0 | 14 |
| **Disease** | 0 | 13 | 212 | 2 | 1 | 18 |
| **Body Part** | 0 | 0 | 7 | 70 | 3 | 16 |
| **Body Function** | 0 | 0 | 3 | 4 | 72 | 12 |

Table 8.18 shows the Precision, Recall, and F-measure performance of the Arabic entity recognition by using rule-based recognition. The lowest performance was found in the nutrient entity recognition, which could be the result of the number and quality of rules used for Arabic documents.

**Table 8.18 Rule-Based Arabic Entities Recognition Performance**

| **Entity** | **Precision** | **Recall** | **F-Measure** |
|---|---|---|---|
| Food | 79.67% | 82.20% | 80.91% |

| | | | |
|---|---|---|---|
| Nutrient | 83.10% | 88.94% | 85.92% |
| Disease | **86.18%** | **92.98%** | **89.45%** |
| Body Parts | 72.92% | 87.50% | 79.55% |
| Body Functions | 79.12% | 91.14% | 84.71% |

Similarly, Table 8.19 and Table 8.20 summarize the results obtained by the annotation system for English entities. The lowest performance was found in the nutrient concepts.

**Table 8.19 Confusion Matrix for Rule-Based English Entities Recognition**

| | **Food** | **Nutrient** | **Disease** | **Body Part** | **Body Function** | **Others** |
|---|---|---|---|---|---|---|
| **Food** | 391 | 33 | 20 | 10 | 10 | 13 |
| **Nutrient** | 20 | 90 | 2 | 0 | 0 | 14 |
| **Disease** | 0 | 13 | 181 | 2 | 1 | 18 |
| **Body Part** | 0 | 0 | 7 | 81 | 3 | 16 |
| **Body Function** | 0 | 0 | 2 | 4 | 58 | 12 |

**Table 8.20 Rule -Based English Entities Recognition Performance**

| **Entity** | **Precisio** | **Recall** | **F-Measure** |
|---|---|---|---|
| Food | 81.97% | 93.10% | 87.18% |
| Nutrient | 71.43% | 90.00% | 79.65% |
| Disease | **84.19%** | **90.50%** | **87.23%** |
| Body Parts | 75.70% | 90.00% | 82.23% |
| Body Functions | 76.32% | 89.23% | 82.27% |

### 8.3.4 Machine Learning-Based Method

SVM models were built for each entity category: food, nutrient, disease, body part, and body function. The next set of tables show the performance of Arabic and English entity based on the SVM model.

**Table 8.21 Confusion Matrix for ML -Based Arabic Entities Recognition**

| | **Food** | **Nutrient** | **Disease** | **Body Part** | **Body Function** | **Others** |
|---|---|---|---|---|---|---|
| **Food** | 297 | 11 | 12 | 5 | 3 | 12 |
| **Nutrient** | 10 | 145 | 3 | 2 | 3 | 11 |
| **Disease** | 3 | 2 | 194 | 4 | 2 | 13 |
| **Body Part** | 3 | 4 | 2 | 65 | 1 | 3 |
| **Body Function** | 2 | 3 | 4 | 4 | 70 | 4 |

**Table 8.22 ML -Based Arabic Entities Recognition Performance**

| **Entity** | **Precision** | **Recall** | **F-Measure** |
|---|---|---|---|
| Food | 87.35% | 72.44% | 79.20% |
| Nutrient | 83.33% | 72.86% | 77.75% |
| Disease | **88.99%** | 85.09% | **87.00%** |
| Body Parts | 83.33% | 81.25% | 82.28% |
| Body Functions | 80.46% | **88.61%** | 84.34% |

Similarly, Table 8.23 and Table 8.24 summarize the results obtained by the annotation system for English entities. The lowest performance was found in the body part concepts.

**Table 8.23 Confusion Matrix for ML -Based English Entities Recognition**

| | Food | Nutrient | Disease | Body Part | Body Function | Others |
|---|---|---|---|---|---|---|
| **Food** | 350 | 11 | 12 | 5 | 3 | 15 |
| **Nutrient** | 10 | 81 | 3 | 2 | 3 | 14 |
| **Disease** | 3 | 2 | 171 | 4 | 2 | 13 |
| **Body Part** | 3 | 4 | 2 | 72 | 1 | 3 |
| **Body Function** | 2 | 3 | 4 | 4 | 49 | 4 |

**Table 8.24 ML-Based English Entities Recognition Performance**

| Entity | Precision | Recall | F-Measure |
|---|---|---|---|
| Food | 88.38% | 83.33% | 85.78% |
| Nutrient | 71.68% | 81.00% | 76.06% |
| Disease | 87.69% | 85.50% | 86.58% |
| Body Parts | 84.71% | 80.00% | 82.29% |
| Body Functions | 74.24% | 75.38% | 74.81% |

### 8.3.5 Entity Recognition Results and Discussion

The same set of selected documents is processed by our annotation engine to evaluate its performance using the Precision, Recall, and F-Measure metrics as compared to the manual annotation. Table 8.25 and Figure 8.2 summarize the results obtained by entity annotation approaches for Arabic and English respectively, allowing direct comparisons with the results for each concept category and approaches.

**Table 8.25 Arabic Entity Recognition Performance Summary**

| | Precision | | | Recall | | | F-Measure | | |
|---|---|---|---|---|---|---|---|---|---|
| | Dictionary | Rule | ML | Dictionary | Rule | ML | Dictionary | Rule | ML |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Food** | 91.4% | 79.7% | 87.4% | 85.6% | 82.2% | 72.4% | 85.6% | 82.2% | 79.2% |
| **Nutrient** | 91.9% | 83.1% | 83.3% | 85.4% | 88.9% | 72.9% | 85.4% | 88.9% | 77.7% |
| **Disease** | 93.7% | 81.4% | 89.0% | 85.1% | 78.9% | 85.1% | 85.1% | 78.9% | 87.0% |
| **Body Part** | 87.8% | 72.9% | 83.3% | 81.3% | 87.5% | 81.3% | 81.3% | 87.5% | 82.3% |
| **Body Function** | 89.7% | 79.1% | 80.5% | 88.6% | 91.1% | 88.6% | 88.6% | 91.1% | 84.3% |
| **All Entities** | 91.6% | 80.1% | 86.0% | 85.3% | 83.9% | 77.4% | 88.36% | 81.96% | 81.46% |

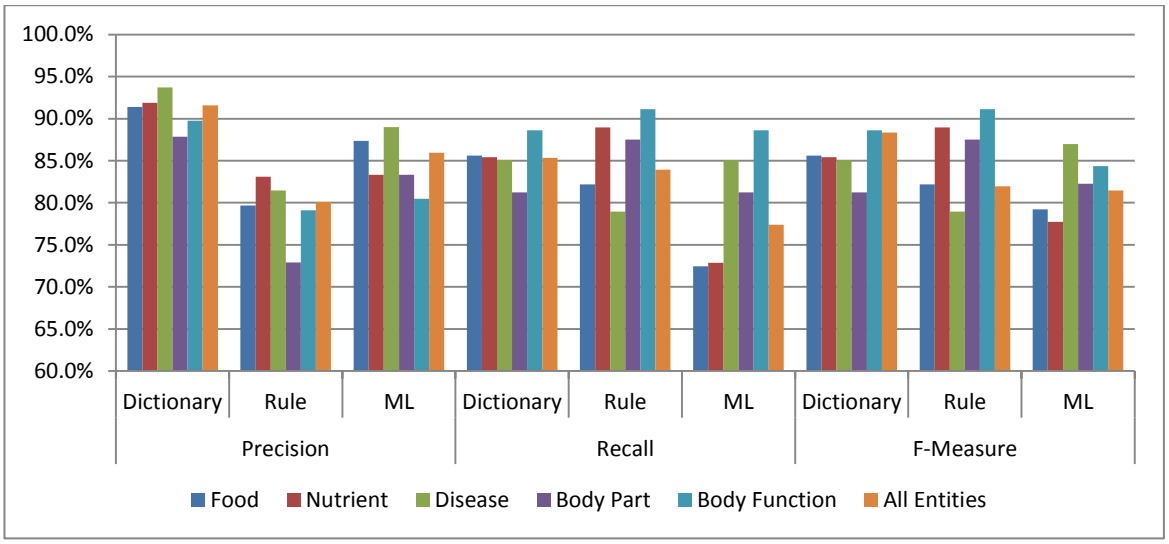



**Figure 8.2 Arabic Entity Recognition Performance Summary**

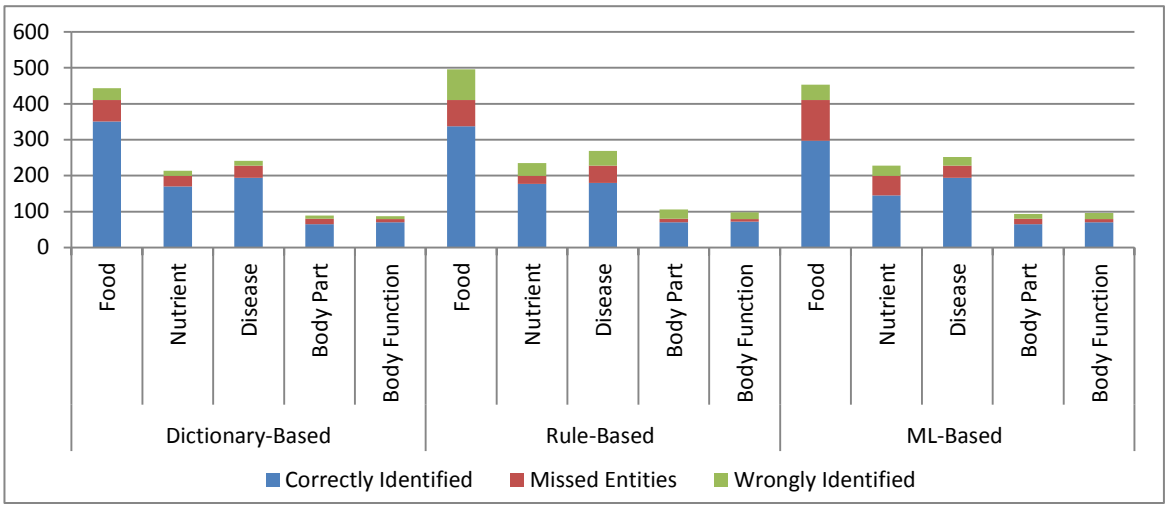



**Figure 8.3 Analysis of Arabic Entity Recognition Methods**

For the Arabic language, dictionary-based entity recognition achieves better results on average, whereas rule-based recognition achieves better results in terms of Recall for Arabic. This could be due to the number of entries in the dictionary for each category. For

rule-based recognition, the number of rules and how comprehensive those rules are with respect to the test set are the two factors that affect performance.

**Table 8.26 English Entity Recognition Performance Summary**

| | Precision | | | Recall | | | F-Measure | | |
|---|---|---|---|---|---|---|---|---|---|
| | Dictionary | Rule | ML | Dictionary | Rule | ML | Dictionary | Rule | ML |
| Food | 91.4% | 82.0% | 88.4% | 83.3% | 93.1% | 83.3% | 83.3% | 93.1% | 85.8% |
| Nutrient | 84.2% | 71.4% | 80.2% | 80.0% | 90.0% | 85.0% | 80.0% | 90.0% | 82.5% |
| Disease | 92.5% | 81.4% | 87.9% | 80.0% | 90.0% | 87.5% | 80.0% | 90.0% | 87.7% |
| Body Part | 88.6% | 75.7% | 84.7% | 77.8% | 90.0% | 80.0% | 77.8% | 90.0% | 82.3% |
| Body Function | 86.4% | 76.3% | 82.1% | 78.5% | 89.2% | 84.6% | 78.5% | 89.2% | 83.3% |
| All Entities | 90.1% | 79.4% | 86.4% | 81.3% | 91.4% | 84.2% | 85.46% | 85.02% | 85.30% |

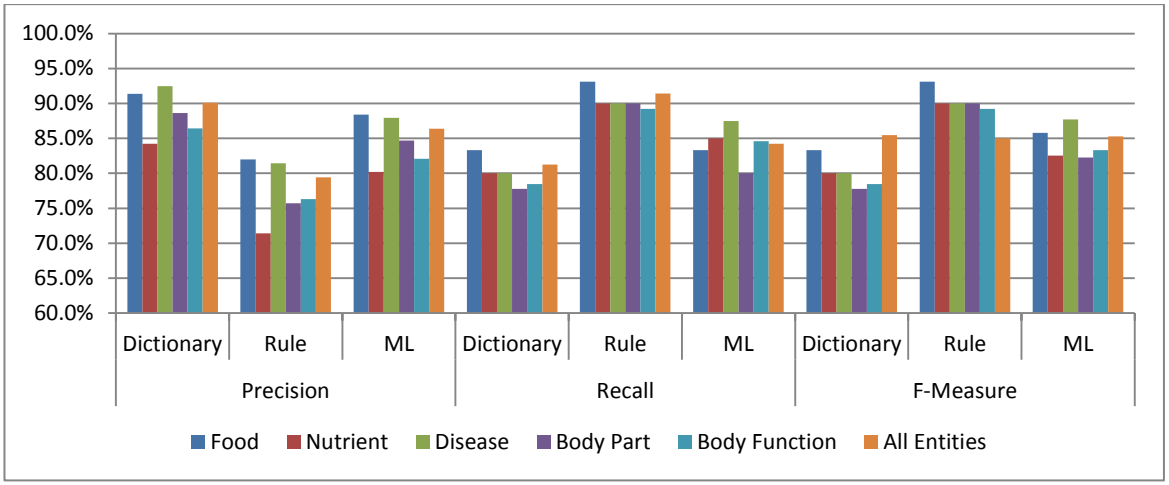



**Figure 8.4 English Entity Recognition Performance Summary**

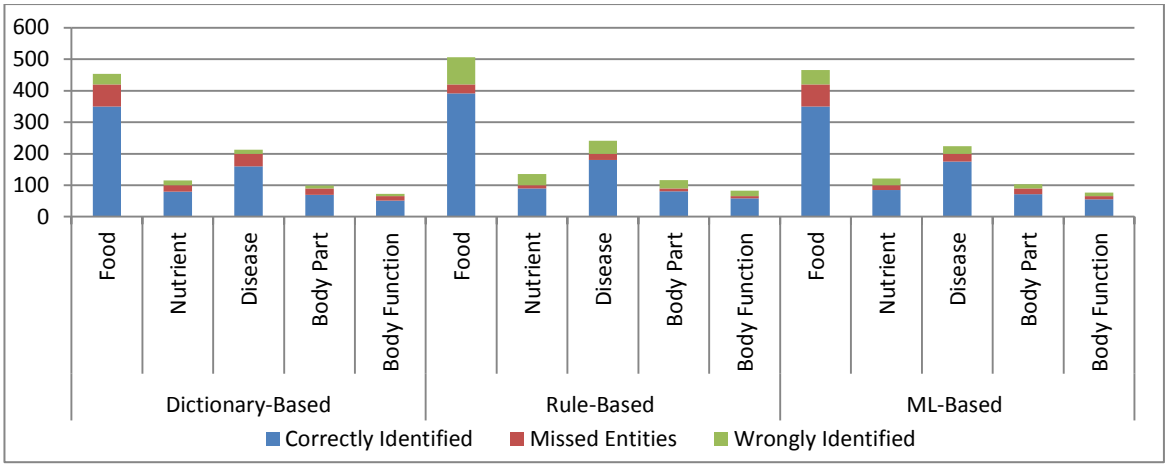



**Figure 8.5 Analysis of English Entity Recognition Methods**

For the English language, dictionary-based entity recognition achieves better results on average than the other method for all three measures. This could a result of the number English entries in the dictionary for each category. For both Arabic and English, ML did not do well; this could be because the number of training samples supplied to SVM was not enough to generalize or for the feature set used.

## 8.4 Third Experiment: Relation Recognition

The aim of the third experiment is to test the recognition capability of the initial implementation of the relationship recognition algorithms on the food, nutrition, and health entities based on domain ontologies. The experiment included documents collected from Arabic and English sources for the purpose of testing the relationship recognition performance of each language.

### 8.4.1 Experiment Setup

In order to evaluate the effectiveness of the annotation subframework, we selected a set of 200 Web documents for Arabic and English, and these were manually annotated by two

independent annotators for each language. Manual annotation was limited to the relationships between food, nutrition, and health according to the domain ontologies and integration ontology in hand. Table 8.27 shows the analysis of the 200 documents with respect to the number of entities discovered by the annotators.

Table 8.28 shows the contingency figures for manual annotation done on the Arabic documents in order to calculate the Kappa [77] agreement matrix for the two annotators

**Table 8.27 Entity and Relation Analysis of the Selected 200 Documents**

| Item type | Item | English | Arabic |
|---|---|---|---|
| General | Documents | 100 | 100 |
| | Tokens | 94100 | 90200 |
| | Sentences | 6273 | 6013 |
| | NP | 9225 | 10488 |
| Entities | Food | 420 | 410 |
| | Nutrient | 100 | 199 |
| | Disease | 200 | 228 |
| | Body Part | 90 | 80 |
| | Body Function | 65 | 79 |
| Relations | Prevent | 94 | 90 |
| | Treat | 75 | 72 |
| | Cause | 69 | 66 |
| | Good For | 88 | 84 |
| | Bad For | 82 | 78 |

**Table 8.28 Contingency Table for Two Annotator on Arabic Documents**

| Arabic Documents | | Annotator 2 | | | | | |
|---|---|---|---|---|---|---|---|
| | | **Prevent** | **Treat** | **Cause** | **Good For** | **Bad For** | **Marginal sum** |
| **Annotator 1** | **Prevent** | 90 | 4 | 0 | 5 | 0 | 99 |
| | **Treat** | 5 | 72 | 0 | 6 | 0 | 83 |
| | **Cause** | 0 | 0 | 66 | 0 | 10 | 76 |
| | **Good For** | 6 | 7 | 0 | 84 | 0 | 97 |
| | **Bad For** | 0 | 0 | 11 | 0 | 78 | 89 |
| | **Marginal sum** | 101 | 83 | 77 | 95 | 88 | 444 |
| | **Agreement** | 90 | 72 | 66 | 84 | 78 | 390 |
| | **By chance** | 45 | 10 | 19 | 10 | 7 | 90 |

The kappa value for Arabic document annotation is shown next which indicates a good agreement between the two annotators:

$$kappa = \frac{390-90}{444-90} = 84.73\%$$

**Table 8.29 Contingency Table for Two Annotators on English Documents**

| English Documents | | Annotator 2 | | | | | |
|---|---|---|---|---|---|---|---|
| | | **Prevent** | **Treat** | **Cause** | **Good For** | **Bad For** | **Marginal sum** |
| **Annotator 1** | **Prevent** | 94 | 5 | 0 | 7 | 0 | 106 |
| | **Treat** | 8 | 75 | 0 | 8 | 0 | 91 |
| | **Cause** | 0 | 0 | 69 | 0 | 8 | 77 |
| | **Good For** | 9 | 6 | 0 | 88 | 0 | 103 |
| | **Bad For** | 0 | 0 | 7 | 0 | 82 | 89 |
| | **Marginal Sum** | 111 | 86 | 76 | 103 | 90 | 466 |
| | **Agreement** | 94 | 75 | 69 | 88 | 82 | 408 |
| | **By chance** | 48 | 11 | 19 | 11 | 7 | 95 |

The kappa value for English documents annotation is shown next which indicates a good agreement between the two annotators is almost similar to Arabic annotation agreement:

$$kappa = \frac{408-95}{466-95} = 84.35\%$$

### 8.4.2 Trigger Word Method

Trigger words were automatically collected from integration ontology for all object properties. Table 8.30 shows a sample of the trigger words for different relationships in both Arabic and English.

**Table 8.30 Sample Trigger Words for Different Relations**

| Relation | English trigger words | Arabic trigger words |
|---|---|---|
| Prevent | prevent, prevents, preventing, prevention, protect, protects, protecting, protection | يحمي، يقي |
| Treat | Treat, Treats, Treating, Treatment, cure, cures, curing, , remedy, remedial | يعالج، معالجة، |
| Cause | Cause, causing, lead | يسبب، تسبب ، تؤدي ، يؤدي |
| Good For | Suitable, recommended, suggested , acceptable, advice, promote | مناسب ، جيد ، ممكن ، ينصح ، مقبول، مصرح |
| Bad For | not suitable, not recommended, | غير مناسب ، ممنوع ، غير مصرح |

Table 8.31 shows the confusion matrix of the relation annotation and the misrecognition between them. The trigger words method has, on average, more than 17% wrongly classified from outside the five relationships.

**Table 8.31 Confusion Matrix for Arabic Relation Recognition Using Trigger Words**

| | Prevent | Treat | Cause | Good For | Bad For | Others |
|---|---|---|---|---|---|---|
| **Prevent** | 80 | 5 | 2 | 3 | 0 | 20 |
| **Treat** | 2 | 60 | 5 | 2 | 1 | 15 |
| **Cause** | 0 | 2 | 57 | 3 | 4 | 13 |
| **Good For** | 3 | 1 | 4 | 70 | 2 | 11 |
| **Bad For** | 1 | 2 | 1 | 3 | 67 | 20 |

Table 8.32 summarizes the recognition accuracy, in terms of Precision, Recall, and F-measure, achieved by the proposed tool for facts in terms of the Arabic documents' relationships between food, nutrition, and health concepts by using trigger words method.

**Table 8.32 Arabic Relation Extraction Performance Using Trigger Words Method**

| Relation | Precision | Recall | F-Measure |
|---|---|---|---|
| Prevent | 72.73% | 88.89% | 80.00% |
| Treat | 70.59% | 83.33% | 76.43% |
| Cause | 72.15% | 86.36% | 78.62% |
| Good For | 76.92% | 83.33% | 80.00% |
| Bad For | 71.28% | 85.90% | 77.91% |

Table 8.33 shows the confusion matrix of the English documents' relationship annotation and the misrecognition between them. The trigger words method has in average more than 16% wrongly classified from outside the five relationships, which is a little better than the Arabic case.

**Table 8.33 Confusion Matrix for English Relation Recognition Using Trigger Words**

| | Prevent | Treat | Cause | Good For | Bad For | Others |
|---|---|---|---|---|---|---|
| **Prevent** | 84 | 2 | 3 | 3 | 1 | 19 |
| **Treat** | 3 | 65 | 3 | 2 | 2 | 13 |
| **Cause** | 2 | 0 | 60 | 1 | 2 | 15 |
| **Good For** | 2 | 3 | 4 | 80 | 3 | 14 |
| **Bad For** | 1 | 1 | 2 | 1 | 77 | 18 |

Table 8.34 summarizes the recognition accuracy, in terms of Precision, Recall, and F-measure, achieved by the proposed tool for facts in terms of the English documents' relationships between food, nutrition, and health concepts by using the trigger words method.

**Table 8.34 English Relation Extraction Performance Using Trigger Words Method**

| Relation | Precision | Recall | F-Measure |
|---|---|---|---|
| Prevent | 75.00% | 89.36% | 81.55% |
| Treat | 73.86% | 86.67% | 79.75% |
| Cause | 75.00% | 86.96% | 80.54% |
| Good For | 75.47% | 90.91% | 82.47% |
| Bad For | 77.00% | 93.90% | 84.62% |

With this empirical evaluation for both Arabic and English documents, the results show promising Recall and Precision. Our long-term evaluation plan involves a larger evaluation corpus to check the accuracy and comprehensiveness of the prototype system. We are also planning to improve the detection rules and to increase the coverage of the detection rules to cover more named entity combinations.

### 8.4.3 Pattern-Based Method

This approach uses a manually constructed set of lexical patterns for each semantic relationship. The constructed patterns are regular expressions describing a set of matching sentences containing entities at specified positions with more or less specific lexical context. More precisely, each pattern consists of a sequence of words, tags corresponding

to the three concept types, and generic markers representing a length-limited character sequence. The patterns were constructed from the training corpus, and external electronic dictionaries were used to enrich them with synonyms of important words. Table 8.35 shows number of constructed patterns and some simplified examples.

**Table 8.35 Sample Constructed Patterns for Relation Extraction**

| Relation | Example Sentence(English) | Example Pattern (English) | English Patterns | Arabic Patterns |
|---|---|---|---|---|
| Prevent | eating apples lower risk of developing lung cancer | eating <FOOD> lower risk of developing <DISEASE> | 25 | 23 |
| Treat | eating two apples per day may lower cholesterol | eating <tokens>? <FOOD> <tokens>? may lower <DISEASE> | 30 | 27 |
| Cause | Alcohol can cause hypoglycemia shortly after drinking | <NUTRIENT> can cause <DISEASE> shortly after drinking | 28 | 30 |
| Good For | carrots and other orange-colored fruits and vegetables promote eye health and protect vision | <FOOD> <TOKEN>? promote <BODY-PART> health | 15 | 21 |
| Bad For | eating a lot of junk foods like potato chips can really damage your eye health and vision | eating <TOKEN>? <FOOD> <TOKEN>? damage <TOKEN>? <BODY-PART> and <BODY-FUNCTION> | 17 | 15 |

Table 8.36 shows the confusion matrix of the relation annotation and the misrecognition between them. The patterns-based method has, on average, more than 13% wrongly classified from outside the five relationships.

**Table 8.36 Confusion Matrix for Arabic Patterns-Based Relation Recognition**

| | Prevent | Treat | Cause | Good For | Bad For | Others |
|---|---|---|---|---|---|---|
| **Prevent** | 79 | 3 | 1 | 2 | 1 | 13 |
| **Treat** | 2 | 59 | 2 | 2 | 1 | 11 |

| | | | | | | |
|---|---|---|---|---|---|---|
| **Cause** | 1 | 2 | 56 | 2 | 3 | 9 |
| **Good For** | 1 | 2 | 2 | 71 | 3 | 10 |
| **Bad For** | 1 | 2 | 1 | 3 | 65 | 12 |

Table 8.37 summarizes the recognition accuracy, in terms of Precision, Recall, and F-measure, achieved by the proposed tool for facts in terms of the Arabic relationships between food, nutrition, and health concepts by using the patterns-based method.

**Table 8.37 Arabic Relation Extraction Performance Using Patterns-Based Method**

| Relation | Precision | Recall | F-Measure |
|---|---|---|---|
| Prevent | 79.80% | 87.78% | 83.60% |
| Treat | 76.62% | 81.94% | 79.19% |
| Cause | 76.71% | 84.85% | 80.58% |
| Good For | 79.78% | 84.52% | 82.08% |
| Bad For | 77.38% | 83.33% | 80.25% |

Table 8.38 shows the confusion matrix of the relationship annotation and the misrecognition between them. The patterns-based method for English had, on average, more than 9.5% wrongly classified from outside the five relationships.

**Table 8.38 Confusion Matrix for English Patterns-Based Relation Recognition**

| | Prevent | Treat | Cause | Good For | Bad For | Others |
|---|---|---|---|---|---|---|
| **Prevent** | 80 | 3 | 2 | 2 | 1 | 9 |
| **Treat** | 2 | 66 | 3 | 2 | 2 | 7 |
| **Cause** | 2 | 0 | 63 | 1 | 2 | 8 |

| | | | | | | |
|---|---|---|---|---|---|---|
| **Good For** | 1 | 3 | 3 | 77 | 3 | 7 |
| **Bad For** | 2 | 1 | 2 | 1 | 75 | 11 |

Table 8.39 summarizes the recognition accuracy, in terms of Precision, Recall, and F-measure, achieved by the proposed tool for facts in terms of the English documents' relationships between food, nutrition, and health concepts by using the patterns-based method.

**Table 8.39 English Relation Extraction Performance Using Patterns-Based Method**

| Relation | Precision | Recall | F-Measure |
|---|---|---|---|
| Prevent | 82.47% | 85.11% | 83.77% |
| Treat | 80.49% | 88.00% | 84.08% |
| Cause | 82.89% | 91.30% | 86.90% |
| Good For | 81.91% | 87.50% | 84.62% |
| Bad For | 81.52% | 91.46% | 86.21% |

In summary, for both languages, the pattern-based method shows high recall with moderate precision performance. On average, the English version of the relationship recognizer shows a better performance, especially for *cause* and *bad for* relationships.

### 8.4.4 Entity-Predicate Pair Detection Method

For Arabic language documents, we used Stanford and our own Noun Phrase tagger, which was developed during this thesis work. Table 8.40 shows the confusion matrix of the relation annotation and the misrecognition between the different relations. The EPP

method for Arabic had, on average, more than 3% wrongly classified from outside the five relationships.

**Table 8.40 Confusion Matrix for Arabic EPP Relation Recognition**

| | **Prevent** | **Treat** | **Cause** | **Good For** | **Bad For** | **Others** |
|---|---|---|---|---|---|---|
| **Prevent** | 61 | 0 | 0 | 0 | 0 | 1 |
| **Treat** | 0 | 45 | 0 | 0 | 0 | 2 |
| **Cause** | 1 | 1 | 42 | 2 | 0 | 1 |
| **Good For** | 0 | 0 | 0 | 53 | 0 | 3 |
| **Bad For** | 1 | 0 | 2 | 1 | 48 | 1 |

Table 8.41 summarizes the recognition accuracy, in terms of Precision, Recall, and F-measure, achieved by the proposed tool for facts in terms of the Arabic documents' relations between food, nutrition, and health concepts by using the EPP method.

**Table 8.41 Arabic Relation Extraction Performance Using EPP Method**

| Relation | Precision | Recall | F-Measure |
|---|---|---|---|
| Prevent | 98.39% | 67.78% | 80.26% |
| Treat | 95.74% | 62.50% | 75.63% |
| Cause | 89.36% | 63.64% | 74.34% |
| Good For | 94.64% | 63.10% | 75.71% |
| Bad For | 90.57% | 61.54% | 73.28% |

For this relationship recognition method in English, we used three common dependency parsers for English: MiniPar, Stanford Parser, and SUPPLE. Starting from parse sentences, we used a set of rules to capture subject, action, and object from the parse tree. Samples from the rules used are shown in Table 8.42.

**Table 8.42 Sample of English Rule Set for EPP**

| Component | Rule Set |
|---|---|
| Noun Phrase<br>(NP) | NP(Noun Phrase): N(Noun),<br>Pronoun,<br>[any number of ADJP(Adjective Phrase)] + N,<br>NP + [any number of ADJP],<br>NP + CONJ + NP |
| Adjective Phrase<br>(ADJP) | ADJ(Adjective),<br>[any number of ADVP(Adverb Phrase)] + ADJP,<br>PREP(Preposition) + NP |
| Adverb Phrase<br>(ADVP) | ADV(Adverb),<br>ADV + ADVP |
| Verb Phrase<br>(VP) | Vi(Intransitive Verb),<br>Vt(Transitive Verb) + NP,<br>VP + [any number of ADVP],<br>VP + CONJ + VP,<br>[any number of ADVP] + VP |
| Sentence<br>(S) | NP(Noun Phrase) + VP(Verb Phrase),<br>NP + AUX_V(Auxiliary Verb) + VP,<br>VP(Verb Phrase) (<=imperative sentence),<br>S + CONJ + S |

Table 8.43 shows the confusion matrix of the relationship annotation, and it shows the misrecognition between the different relationships. The EPP method for English had, on average, slightly more than 3% wrongly classified from outside the five relationships.

**Table 8.43 Confusion Matrix for English EPP Relation Recognition**

| | **Prevent** | **Treat** | **Cause** | **Good For** | **Bad For** | **Others** |
|---|---|---|---|---|---|---|
| **Prevent** | 62 | 1 | 1 | 0 | 1 | 2 |
| **Treat** | 1 | 59 | 1 | 1 | 0 | 2 |
| **Cause** | 1 | 0 | 55 | 1 | 2 | 3 |
| **Good For** | 1 | 2 | 1 | 65 | 0 | 2 |
| **Bad For** | 0 | 1 | 1 | 0 | 61 | 1 |

Table 8.44 summarizes the recognition accuracy, in terms of Precision, Recall, and F-measure, achieved by the proposed tool for facts in terms of the English documents' relationships between food, nutrition, and health concepts by using the EPP method.

**Table 8.44 English Relation Extraction Performance Using EPP**

| **Relation** | **Precision** | **Recall** | **F-Measure** |
|---|---|---|---|
| Prevent | 92.54% | 65.96% | 77.02% |
| Treat | 92.19% | 78.67% | 84.89% |
| Cause | 88.71% | 79.71% | 83.97% |
| Good For | 91.55% | 73.86% | 81.76% |
| Bad For | 95.31% | 74.39% | 83.56% |

In summary, for both Arabic and English, the EPP method shows high precision with low recall performance. On average, the English version of the relationship recognizer shows

better performance due to the use of three types of parsers, compared to the Arabic version, wherein only one is used.

### 8.4.5 ML Method

SVM models were built for each relation type: prevent, treat, cause, good for, bad for. The next two tables show the performance of relationships based on the SVM model. The performance of prevent relation recognition was shown as the highest, which could be due to the number of cases presented during the training of the SVM model.

Table 8.45 shows the confusion matrix of the relationship annotation and the misrecognition between them. The ML method for Arabic has, on average, more than 5% wrongly classified from outside the five relationships.

**Table 8.45 Confusion Matrix for Arabic Relation Recognition Using SVM**

| | **Prevent** | **Treat** | **Cause** | **Good For** | **Bad For** | **Others** |
|---|---|---|---|---|---|---|
| **Prevent** | 79 | 5 | 2 | 3 | 0 | 5 |
| **Treat** | 2 | 58 | 5 | 2 | 1 | 3 |
| **Cause** | 1 | 1 | 58 | 3 | 2 | 6 |
| **Good For** | 3 | 1 | 4 | 67 | 2 | 2 |
| **Bad For** | 1 | 2 | 1 | 3 | 67 | 4 |

Table 8.46 summarizes the recognition accuracy, in terms of Precision, Recall, and F-measure, achieved by the proposed tool for facts in terms of the Arabic documents' relationships between food, nutrition, and health concepts by using the SVM method.

**Table 8.46 Arabic Relation Extraction Performance Using SVM**

| Relation | Precision | Recall | F-Measure |
|---|---|---|---|
| Prevent | 84.04% | 87.78% | 85.87% |
| Treat | 81.69% | 80.56% | 81.12% |
| Cause | 81.69% | 87.88% | 84.67% |
| Good For | 84.81% | 79.76% | 82.21% |
| Bad For | 85.90% | 85.90% | 85.90% |

Table 8.47 shows the confusion matrix of the relationship annotation and the misrecognition between them. The SVM method for English has, on average, more than 4.6% wrongly classified from outside the five relationships.

**Table 8.47 Confusion Matrix for English Relation Recognition Using SVM**

| | Prevent | Treat | Cause | Good For | Bad For | Others |
|---|---|---|---|---|---|---|
| **Prevent** | 77 | 5 | 2 | 3 | 0 | 4 |
| **Treat** | 2 | 64 | 5 | 2 | 1 | 3 |
| **Cause** | 1 | 3 | 60 | 3 | 2 | 3 |
| **Good For** | 3 | 1 | 4 | 75 | 2 | 5 |
| **Bad For** | 1 | 2 | 2 | 2 | 72 | 4 |

Table 8.48 summarizes the recognition accuracy, in terms of Precision, Recall, and F-measure, achieved by the proposed tool for facts in terms of the English documents' relationships between food, nutrition, and health concepts by using SVM method.

**Table 8.48 English Relation Extraction Performance Using SVM**

| Relation | Precision | Recall | F-Measure |
|---|---|---|---|

| | | | |
|---|---|---|---|
| Prevent | 84.62% | 81.91% | 83.24% |
| Treat | 83.12% | 85.33% | 84.21% |
| Cause | 83.33% | 86.96% | 85.11% |
| Good For | 83.33% | 85.23% | 84.27% |
| Bad For | 86.75% | 87.80% | 87.27% |

In summary, for both English and Arabic documents, the SVM method shows moderate precision and recall performance. On average, SVM shows competitive performance for both languages.

### 8.4.6 Relation Recognition Results and Discussion

Several semantic relation extraction approaches only address relationship detection. In the context of semantics, we are interested not only in relationship detection but also in the linked domain entities. We focus on searching <subject,relation,object> triples such that the subject and the object have known categories (semantic types) and the relation is valid w.r.t domain knowledge and linguistic considerations (i.e., the sentence really states that the source treats the target). In this context, the same sentence may contain several <subject,relation,object> triples.

**Table 8.49 Arabic Relation Recognition Methods Performance Summary**

| Arabic Relation | Precision | | | | Recall | | | | F-Measure | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Trigger | Rule | MPP | ML | Trigger | Rule | MPP | ML | Trigger | Rule | MPP | ML |
| Prevent | 72.7% | 79.8% | 98.39% | 84.0% | 88.9% | 87.8% | 67.78% | 87.8% | 88.9% | 87.8% | 80.26% | 85.9% |
| Treat | 70.6% | 76.6% | 95.74% | 81.7% | 83.3% | 81.9% | 62.50% | 80.6% | 83.3% | 81.9% | 75.63% | 81.1% |
| Cause | 72.2% | 76.7% | 89.36% | 81.7% | 86.4% | 84.8% | 63.64% | 87.9% | 86.4% | 84.8% | 74.34% | 84.7% |
| Good For | 76.9% | 79.8% | 94.64% | 84.8% | 83.3% | 84.5% | 63.10% | 79.8% | 83.3% | 84.5% | 75.71% | 82.2% |
| Bad For | 71.3% | 77.4% | 90.57% | 85.9% | 85.9% | 83.3% | 61.54% | 85.9% | 85.9% | 83.3% | 73.28% | 85.9% |

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| All Relations | 72.8% | 78.2% | 93.96% | 83.7% | 85.6% | 84.6% | 63.85% | 84.4% | 78.7% | 81.3% | 76.03% | 84.0% |

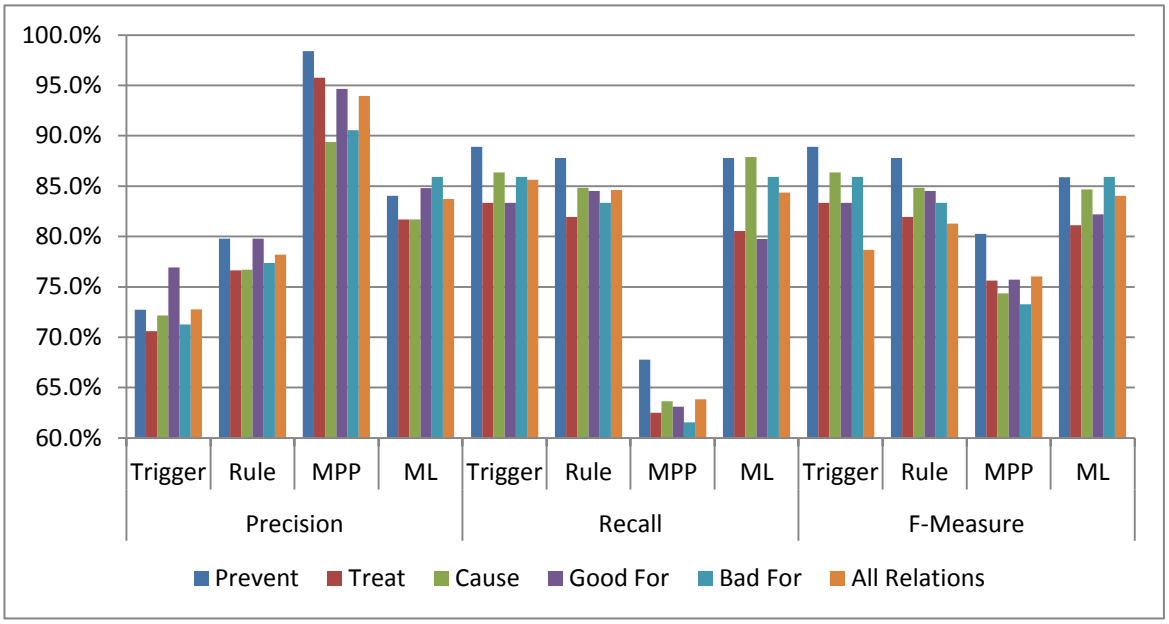



**Figure 8.6 Arabic Relation Recognition Methods Performance Summary**

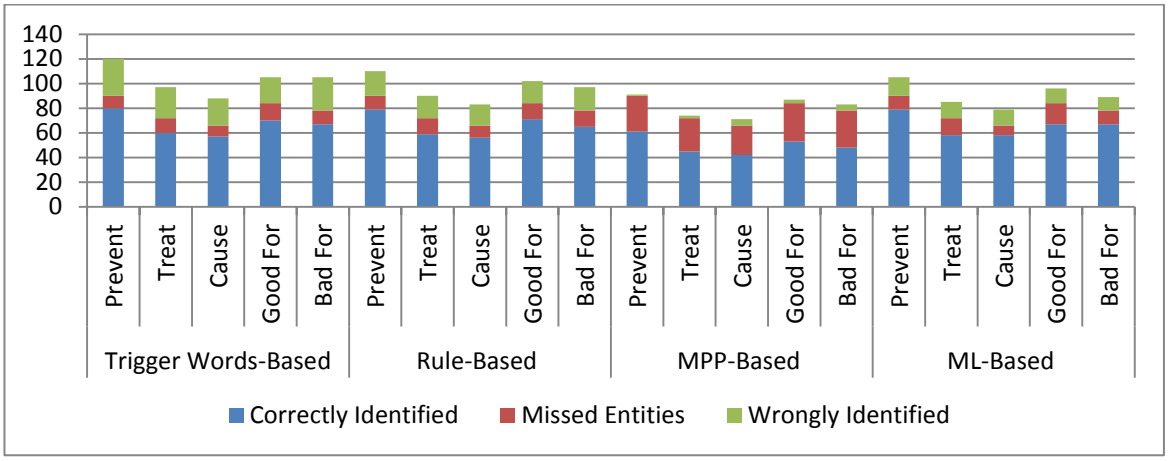



**Figure 8.7 Analysis of Arabic Relation Recognition Methods**

**Table 8.50 English Relation Recognition Methods Performance Summary**

| English Relation | Precision | | | | Recall | | | | F-Measure | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Trigger | Rule | MPP | ML | Trigger | Rule | MPP | ML | Trigger | Rule | MPP | ML |
| Prevent | 75.0% | 82.5% | 92.54% | 84.6% | 89.4% | 85.1% | 65.96% | 81.9% | 89.4% | 85.1% | 77.02% | 83.2% |
| Treat | 73.9% | 80.5% | 92.19% | 83.1% | 86.7% | 88.0% | 78.67% | 85.3% | 86.7% | 88.0% | 84.89% | 84.2% |
| Cause | 75.0% | 82.9% | 88.71% | 83.3% | 87.0% | 91.3% | 79.71% | 87.0% | 87.0% | 91.3% | 83.97% | 85.1% |
| Good For | 75.5% | 81.9% | 91.55% | 83.3% | 90.9% | 87.5% | 73.86% | 85.2% | 90.9% | 87.5% | 81.76% | 84.3% |
| Bad For | 77.0% | 81.5% | 95.31% | 86.7% | 93.9% | 91.5% | 74.39% | 87.8% | 93.9% | 91.5% | 83.56% | 87.3% |

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| All Relations | 75.3% | 81.9% | 92.07% | 84.3% | 89.7% | 88.5% | 74.02% | 85.3% | 81.9% | 85.0% | 82.07% | 84.8% |

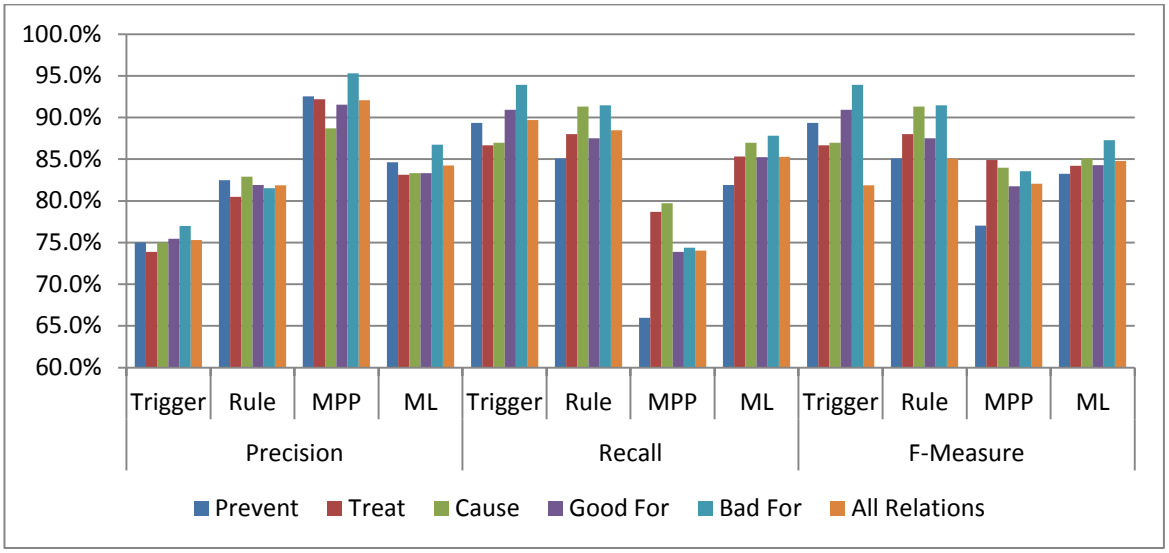



**Figure 8.8 English Relation Recognition Methods Performance Summary**

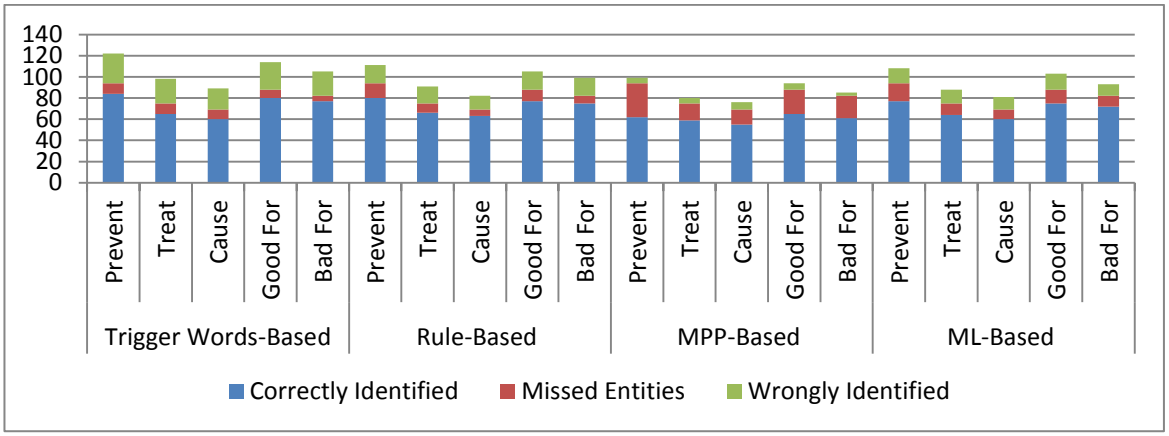



**Figure 8.9 Analysis of English Relation Recognition Methods**

For rule-based relationship recognition, the first analysis of the false positives shows that the main causes of errors are:

1. Errors in the extraction of domain entities
2. Rules built for each relationship that also cover forms of expression of other relationships

3. Sentences that contain possible subject and object entities without being connected with explicit relationships

A classic disadvantage of the rule-based method is the high cost required to obtain a good recall. Nevertheless, it is interesting to test and improve manual rules to maintain good control on the precision of the extraction. Also, such methods can be integrated in hybrid extraction approaches to balance their qualities with those of ML methods.

For ML-based methods, the first analysis of the false positives shows that the main causes of errors are:

1. Errors in the extraction of domain entities
2. Limited number of training cases provided during training for each relationship type
3. Number and type of feature sets provided
4. Configuration of SVM in terms of type of Kernel used and other settings

Classic disadvantages of the ML-based method include the high cost needed to obtain comprehensive and enough training samples that cover different aspects of feature space.

.

## 8.5 Speed Evaluation

One important characteristic of entity and relationship recognition solutions is the annotation speed, because large data sets may be annotated to collect as much information as possible. To evaluate the annotation speed achievable with the MLSAF prototype, we performed various experiments using 200 full-text articles with 4,231 sentences. The

documents were processed on a machine with 8 processing cores @ 2.67 GHz and 16GB of RAM.

The entity annotation process, using the dictionaries, the rule-based and ML models previously described, and 5 threads, took 315 seconds, which corresponds to processing 17.81 sentences per second. Thus, it took 1.575 seconds on average to process a full-text article. Because generating the complex features for the ML model, parsing sentences, and collecting POS and chunking features are resource-intensive, we also measured the processing speed without using parsing and ML by applying only dictionary matching, rule-based recognition, and tokenization from the NLP module. With this configuration, the document set was processed in 520 seconds, which corresponds to 10 sentences per second. Thus, a full-text article was processed in an average of 2.6 seconds.

## 8.6 Chapter Summary

In this chapter, we presented several approaches for the recognition of domain entities and the semantic relationships linking them. These approaches are performed in two main steps: (1) the recognition of domain entities and (2) the exploitation of several methods, taking into account the semantic types of domain entities. The results obtained on the test corpus show the effectiveness of our approach and its advantages for semantic Web applications. In the short term, we intend to study the false negatives in order to improve the performance of the prototype.

# CHAPTER 9

# CONCLUSION AND FUTURE WORK

This thesis describes the details of a multilingual semantic annotation framework (MLSAF). The MLSAF framework is designed as a modular structure consisting of six layers: acquisition layer, preprocessing layer, based and advance NLP layers, recognition layer, and postprocessing layer. The framework provides a complete solution for semantic annotation of multilingual Web sources based on multilingual domain ontologies. This thesis presents an approach to building an ontology using multilingual online structured and semistructured data sources such as Wikipedia and WordNet. The thesis also presents the processes for developing domain ontologies and how to use them to semantically annotate Web sources related to food, nutrition, and health domains. The thesis also presents the implementation details of the proposed framework. Several experiments with different objectives were conducted to test the benefits gained from the semantic annotation for the information retrieval task, and the results show better improvement in precision and recall. In the future, we will pursue the following activities:

1. Creating a flexible GUI to work as a test bed to try different annotation algorithms for different domains, with different configuration parameters for each algorithm.

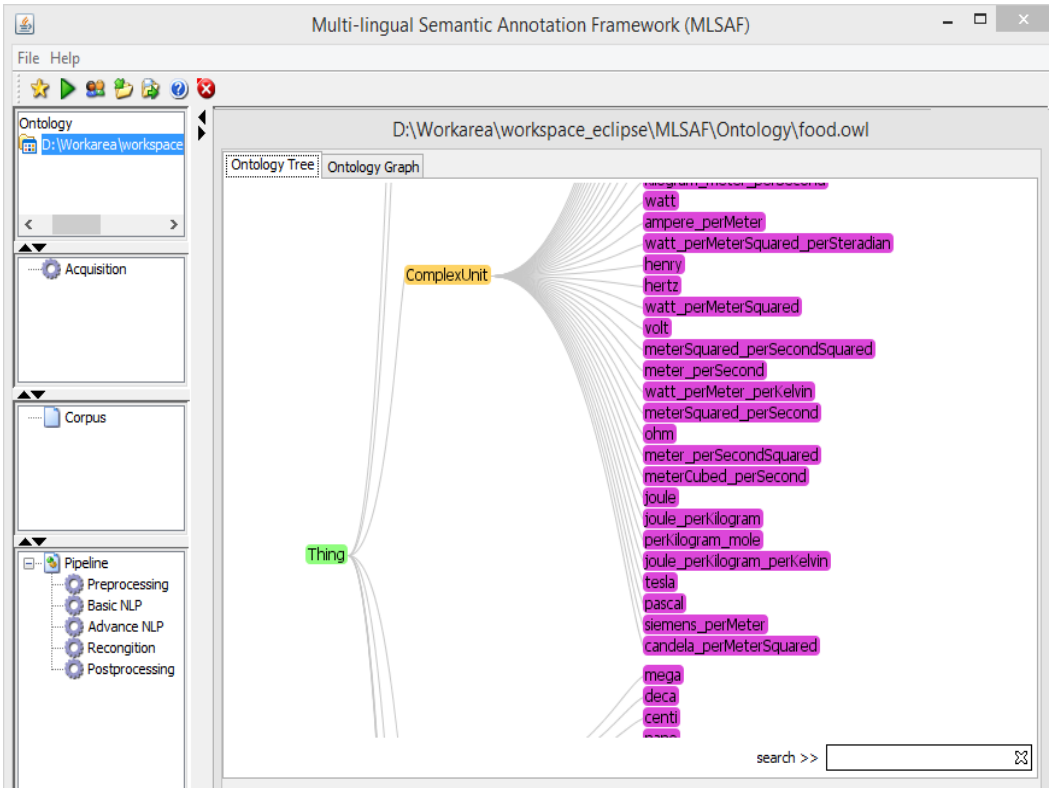



**Figure 9.1 Framework Flexible Interface**

2. Adding different implementations for ML algorithms in addition to SVM, including Perceptron, Bayesian Network, Decision Tree (C4.5), CRF, and HMM.
3. Providing a Web interface version and Web service API that enable developers to develop applications based on this framework.
4. Extending the framework prototype to cover different domains, such as education and travel.

# References


[1] T. Berners-Lee., "Weaving the web: the original design and the ultimate destiny of the World Wide Web by its inventor," *HarperCollins Publ. New York*, 1999.

[2] T. Berners-Lee, J. Hendler, and O. Lassila., "The Semantic Web," *Sci. Am.*, vol. 284, no. 5, pp. 34–43, 2001.

[3] "Pew Research Center's Internet & American Life Project," *Online Health Search*, 2006. [Online]. Available: http://www.pewinternet.org/.

[4] E. Hyvönen, K. Viljanen, and O. Suominen, "HealthFinland—Finnish Health Information on the Semantic Web," *Proc. 6th Int. Semant. Web Conf. 2nd Asian Semant. Web Conf. (ISWC/ASWC 2007)*, 2007.

[5] T. Reuters, "OpenCalais: Connect. Every- thing.," 2008. [Online]. Available: http://www.opencalais.com/about. [Accessed: 01-Sep-2013].

[6] S. K. Malik, N. Prakash, and S. A. M. Rizvi, "Semantic Annotation Framework For Intelligent Information Retrieval Using KIM Architecture," vol. 1, no. October, pp. 12–26, 2010.

[7] W. Zaghouani, B. Pouliquen, M. Ebrahim, and R. Steinberger, "Adapting a resource-light highly multilingual Named Entity Recognition system to Arabic," vol. 2, no. 1, p. 600, 2010.

[8] P. Mendes, M. Jakob, and C. Bizer, "DBpedia: A Multilingual Cross-Domain Knowledge Base," *Proc. Eight Int. Conf. Lang. Resour. Eval.*, 2012.

[9] S. Helmreich, D. Farwell, and B. Dorr, "Interlingual annotation of multilingual text corpora," *Proc. NAACL/HLT Work. New Front. Corpus Annot.*, pp. 55–62, 2004.

[10] E. Montiel-Ponsoda and G. A. De Cea, "Modelling multilinguality in ontologies," no. August, pp. 67–70, 2008.

[11] M. Espinoza, A. Gómez-Pérez, and E. Montiel-Ponsoda, "Multilingual and Localization Support for Ontologies," *Semant. Web Res. Appl. Lect. Notes Comput. Sci.*, vol. 5554, pp. 821–825, 2009.

[12] A. M. Iqbal, "A Network of Heterogeneous and Distributed Ontologies for Health and Nutrition Information System," KFUPM, 2014.

[13] V. Uren, P. Cimiano, J. Iria, S. Handschuh, M. Vargasvera, E. Motta, and F. Ciravegna, "Semantic annotation for knowledge management: Requirements and a

survey of the state of the art," *Web Semant. Sci. Serv. Agents World Wide Web*, vol. 4, no. 1, pp. 14–28, Jan. 2006.

[14] H. S. Al-khalifa and A. Al-Wabil, "The Arabic language and the semantic web : Challenges and opportunities," *1st Int. Sysmposium Comput. Arab. Lang. Exhib.*, 2007.

[15] H. Davulcu, S. Vadrevu, S. Nagarajan, and I. V. Ramakrishnan, "OntoMiner: Bootstrapping and Populating. Ontologies From Domain Specific Web Sites," *EEE Intell. Syst.*, vol. 18, no. 5, pp. 24–33, 2003.

[16] "AlchemyAPI," *Web site*, 2014. [Online]. Available: http://www.alchemyapi.com/. [Accessed: 10-Apr-2014].

[17] J. Piskorski and J. Belayeva, "Exploring the Usefulness of Cross-lingual Information Fusion for Refining Real-time News Event Extraction: A Preliminary Study," *Proc. 8th*, no. September, pp. 210–217, 2011.

[18] D. Embley, S. Liddle, D. Lonsdale, and Y. A. Tijerino, "Multilingual ontologies for cross-language information extraction and semantic search," *Inf. Syst. J.*, vol. 6998/2011, pp. 147–160, 2011.

[19] J. Cardeñosa, C. Gallardo, and M. A. de la Villa, "Interlingual information extraction as a solution for multilingual QA systems," *Query Answering Syst.*, 2009.

[20] IMT Holdings, "Rosoka Cloud," 2007. [Online]. Available: http://rosoka.com. [Accessed: 01-Sep-2013].

[21] Semantic Web Company, "PoolParty Extractor," 2011. [Online]. Available: http://semanticweb.org/wiki/PoolParty. [Accessed: 01-Sep-2013].

[22] Rocket Software, "Rocket AeroText," 2011. [Online]. Available: http://www.rocketsoftware.com/products/rocket-aerotext. [Accessed: 01-Sep-2013].

[23] ATTENSITY, "Key Considerations for Choosing a Text Analytics Solutions," *White paper*, 2013.

[24] IsoQuest Inc, "NetOwl ® - Entity Extraction and Entity Analytics for Big Data," 2011. [Online]. Available: http://www.netowl.com. [Accessed: 01-Sep-2013].

[25] T. Roth-Berghofer, B. Adrian, and A. Dengel, "Case Acquisition from Text: Ontology-Based Information Extraction with SCOOBIE for myCBR," *18th Int. Conf. Case-Based Reason. ICCBR 2010*, pp. 451–464, 2010.

[26] A. Al-Nazer, “Agent-based Framework for Semantic Query-Manipulation and Personalized Retrieval of Health and Nutrition Information,” KFUPM, 2014.

[27] H. P. Luong, S. Gauch, and Q. Wang, “Ontology-Based Focused Crawling,” *Int. Conf. Information, Process. Knowl. Manag.*, pp. 123–128, Feb. 2009.

[28] R. de Córdoba, L. F. D’Haro, F. F., Fernández-Martínez, J. M. Guarasa, and J. Ferreiros, “Language Identification based on n-gram Frequency Ranking,” *proceeding INTERSPEECH 2007, 8th Annu. Conf. Int. Speech Commun. Assoc. Antwerp, Belgium*, 2007.

[29] G. R. Ranganathan, Y. ; Biletskiy, and A. Kaltchenko, “Semantic annotation of semi-structured documents,” *Electr. Comput. Eng. CCECE 2008*, vol. 6, pp. 919–922.

[30] M. Popel and Z. Žabokrtský, “TectoMT: modular NLP framework,” *Adv. Nat. Lang. Process.*, pp. 293–304, 2010.

[31] H. Al Khalifa and A. Al Wabil, “Arabic Language and the Semantic Web : Challenges and Opportunities Arabic Research in Semantic Web Technologies,” *1st Int. Sysmposium Comput. Arab. Lang. Exhib.*, 2007.

[32] C. Huang, P. Šimon, S. Hsieh, and L. Prévot, “Rethinking Chinese word segmentation: tokenization, character classification, or wordbreak identification,” *Proc. 45th Annu. Meet. ACL Interact. Poster Demonstr. Sess.*, pp. 69–72,, 2007.

[33] N. Barrett and J. Weber-Jahnke, “Building a biomedical tokenizer using the token lattice design pattern and the adapted Viterbi algorithm,” *BMC Bioinformatics*, 2011.

[34] R. Saetre, K. Yoshida, and A. Yakushiji, “AKANE system: protein-protein interaction pairs in BioCreAtIvE2 challenge, PPI-IPS subtask,” *Proc. Second BioCreative Chall. Eval. Work.*, pp. 209–212, 2007.

[35] R. Alfred, A. Mujat, and J. Obit, “A ruled-based part of speech (RPOS) tagger for malay text articles,” *Intell. Inf. Database Syst.*, vol. 7803, pp. 50–59, 2013.

[36] B. Ali and F. Jarray, “Genetic approach for arabic part of speech tagging,” *Int. J. Nat. Lang. Comput.*, vol. 2, no. 3, 2013.

[37] P. Nugues, “Part-of-Speech Tagging Using Stochastic Techniques,” *An Introd. to Lang. Process. with Perl Prolog Cogn. Technol.*, pp. 163–184, 2006.

[38] D. Wimalasuriya and D. Dou, “Ontology-based information extraction: An introduction and a survey of current approaches,” *J. Inf. Sci.*, no. December 2009, pp. 1–20, 2010.

[39] P. Elango, “Coreference resolution: A survey,” *Univ. Wisconsin, Madison*, 2005.

[40] A. Jivani, “A Comparative Study of Stemming Algorithms,” *Int. J. Comput. Technol. Appl.*, vol. 2, no. 6, pp. 1930–1938, 2011.

[41] S. Khoja, “APT : Arabic Part-of-speech Tagger,” *Proc. Student Work. Second Meet. North Am. Chapter Assoc. Comput. Linguist.*, 2001.

[42] A. Schenker, H. Bunke, M. Last, and A. Kandel, “Clustering of web documents using graph representations,” *Pattern Recognit. Image Anal. Lect. Notes Comput. Sci.*, vol. 265, 2003.

[43] H. Nguyen and T. Cao, “Named Entity Disambiguation: A Hybrid Approach,” *Int. J. Comput. Intell. Syst.*, vol. 5, no. 6, 2012.

[44] Sridevi.U.K and N. .N, “Ontology based Similarity Measure in Document Ranking,” *2010 Int. J. Comput. Appl. (0975 - 8887)*, vol. 1, no. 26, pp. 135–139, 2010.

[45] K. Nebhi, “Named Entity Disambiguation using Freebase and Syntactic Parsing,” *ISWC 2013 Work. Linked Data Inf. Extr.*, 2013.

[46] L. Boytsov, “Indexing methods for approximate dictionary searching: Comparative analysis,” *J. Exp. Algorithmics*, 2011.

[47] T. Rocktäschel, M. Weidlich, and U. Leser, “ChemSpot: a hybrid system for chemical named entity recognition,” *Bioinformatics*, 2012.

[48] H. Van Halteren, W. Daelemans, and J. Zavrel, “Improving accuracy in word class tagging through the combination of machine learning systems,” *Comput. Linguist.*, 2001.

[49] C. Sutton, A. McCallum, and K. Rohanimanesh, “Dynamic conditional random fields: Factorized probabilistic models for labeling and segmenting sequence data,” *Proc. Twenty-First Int. Conf. Mach. Learn.*, pp. 99–107, 2004.

[50] N. Bach and S. Badaskar, “A review of relation extraction,” *Lit. Rev. Lang. Stat. II*, 2007.

[51] R. Mcdonald, F. Pereira, S. Kulick, S. Winters, and P. White, “Simple Algorithms for Complex Relation Extraction with Applications to Biomedical IE,” 2001.

[52] P. Gamallo, M. Garcia, and S. Fern, “Dependency-Based Open Information Extraction,” pp. 10–18, 2012.

[53] A. Minard and A. Ligozat, “Hybrid methods for improving information access in clinical documents: concept, assertion, and relation identification,” *J. Am. Med. Informatics Assoc.*, 2011.

[54] R. Sam, H. Le, and T. Nguyen, “Relation extraction in Vietnamese text using conditional random fields,” *Inf. Retr. Technol.*, 2010.

[55] P. Thomas, M. Neves, and I. Solt, “Relation extraction for drug-drug interactions using ensemble learning,” *Chall. Task Drug-Drug Interact. Extr.*, pp. 11–18, 2011.

[56] M. Missikoff, R. Navigli, and P. Velardi, “The Usable Ontology: An Environment for Building and Assessing a Domain Ontology,” *Proc. Int. Semant. Web Conf.*, pp. 39–53, 2002.

[57] A. Gómez-Pérez and D. Manzano-Macho, “Deliverable 1.5: A survey of ontology learning methods and techniques,” *IST Program. Comm. Eur. Communities, Univ. Politécnica Madrid*, 2003.

[58] P. Cimiano and J. V¨olker, “Text2onto - a framework for ontology earning and datadriven change discovery,” *2nd Eur. Semant. Web Conf.*, 2005.

[59] P. Cimiano and S. Staab, “Learning by googling,” *ACM SIGKDD Explor. Newsl.*, vol. 6, no. 2, pp. 24–33, 2004.

[60] N. Weber and P. Buitelaar, “Web-based Ontology Learning with ISOLDE,” *Proc. Work. Web Content Min. with*, pp. 1–10, 2006.

[61] S. Tamagawa, S. Sakurai, T. Tejima, T. Morita, N. Izumi, and T. Yamaguchi, “Learning a Large Scale of Ontology from Japanese Wikipedia,” *2010 IEEE/WIC/ACM Int. Conf. Web Intell. Intell. Agent Technol.*, pp. 279–286, Aug. 2010.

[62] F. M. Suchanek and G. Weikum, “YAGO: A Core of Semantic Knowledge Unifying WordNet and Wikipedia,” *Int. World Wide Web Conf.*, pp. 697–706, 2007.

[63] B. Hecht and D. Gergle, “The Tower of Babel Meets Web 2 . 0 : User-Generated Content and Its Applications in a Multilingual Context,” 2010.

[64] Wikipedia Contributors, “Wikipedia: The Free Encyclopedia.,” 2012. [Online]. Available: http://en.wikipedia.org/w/index.php?title=Wikipedia.

[65] M. Negri and B. Magnini, “Using wordnet predicates for multilingual named entity recognition,” *Proc. Second Glob. Wordnet …*, pp. 169–174, 2004.

[66] P. Vossen, L. Bloksma, and H. Rodriguez, “The EuroWordNet Base Concepts and Top Ontology,” *Tech. Rep. Deliv. D017*, 1998.

[67] E. Gabrilovich and S. Markovitch, “Computing Semantic Relatedness using Wikipedia-based Explicit Semantic Analysis,” *Proc. Twent. Int. Jt. Conf. Artif. Intell.*, pp. 1606–1611, 2007.

[68] K. Dellschaft and S. Staab, “On How to Perform a Gold Standard Based Evaluation of Ontology Learning,” *Int. Semant. Web Conf.*, pp. 228–241, 2006.

[69] C. Snae and M. Brückner, “FOODS : A Food-Oriented Ontology-Driven System,” *2nd IEEE Int. Conf. Digit. Ecosyst. Technol. (DEST 2008)*, pp. 168–176, 2008.

[70] J. Cantais, D. Dominguez, V. Gigante, L. Laera, and V. Tamma, “An example of food ontology for diabetes control,” *Proc. ISWC Work. Ontol. Patterns Semant. Web*, 2005.

[71] “United States Department of Agriculture.” [Online]. Available: http://www.usda.gov/wps/portal/usda/usdahome. [Accessed: 14-Apr-2014].

[72] G. Salton, A. Wong, and C. Yang, “A vector space model for automatic indexing,” *Commun. ACM*, 1975.

[73] H. Cunningham, D. Maynard, K. Bontcheva, V. Tablan, N. Aswani, I. Roberts, G. Gorrell, A. Funk, A. Roberts, D. Damljanovic, T. Heitz, M. A. Greenwood, H. Saggion, J. Petrak, Y. Li, and W. Peters, *Text Processing with GATE (Version 6)*. The University of Sheeld, Department of Computer Science, 2010.

[74] K. Miller, R. C. and Bharat, “Sphinx: A framework for creating personal,site-specic web crawlers,” *Comput. Networks*, vol. 30, no. (1–7), pp. 119–130., 1998.

[75] S. Hasib, M. Motwani, and A. Saxena, “Importance of Aho-Corasick String Matching Algorithm in Real World Applications,” *Int. J. Comput. Sci. Inf. Technol.*, vol. 4, no. 3, pp. 467–469, 2013.

[76] “Jena Java Libaray.” [Online]. Available: http://jena.sourceforge.net. [Accessed: 10-Jul-2013].

[77] G. Hripcsak and A. Rothschild, “Agreement, the f-measure, and reliability in information retrieval,” *J. Am. Med. Informatics Assoc.*, vol. 12, no. 3, pp. 296–298, 2005.

# Vitae


Name : Saeed Yousef Ahmed Albukhitan

Nationality : Saudi Arabian

Date of Birth : 1st of July 1973

Email : albokhitan@gmail.com

Address : P.O. Box 10053, Jubail 31961


Academic Background :

BS in Computer Science, KFUPM University, 1997

Master in Computer Science, KFUPM University, 2007

Publications :

Several publications in the area of machine learning

Several publications in the area of Semantic Web

Research Areas

Machine Learning, Semantic Web, Database Management