

WEB SERVICES FOR ONTOLOGY BASED INFORMATION EXTRACTION

L. C. T. Silva



University of Moratuwa, Sri Lanka.
Electronic Theses & Dissertations
www.lib.mrt.ac.lk 138234L

Dissertation submitted in partial fulfillment of the requirements for the degree Master
of Science

Department of Computer Science and Engineering

University of Moratuwa
Sri Lanka

March 2015

WEB SERVICES FOR ONTOLOGY BASED INFORMATION EXTRACTION



University of Moratuwa, Sri Lanka.
Electronic Theses & Dissertations
www.lib.mrt.ac.lk

L.C.T.SILVA

138234L

This dissertation was submitted to the Department of Computer Science and Engineering of the University of Moratuwa in partial fulfillment of the requirements for the Degree of MSc in Computer Science specializing in Software Architecture

Department of Computer Science & Engineering
University of Moratuwa, Sri Lanka

March 2015

DECLARATION

The work included in this report was done by me, and only by me, and the work has not been submitted for any other academic qualification at any institution.

.....

Chamendri Silva

Date

I certify that the declaration above by the candidate is true to the best of my knowledge and that this report is acceptable for evaluation for the CS6999 MSc Research Project



University of Moratuwa, Sri Lanka.
Electronic Theses & Dissertations
www.lib.mrt.ac.lk

.....


Daya Chinthana Wimalasuriya (PhD)

Date

ABSTRACT

The amount of data contained in a textual format has increased rapidly in the recent past. Such data includes web sites, documents of business organizations, etc., and contain lots of information. Information Retrieval (IR) is a field that allows identifying relevant document for a given query out of all these available documents. Information Extraction is taking another step in this direction. Instead of returning the set of documents that contains the relevant information, IE recognizes and returns the information among the natural text in these documents.

Ontology is defined as the “*formal, explicit specification of a shared conceptualization*”. It contains classes, properties, individuals and values to represent data in a certain domain. Most of the time in Ontology-Based Information Extraction, an IE technique is used to discover individuals for classes and values for properties to build ontology for a given domain. However, sometimes these classes and properties also identified as part of the IE technique rather than using a template with the pre-identified classes and properties in the Ontology.

 A traditional Ontology Based Information Extraction system contains two main operations, ontology construction and ontology population. In the component-based approach defined in the “Ontology-Based Components for Information Extraction (OBCIE)”, the operation of constructing ontology is not changed. However, the operation to populate the ontology is refined in to a pipeline of three separate components: pre-processors, information extractors and aggregators.

By developing these components as web services, we have provided the ability for other applications to use them to extract the information out of any text based document. To demonstrate this concept, we have developed an application that accepts a set of text documents, and extracts useful information. It uses “metadata files”, which are dependent of the domain in which the ontology is created and populate the given ontology.

ACKNOWLEDGMENTS

I would like to express profound gratitude to my advisor, Dr. Daya Chinthana Wimalasuriya, for his invaluable support, encouragement, supervision and useful suggestions throughout this research work. His continuous guidance enabled me to complete my work successfully.



University of Moratuwa, Sri Lanka.
Electronic Theses & Dissertations
www.lib.mrt.ac.lk

TABLE OF CONTENTS

DECLARATION	iii
ABSTRACT.....	iv
ACKNOWLEDGMENTS	v
TABLE OF CONTENTS.....	vi
LIST OF FIGURES	viii
LIST OF ABBREVIATIONS	ix
Chapter 1 Introduction	1
Chapter 2 Literature Review	4
2.1 Ontology	5
2.1.1 Ontology creation	6
2.1.2 Ontology Population.....	6
2.2 Information Extraction.....	7
2.2.1 Ontology Based Information Extraction.....	9
2.3 Ontology Based Components for Information Extraction	11
2.3.1 Web Services for Ontology Based Information Extraction.....	12
2.4 Web Services	12
2.4.1 Service Oriented Architecture.....	14
2.4.2 SOAP.....	15
2.4.3 REST	16
2.5 Related Work	18
2.5.1 The OwlExporter	18
2.5.2 SOBA	19
Chapter 3 Methodology	21
3.1 High Level Architecture	22
3.2 Extraction Methods.....	22
3.2.1 Extraction Rules	22
3.2.2 Two Phase Classification.....	23
3.2.3 Tools used.....	24
3.3 Framework for Building RESTful Web Services	26
3.3.1 Apache Wink	26
3.3.2 Project Jersey	27
3.3.3 JBoss RESTEasy	27
3.3.4 Restlet Framework.....	28
3.3.5 Comparison.....	28

3.4 Implementation	29
3.4.1 Front End	30
3.4.2 Preprocessor	31
3.4.3 Extractor	32
Chapter 4 Results and Conclusion	37
4.1 Implementation	38
4.2 Conclusion and Future Work.....	40
REFERENCES	41
APPENDIX A: Sample Metadata Files – Information Extractor	44
APPENDIX B: Sample Metadata File – Two-Phase Classifier.....	46



University of Moratuwa, Sri Lanka.
Electronic Theses & Dissertations
www.lib.mrt.ac.lk

LIST OF FIGURES

Figure 1 - Relationship between two classes	5
Figure 2 - Part of generalization/specialization hierarchy	5
Figure 3 - General Architecture of an OBIE System [3]	10
Figure 4 – Pipeline for Ontology Population in OBCIE	12
Figure 5 – General Process of Engaging Web Services [13]	14
Figure 6 – Obtaining REST architectural style from WWW	17
Figure 7 - General workflow of the OwlExporter [23]	19
Figure 8 – Pipeline used in SOBA	20
Figure 9 – High-level architecture	22
Figure 10 – Performance Metrics for different JAX-RS implementation [32]	29
Figure 11 – Message passing between web services	31
Figure 12 – A sample request json object sent to preprocessor	32
Figure 13 – Information Extractor method: a sample response json object from preprocessor	32
Figure 14 – Two-phase classifier method: a sample response json object from preprocessor.	33
Figure 15 – Information Extractor method: request json object for processing the metadata file	34
Figure 16 – Two-phase classifier method: request json object for processing the metadata file	34
Figure 17 – Information Extractor method: response json object for processing the metadata file	35
Figure 18 - Two-phase classifier method: response json object for processing the metadata file	35
Figure 19 – Information Extractor Method: A sample request json object to extractor	35
Figure 20 – Two-phase classifier Method: A sample request json object to extractor	36
Figure 21 - Information Extractor Method: a sample response json object to extractor	36
Figure 22 – UI developed to populate ontology using information extractor method	38
Figure 23 – UI developed to populate ontology using two-phase classifier method	39
Figure 24 - UI developed to view a given ontology	39

LIST OF ABBREVIATIONS

IR	Information Retrieval
IE	Information Extraction
OBIE	Ontology Based Information Extraction
SOA	Service Oriented Architecture
NLP	Natural Language Processing
HTML	Hype-Text Markup Language
XML	Extensible Markup Language
OBCIE	Ontology Based Components for Information Extraction
WWW	World Wide Web
WSDL	Web Service Definition Language
UDDI	Universal Description Discovery and Integration
SOAP	Simple Object Access Protocol
REST	Representational State Transfer
RPC	Remote Procedure Call
HTTP	Hyper Text Transfer Protocol
EAI	Enterprise Application Integration
URI	Uniform Resource Identification
SOBA	SmartWeb Ontology Based Annotation
GATE	General Architecture for Text Engineering
JAX-RS	Java API for RESTful Web Services
JAPE	Java Annotation Pattern Engine
GDM	GATE Document Manager
CREOLE	Collection of Re-usable Objects for Language Engineering
GGI	GATE Graphical Interface
WEKA	Waikato Environment for Knowledge Analysis
MALLET	Machine Learning for Language Toolkit



University of Moratuwa, Sri Lanka.
Electronic Theses & Dissertations
www.lib.mru.ac.lk

JSON	Java Script Object Notation
MIME	Multi-purpose Internet Mail Extension
CDDL	Common Development and Distribution License
API	Application Program Interface



University of Moratuwa, Sri Lanka.
Electronic Theses & Dissertations
www.lib.mrt.ac.lk

Chapter 1

Introduction



University of Moratuwa, Sri Lanka.
Electronic Theses & Dissertations
www.lib.mrt.ac.lk

The amount of data contained in a textual format has increased rapidly in the recent past. Such data includes web sites, documents of business organizations, etc., and contain lots of information. Finding useful information in a large set of documents is difficult. As a result, the field Information Retrieval (IR) was born out of necessity in 1950s [1]. IR works for text-based documents and when a query is entered, which contains a word or a set of words; it is searched among all the available documents and returned the most relevant document. The challenge in IR is to return the most relevant set of documents and reducing the amount of non-relevant documents returned.

Information Extraction (IE) is taking this a step further. It scans texts and finds information that is included implicitly and explicitly and combines and structures this data. This is quite useful in many industries such as governments, publishers and finance companies where they want to get the scattered information in to a single database and find the relationships among them [2].

Ontology is defined as the “*formal, explicit specification of a shared conceptualization*”. It typically contains components such as classes, individuals, properties and values [3] [4] [5].

Ontology Based Information Extraction (OBIE) has emerged as a sub field of IE where the ontologies are used by the information extraction process and the output is presented using ontology. Ontologies are commonly defined for a particular domain and since in most cases IE is concerned about extracting information in a particular domain, ontologies are very useful [3] [4].

Most of the time in Ontology-Based Information Extraction, an IE technique is used to discover individuals for classes and values for properties to build ontology for a given domain. However, these classes and properties are identified as part of the IE technique rather than using a template. OBIE systems can have one or more ontologies [5].

A traditional Ontology Based Information Extraction system contains two main operations, Ontology construction where classes and properties are identified and

Ontology population where individuals and values are identified for the previously identified classes and properties from the texts available in the given domain [5].

In the component-based approach defined in the “Ontology-Based Components for Information Extraction (OBCIE)”, the operation of constructing ontology is not changed. However, the operation to populate the ontology is refined in to a pipeline of three separate components, pre-processors, information extractors and aggregators [5].

The pre-processor component takes various types of documents and converts them to texts that can be understood by the information extractors. For example, the pre-processor will remove all the unnecessary characters that will hinder the extraction process. The information extractor components takes these pre-processed documents as the input, tag them based on the rules defined using a given Ontology and extract the information. The aggregator component combines the output from all the information extractor components and does some adjustments [5].

The objective of this research is to use service-oriented architecture to separate these components and implement them as separate entities so that anyone can use these components all together or separately. The system developed in the “Ontology-Based Components for Information Extraction (OBCIE)” contains components that perform various functionalities in the different stages of creating ontologies. These components can be implemented as web service to adhere to SOA.

Creating Ontologies has two steps, ontology construction and ontology population. In ontology construction, the classes and properties are identified while in ontology population individuals for the classes and values for the properties are identified. The scope of this research is limited to developing web services for components in ontology population, and populates an ontology using the extracted information.

The rest of the document is structured in the following manner. Chapter 2 contains the literature review, which covers the theoretical aspects of ontology, ontology based information extraction, web services and different implementation of web services as well as related work done on this area. Chapter 3 contains the implementation details and the architecture of the system developed. Finally, Chapter 4 contains the results and the conclusions drawn from this research.

Chapter 2

Literature Review



University of Moratuwa, Sri Lanka.
Electronic Theses & Dissertations
www.lib.mrt.ac.lk

2.1 Ontology

“Ontology” is a term originated from philosophy, which describes the existence of being in the world. The field Artificial Intelligence (AI) in Computer Science has adopted this name to describe the world to a program. Ontology is defined as “*a formal, explicit specification of a shared conceptualization*”, which means that an abstract model of a specific domain is presented in such a way that it can be understood by a machine, and is accepted by a group of people [4].

Ontologies provide a shared understanding of a given domain thus eliminating the conceptual and terminological confusion. Therefore, there will be less miscommunication among different people as well as among different computer systems. This shared understanding increases the interoperability among the systems [6].

When defining Ontologies, entities of a particular domain (also known as concepts or classes) and the relationship among these entities are identified. Then these ontologies are used to formally model the structure of a system. For example, consider Automobile domain. In this domain, Car and Person can be considered as entities. The relationship between these two classes can be described as “a person drives a car”, as depicted in Figure 1. In addition, a car can have a generalizations/specializations hierarchy, which is also known as taxonomy. For example, a car can be manual gear or auto gear [7]. This is depicted in Figure 2.

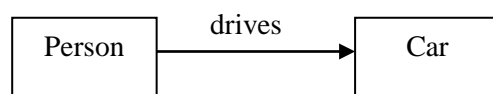


Figure 1 - Relationship between two classes

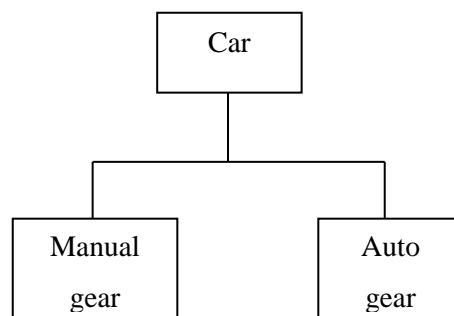


Figure 2 - Part of generalization/specialization hierarchy

“Individuals” are the real world objects and belongs to classes. In other words, a class is a collection of individuals. Examples of individuals for the class “car” can be Ford, Nissan etc. “Properties” are a collection of relationships between individuals such as *has_identification_number*, *owned_by* etc. These are used to describe the relationship between individuals. Individuals are related to other individuals via properties.

2.1.1 Ontology creation

There is no standard method to create Ontology [6]. Following steps gives a basic understanding of how ontology of a particular domain is developed [8].

- Identify the classes in the ontology, as they are the focus of most ontologies and describes the concepts in the domain.
- Arrange the classes in to a hierarchical (super class - subclass) structure.
- Identify the properties of these classes. These properties give additional details that describe the class.

There are many Ontologies widely available in different domains. It is always best to use a readily available ontology and refine it to our needs as it will provide a strong foundation to building the ontology and will save lot of time. Most of the ontologies are available in electronic form and is possible to import in to any ontology development environment [8].

There are libraries that contain re-usable ontologies, “Ontoligua Ontology Library” or “DAML Ontology Library” is some of them. There are many ontologies which are commercially available as well [8].

2.1.2 Ontology Population

Once the classes (entities) and properties are identified in an Ontology, then information belonging to these classes and properties can be identified from natural text. This process is known as “Ontology Population”. In other words, Ontology Population can be described as identifying instances for classes and property values for properties [5]. This can be done either manually, which is time consuming and not efficient or using an automated method. Ontology Based Information Extraction (OBIE) is a newly emerged sub-field, which is described under section 2.2.1, in which

most systems provide the ability to populate an ontology by extracting information from natural text.

2.2 Information Extraction

Information Extraction (IE) is a newly emerging field in which widely scattered texts in natural language is reduced in to a simple database. This database is created in such a way that it contains the entities in a particular domain; i.e. information extraction identifies these entities in natural language texts. On the other hand, Information Retrieval, which is a more mature field than IE, retrieves a sub set of relevant documents from a larger set of documents. This does not provide the required information directly, but rather returns a set of documents that contains the information. A good example of an IR system is a web search engine. However, in IE, the information is extracted from these documents and presented in a structured manner [2].

IE has the potential to be used in many industries in which the end user requires a large amount of information to make decisions, such as finance companies, banks, publishers and governments. As an example, Lloyds of London pays a large sum of money to individuals to find information about daily ship sinking, which can be easily done using an information extraction system, saving both time and money. Another example will be searching movie reviews for actors and directors, especially if the person is playing a non-standard role such as, a famous actor playing the role of a movie director in some movies [2].

Nevertheless, IE techniques are not widely used and commercialized as IR, even though it is more powerful, as it provides the information itself rather than the documents that contains the information. As identified in [5], there are two reasons for this. One is the requirement of a set of templates for each domain, which needs to be created manually by a domain expert to be filled by the extraction process. In many cases, creating these templates is not practical. Another reason is that the IE systems are usually developed in such a way that there is no clear separation between the domains. Therefore using these systems in a new situation is difficult. Due to these reasons, the cost and complexity involved in developing an IE system is much higher than an IR system.

IE process follows several steps when processing texts and produces unambiguous data as the output. First, it takes documents in different formats as the input and processes them using information extractors. These information extractors identify the entities and relationships among these entities from the documents and ignore the non-relevant information in them. Then this extracted information can be displayed to the user as the output, stored in a database or used as an input to another system [9].

There are two types of information extraction techniques. One is the “Traditional IE” where it uses the supervised learning techniques such as Hidden Markov Model and self-supervised methods. These methods require a set of manually tagged documents with rules to learn from and then apply these rules to new documents. This model performs well when used in similar set of documents, but it performs poorly when used in a documents in different style or domain [9]. The second technique provide a solution to this problem is called “Open IE”, where rather than providing a pre-defined set of rules for the system to learn, this method find the relations from the texts [5] [9]. However, as mentioned in [5], we should make sure that the relations discovered in this method are useful, and in a structured manner so that they can be reused.



University of Moratuwa, Sri Lanka.
Electronic Theses & Dissertations
www.lib.mrt.ac.lk

In IE, the performance can be measured by precision and recall as done in IR. In IR precision indicates the number of correctly identified documents as a portion of the total number of identified documents and recall indicates the number of correctly identified documents as a portion of the total number of correct documents available.

$$Precision = \frac{|{\{Relevant\}} \cap {\{Retrieved\}}|}{|{\{Retrieved\}}|}$$

$$Recall = \frac{|{\{Relevant\}} \cap {\{Retrieved\}}|}{|{\{Relevant\}}|}$$

A weighted average of precision and recall is called F-measure and it is given in the equation below. In this equation, β denotes the weighting of precision vs. recall, which is also called F1 score. In most cases this is given the value 1, thus giving equal weighting for precision and recall [3].

$$F - Measure = \frac{(1 + \beta^2) * Precision * Recall}{(\beta^2 * Precision) + Recall}$$

2.2.1 Ontology Based Information Extraction

Ontology contains a model that provides the application the ability to process the information in it and reason about it. These provide a very good vocabulary along with a formal semantics. Therefore, using Ontology in information extraction systems enables using manually composed rules in a more flexible way. IE and Ontology can be used in two ways. One is using Ontology to interpret natural language. The other is to use IE to populate and improve the Ontology [9].

This sub-field of IE, which recently emerged, uses Ontology in the information extraction process and returns the extracted information through the same Ontology. I.e. in IE, Ontology is used to guide the information extraction process. As identified in [3], listed below are the key characteristics of OBIE.

- Since IE is a sub-field of Natural Language Processing (NLP) and OBIE is a sub-field of IE, it can be said that OBIE processes unstructured and semi-structured texts, such as web pages, text files etc.
- An OBIE system can either use an existing Ontology as the input to guide the information extraction process or construct Ontology to be used in the extraction process. Therefore when identifying common characteristics, it will be correct to say that an OBIE presents the output using Ontology.
- The word “guide” can describe the interaction between Ontology and an IE system, which means that no new IE process is invented, but the existing ones are oriented to identify the components of Ontology.

In [3], these key factors and definitions of IE is used to provide a definition for OBIE. *“An ontology-based information extraction system: a system that processes unstructured or semi-structured natural language text through a mechanism guided by ontologies to extract certain types of information and presents the output using ontologies.”*

As the output of the OBIE is given as an ontology, it can be used in Semantic Web, where meaning is given to the web and a software agent can easily perform a task by going through the documents in the web. In normal web, information is produced targeting a wide range of users with normal user and machine at the two extreme ends of the user spectrum. Information targeted for normal user includes TV commercials, poetry etc, while information targeted for machines includes databases, sensor outputs

etc. Semantic Web tries to remove this distinction and allow the machines to understand the content of the web in a much smarter manner. Ontologies defined will allow the software agents to identify related information even though they are defined in different ways in different documents and can be seen as the backbone of Semantic Web. Therefore, OBIE is important to Semantic Web because it populates an Ontology by extracting information from a set of documents [3] [10].

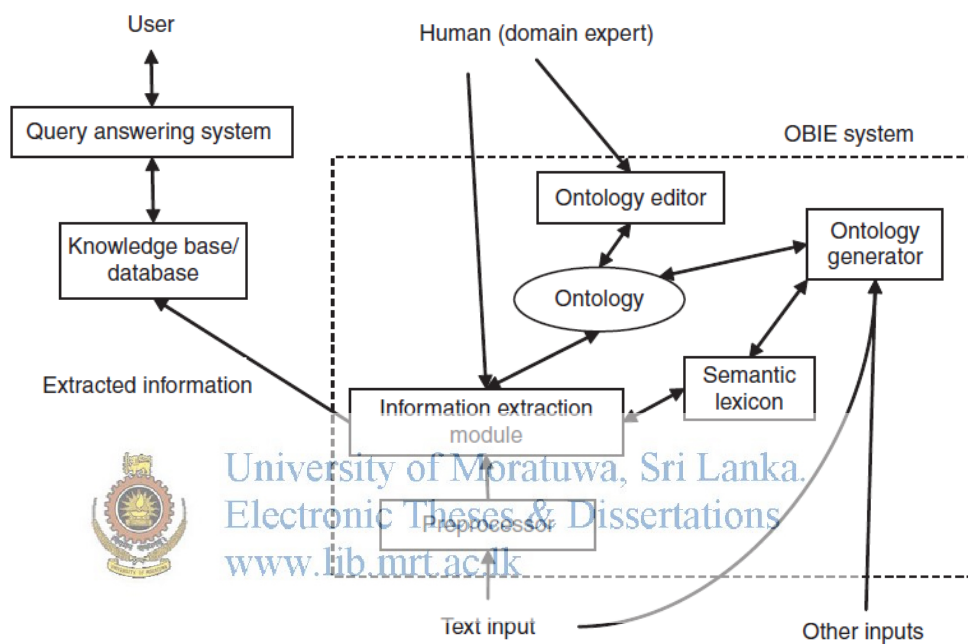


Figure 3 - General Architecture of an OBIE System [3]

In [3], a common architecture is identified for the OBIE system, which is given in figure 3. However, in practice, not all the components given in this architecture are used. For example, some systems will use an already existing Ontology, rather than generating one by scanning the documents. As depicted in figure 3, OBIE is a part of a larger system. The preprocessor converts the input text in to format that can be understood by the IE module. For example, a web page can be preprocessed and remove all the HTML tags, before feeding it to the IE module. The IE module is guided by an Ontology and uses techniques such as regular expressions, gazetteer lists, construction of partial parse trees, analyzing HTML/XML tags as explained before and classification techniques to extract the information. Ontology generator component can generate an Ontology to be used by the IE system. This process might use a semantic lexicon as well as human assistance when generating the Ontology. As OBIE system is part of a larger system, the output is stored in a database or a

knowledge base. The larger system, for example a query answering system can refer to this database or knowledge base to perform its functionality [3].

2.3 Ontology Based Components for Information Extraction

Even though the information extraction field has improved rapidly over the time, it is still not widely used as information retrieval. As mentioned in [5], one of the main reasons is the cost of setting up a new IE system in a new domain. This requires creating a new ontology for that domain, scanning and processing the documents to identify individuals and values for classes and properties in the ontology. These tasks involve a large cost, rather than in information retrieval, where the only cost associated is scanning the documents to build and maintain an index.

In a given domain, there exist multiple ontologies falling under one of the following two scenarios. One scenario is ontologies having different perspective of the same domain. For example in the domain of marriages, an ontology can have two classes called Husband and Wife, while another can have an object property names "isSpouseOf". The other scenario is specializing in sub-domains, such as the domain university can have North American universities, British universities as the specialized sub domains with each sub-domain containing characteristics only relevant to that sub-domain [11]. Therefore, when using multiple ontologies of the same domain in an IE system, we can see that, it is possible to re-use the information extractors, which extracts individuals for a class and values for a property, if a class or a property in one ontology is related to a class or a property in another ontology [5].

In [5], the above-mentioned idea is extended to apply information extractors from different domains. To make the re-use of the information extractors straightforward, the authors have introduced a platform and metadata. The platform is a domain independent implementation of the IE techniques and metadata contains the domain details. In this component-based approach, the population of ontology is refined in to a pipeline with three steps. Each of these steps is a component and they are:

- Preprocessor Component – This pre-process the documents in to a format that can be used by the information extractor component.
- Information Extractor Component – This contains a platform for an IE technique which is domain independent and the metadata that are associated

with a particular class or a property represented by the information extractor. We can also consider that the platform and metadata can be two sub-components within this Information Extractor component.

- **Aggregator Component** – This combines the results produced by each individual information extractor.

In this pipeline, the output of the previous component is given as the input to the next component as depicted in figure 4.

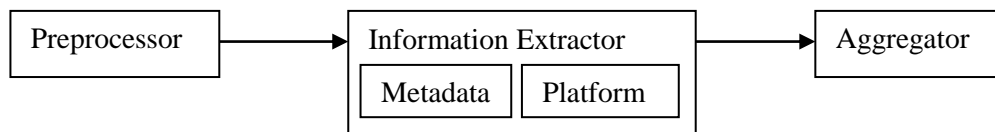


Figure 4 – Pipeline for Ontology Population in OBCIE

A complete OBIE system, which contains the above pipeline, can have preprocessor components, information extractor components aggregators. In addition, it can also have multiple ontologies and an ontology construction module, which scans the documents to identify the classes, and properties.



University of Moratuwa, Sri Lanka.
Electronic Theses & Dissertations
www.lib.mrt.ac.lk

2.3.1 Web Services for Ontology Based Information Extraction

The pipeline introduces separate components to process each step of the pipeline. By introducing each component as a web service, it allows anyone to access these components and build their own pipeline. Also these components can be used separately as well, thus providing the ability for anyone to use them and build their own application.

2.4 Web Services

Web services provide a method to communicate between two machines within a network. In W3C working group note 11February 2004, the following definition for web services is given [12]. “A *Web service* is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP-

messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.”

The purpose of web services is to provide a set of functionalities on behalf of the owner of the web service. It can be either a “provider” or a “requester”. Usually the “provider” initiates the communication, where the “requester” wants to use some functionalities provided by the “provider”, although there are cases where the communication happens the other way as well [13].

Web services communicate with each other using messages. A “service” defines the set of functionalities exposed by the web service and an “agent” implements this. The agent can be implemented using different technologies without any changes to the “service” [13].

There are several steps a web service needs to follow, in order to have a successful communication. Explained below are these steps of communication when SOAP (Simple Object Access Protocol) is used. First, the two web services that need to communicate should “become known” to each other, or at least the requester should know about the provider. Web service registry called UDDI (Universal Description Discovery and Integration) is used for this purpose, which is a platform independent XML based registry for web services. Then the two web services agree on the service description and the semantics that is to be used for the communication and separately send these agreed upon information to their agents. WSDL (Web Service Definition Language), an XML based document is used to describe the web services. Finally, the agents use this information to send messages among each other, and are done using SOAP (Simple Object Access Protocol) [13] [14]. Figure 5 depicts this process.

Based on how the web services communicate with each other, we can identify two styles of web services, namely RPC (Remote Procedure Call) style and REST (Representational State Transfer). SOA is an example for RPC-style web service architecture.

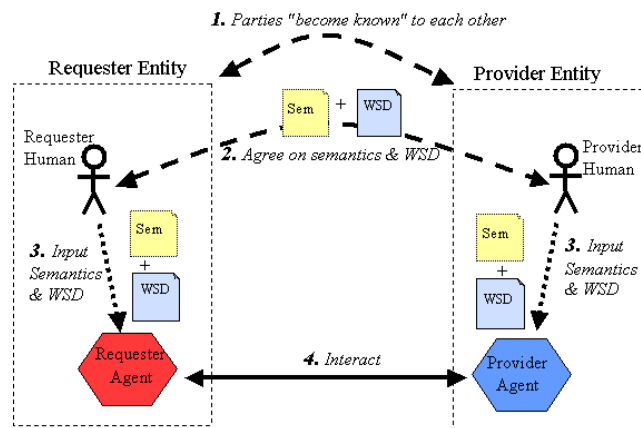


Figure 5 – General Process of Engaging Web Services [13]

2.4.1 Service Oriented Architecture

This is an architectural style with the goal of achieving loose coupling among different components and can be implemented using web services. Each of these components performs tasks they are specialized to do and interact with other components to get additional services. In order to provide the services to other components, each component has clearly defined universally available interfaces. Component use messages to interact with each other and these messages follow a schema that is understood by all the components [15].

As mentioned in [15], there are four rules that an architecture should follow before it can be called service oriented.

- The components are responsible for solving the problem. Therefore, the messages should be descriptive rather than instructive.
- The messages should be written in a format that is understood by all the components involved in the system.
- As the environment change, the system should have the ability to change accordingly. Therefore, extensibility is very important in SOA and can be achieved by making the messages used extensible. Even though, introducing restrictions to the messages will make it easier to understand, it will affect the extensibility of the system. Therefore, it is important to find a good balance between the restrictions and extensibility.

- There should be a mechanism for a consumer to discover a service provider that has the functionalities sought by the consumer.

One drawback on SOA is that it is RPC- style architecture. Therefore, the services are abstracted based on RPC. The interface defines a set of discrete number of procedures to be accessed, limiting the set of services produced by the service provider. If the number of procedures exposed is increased, the complexity and the cost of the system will increase thus making it difficult to scale the system [16].

In SOA, SOAP messages are used to communicate between the web services.

2.4.2 SOAP

This is a lightweight protocol proposed by W3C, for information exchange in a distributed, decentralized environment using XML. SOAP consists of three parts [17]:

- SOAP envelope – a framework for defining what is in the message, who is the recipient and whether it is optional or mandatory.
- SOAP encoding rules – defines a serialization mechanism that can be used to exchange instances of application-defined data types.
- SOAP RPC representation – A convention that can be used to represent remote procedure calls and responses.

SOAP messages are generally uses HTTP envelopes and HTTP POST to transfer data from one web service to another. It is an XML document, which contains mandatory SOAP envelope and SOAP body, and an optional SOAP header.

Listed below are some of the advantages and disadvantages of using SOAP [14] [16] [18].

Advantages

- Supported by major software platforms, thus making it easier to transform an interface defined in any major platform (such as Java or C#) in to a SOAP interface and generate the WSDL and required stubs.
- It opened a new market for middleware products that aims to simplify some of the EAI challenges.
- SOAP has built in error handling.

- Support from other standards such as WDSL, WS-*

Drawbacks

- SOAP promotes high coupling between the producer and consumer.
- Does not offer any separation of concerns as the supported model does not separate network centric operations from local operations.
- Additional parsing and packaging is required on both client and server sides to pack and unpack a SOAP package.
- SOAP package contains additional information; therefore, the payload is rather high.

2.4.3 REST

REST (Representational State Transfer) is an architectural style first proposed by Fielding in his doctoral dissertation [19] for distributed hypermedia systems. He applied a set of constrained to the World Wide Web (WWW) to define this style. These set of constraints in many ways abstracted the principles that make the World Wide Web scalable [20]. In the first step, Separation of concern is applied to the WWW, where the result is the client-server architecture style. Then the communication in the server is constrained to be stateless, thus resulting in the client-stateless-server architecture style. Figure 6 depicts how these constraints are applied to obtain REST architectural style from WWW.

The interface defined for REST is optimized for large grain of hypermedia data transfer, and contains four interface constraints. They are identification of resources, manipulation of resources through representations, self-descriptive messages and hypermedia as engine of application state. [19].

REST is similar to navigating web links and when a link is selected, the information is moved from where it is stored to where it will be used [19] [20]. The most known REST implementation is the HTTP protocol. HTTP allows the ability to uniquely locate the resource as well as tell how to operate the resource. In REST, request and response is done through four operations namely GET, POST, PUT and DELETE. The functionalities of each method are described below.

- GET – Allows retrieving the current state of the resource. This method is safe, as it does not change the server state.
- POST – Transfer the state of the resource in to a new state
- PUT – Add a new resource
- DELETE – Removes an existing resource

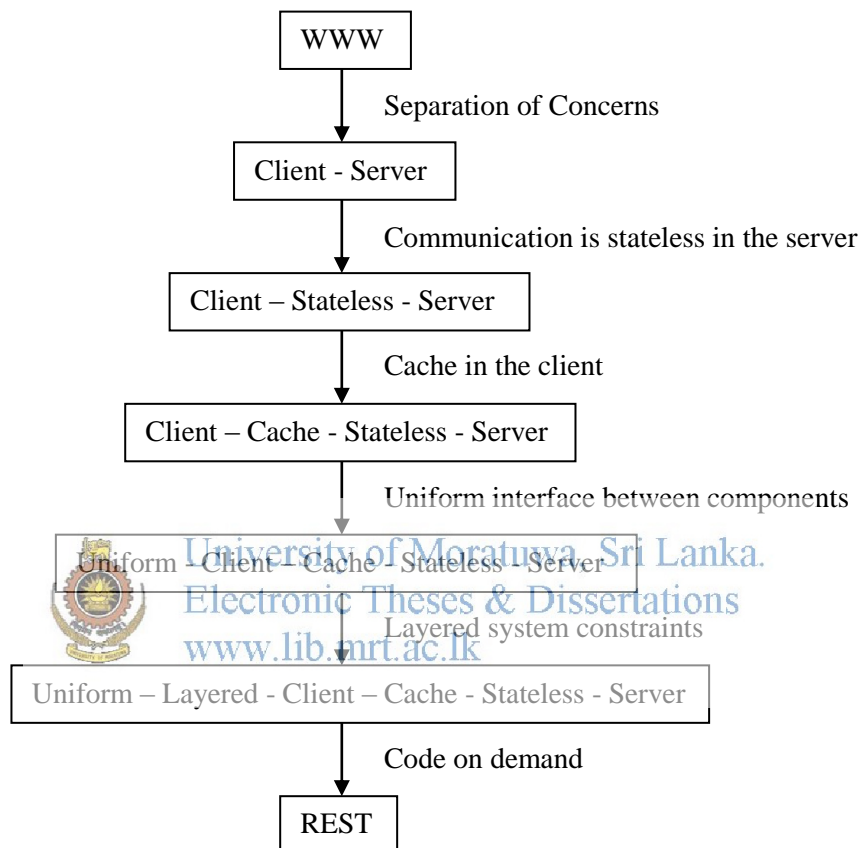


Figure 6 – Obtaining REST architectural style from WWW

The security model of REST is simple and effective. It abstracts everything in to resources and every resource uses a URI to identify itself. These URIs can be exposed or hidden based on whether the resource identified by the URI is required to be hidden or exposed. In addition, we can achieve different security policies by setting different security settings for each of the four operations described above [16].

Listed below are some of the advantages of using REST [14] [16] [18] [21].

Advantages

- In REST, separation of concern is clear.
- Same protocol is used to access all the resources, and a resource can be replaced by just changing its name.
- The standards used in the messages sent between the producer and the consumer does not contain any contextual information of the message.
- High performance in REST is due to inherent simplicity.
- As HTTP is a widely used, no additional platforms are required.
- Small learning curve, and relies less on tools.

Drawbacks

- Tool support is minimal, as the focus was in SOAP.
- Since there is a limited API, interfaces might be perceived as too thin when managing everything as resources.
- Using HTTP as a protocol inside an EAI middleware is not cost efficient.
- Assumes a point-to-point communication model.



University of Moratuwa, Sri Lanka.
Electronic Theses & Dissertations
www.lib.mrt.ac.lk

2.5 Related Work

Many systems exist that are implemented using OBIE. Given below are two such systems. Both systems follow a pipeline to populate the ontology, while SOBA has implemented the components in the pipeline as web services, which is similar to the high-level architecture proposed in the methodology of this document.

2.5.1 The OwlExporter

GATE (General Architecture for Text Engineering) [22] is a framework which facilitates NLP application development. A large number of components support standard tasks such as tokenization, POS tagging, and these components can be easily assembled into more complex application pipelines. However, populating an ontology based on these analysis is not an easy task. As a solution to this “OwlExporter” is introduced. This automates the ontology population process and can be used in an existing pipeline in GATE [23].

The idea behind OwlExporter is to get the annotated set of documents from an existing pipeline in GATE, and use them to populate the given ontology. Figure 7 below depicts this [23].

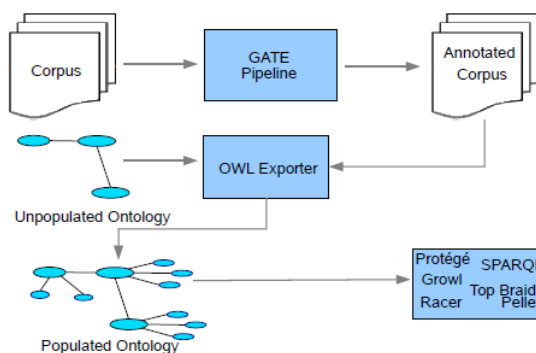


Figure 7 - General workflow of the OwlExporter [23]

2.5.2 SOBA

“SmartWeb” is a multi-model dialog system, which derives its answers from unstructured resources such as web. SOBA (SmartWeb-Ontology Based Annotation), which is a component in SmartWeb, automatically populates the knowledge base on soccer using information extraction and respect to an underlying ontology. The resources used to populate the knowledge base are found on web [24].

SOBA consists of three parts. The web crawler, which monitors the relevant web sites such as FIFA site on a daily basis, downloads and sends the relevant documents, which have a recent update, to the linguistic annotation component. In the linguistic annotation component, the retrieved documents are annotated using gazetteer, part-of-speech and a set of rules defined for soccer specific entries such as actors (players, referees etc.), teams and tournaments. Then these linguistic annotated documents are processed further by the semantic transformation component, which maps the annotated entities in to classes and their properties in the ontology. Each of these components is implemented as web services. Figure 8 depicts the pipeline followed by SOBA to populate the ontology using the latest updates in the sites it monitors in the web [24].

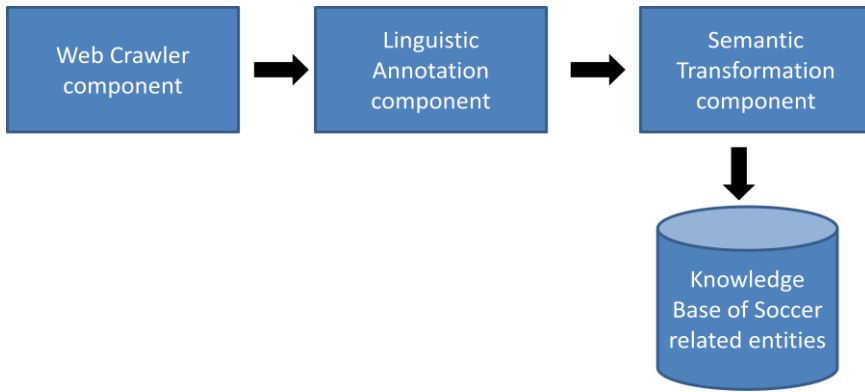


Figure 8 – Pipeline used in SOBA



University of Moratuwa, Sri Lanka.
Electronic Theses & Dissertations
www.lib.mrt.ac.lk

Chapter 3

Methodology



University of Moratuwa, Sri Lanka.
Electronic Theses & Dissertations
www.lib.mrt.ac.lk

3.1 High Level Architecture

This section describes the high-level architecture proposed for the system “Web Services for Ontology Based Information Extraction”. This architecture follows the pipeline proposed in [5] to a certain extent and identifies each component as a web service. The last component, the aggregator is not developed separately as a web service since it will not add any value by having it as separate web service.

A new web service called the “front end, is introduced to demonstrate how these web services can be used to implement a system. This will accept a corpus, metadata files and an ontology and will return the populated ontology.

These web services follow the REST architecture style. The major reason behind using REST, rather than SOAP is that, SOAP introduces additional overhead to the messages passed between the two web services. As this proposed system passes a large amount of text between the web services, this additional overhead introduced by SOAP will have a significant effect. The below depicts the high-level architecture of this system.

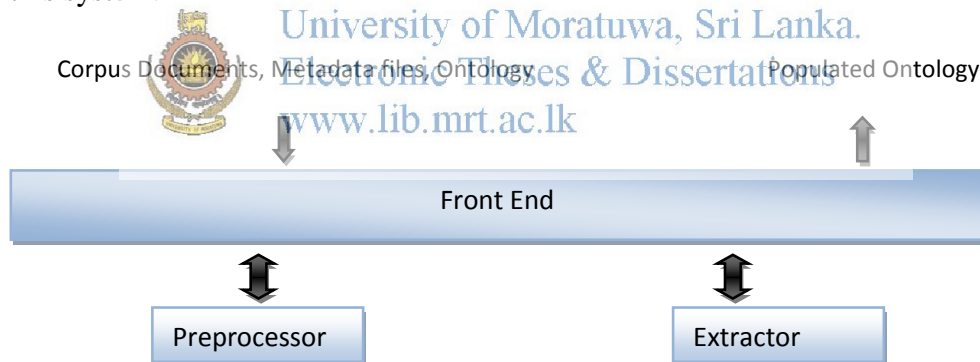


Figure 9 – High-level architecture

3.2 Extraction Methods

The implemented system contains two extraction methods to extract information from the given text. These two extraction methods are developed in [5], and it is used to implement the web services. Given below is a detailed description of each extraction method.

3.2.1 Extraction Rules

The metadata file for extraction rules method contains the class, the identifier name and JAPE grammar rules to extract the information. JAPE stands for Java Annotation

Pattern Engine, and it can identify regular expressions in annotations on documents. This grammar consists of a set of phases and each phase consists of a set of pattern or action rules. There are two sides in this grammar. The left hand side consists of an annotation pattern description and the right hand side consists of annotation manipulation statements [25].

When this metadata file is sent to the extractor, it extracts the JAPE grammar from the xml file and store as a .jape file under a unique id. When a text string is received with this unique id, a GATE pipeline is started to process this text string and annotate it using the previously stored gate file. Then this annotated string is further processed to extract the information from the annotated text string and return the extracted information for the given class and the identifier name [5].

3.2.2 Two Phase Classification

In this method, first the related sentences are identified and then the words within these sentences are identified. There are two different approaches to combining these above two phases.

- Pipeline approach – word level classifier only operates on the sentences selected by the sentence level classifier.
- Combined approach – The word level classifier operates on all sentences, but uses the sentence selected by the sentence level classifier as one feature in classification.

Out of these two methods, second one yield best results, so it is incorporated in to the platform. For the two phase classification technique, it is found that the following classifiers are best for each phase [5].

1. Sentence level classifier – Bayesian technique (specially naïve bayes models) used with bagging (Weka system for this and any other classification techniques)
2. Word level classifier – Conditional Random Field (CRF), which is a sequence tagging technique (uses Mallet system for this)

The metadata file for the two-phase classifier also contains the class and the identifier name. However, unlike in extraction rules where it contains the JAPE rule, this

contains a list of words, synsets and gazetteer lists to be used during the extraction process. The extractor will extract this information from the metadata file; store them as separate lists under a unique id. Then this id can be used to identify which metadata should be used to process a text string.

The processing of a text document is done in two levels; sentence level and word level. In the sentence level, the text document is first annotated using the gazetteer information given in the metadata file. Then this text is annotated with the keys for that document and finally it is annotated with the synsets and words given in the metadata file. Once the text string is annotated, a Weka training file is created from this string, one for the training corpus and one for the test corpus. Both of these files are then used to train using Weka, which will give output files for training corpus and test corpus [5].

Once the training is done by weka, the word level processing is done separately to the training corpus and the test corpus. In this level of processing, first the preprocessed text string is annotated so that it can be understood by Kylin. Some of the annotations given to the strings are "@single_letter@", "@atleast_one_digit@", "@two_digits@", "@contains_underscore@" and "@stopword@". Then this annotated string is used to generate a training file for Kylin and run it. The output file generated from running Kylin is then converted to be used in Mallet and train it using this converted files for both training corpus and test corpus and give the final output for two phase extraction method [5].

3.2.3 Tools used

The following sections give a brief description of the tools used to develop the extractor component in the pipeline. The use of these tools are mentioned in the above section for each extraction method.

3.2.3.1 GATE (General Architecture for Text Engineering)

GATE is originally developed at University of Sheffield in 1995 and now it is widely used for all sorts of natural language processing tasks. It is designed to meet the following objectives [26].

- Provide the ability to interchange information between other language engineering modules without specifically mentioning a theoretical approach.
- Provide the ability to integrate modules written in any language.
- Provide a graphical user interface to visualize data and manage corpora, and the ability to evaluate and refine other language engineering modules and systems built from them.

GATE comprises of three main elements: GDM: GATE document manager based on TIPSTER document manager, CREOLE: a collection of re-usable objects for language engineering and the GGI: GATE graphical interface [26].

The GDM is the central repository which stores all the information generated about the texts it processes and provides a uniform API for the language engineering systems to manipulate the data they produce and consume [26].

CREOLE modules are usually wrappers for pre-existing language engineering modules such as taggers, parsers and lexicon index. These objects can be developed from scratch as well. This element does the text analyzing part in GATE [26].

The GGI encapsulates the GDM and CREOLE to provide the ability to build and test language engineering components and systems interactively [26].

The GATE API allows other applications to initialize and run CREOLE modules and use the output.

3.2.3.2 Weka

Weka (Waikato Environment for Knowledge Analysis) is a tool developed by the Machine Learning Group in java, which provides a set of machine learning algorithms for data mining tasks. Weka provides the ability for data preprocessing, classification, regression, clustering, association rules and visualization. Weka works under the assumption that data is available in a single flat file or relation where each data point is described by a fixed number of attributes [27].

3.2.3.3 Mallet

Mallet (Machine Learning for Language Toolkit), is a Java-based package for statistical natural language processing, document classification, clustering, topic

modeling, information extraction, and other machine learning applications to text. Mallet includes a variety of document classification algorithms such as Naïve Bayes, Maximum Entropy, and Decision Trees. This also includes algorithms for sequence tagging such as Hidden Markov Models, Maximum Entropy Markov Models and Conditional Random Fields [28].

3.3 Framework for Building RESTful Web Services

JAX-RS (Java API for RESTful Web Services) is a Java API that provides support in creating web services according to the REST architectural pattern. This uses annotations, which helps to map a resource class to a web resource. These annotations are:

- @Path – the relative path to a resource or a method
- @GET, @POST, @DELETE, @PUT and @HEAD – Http request types of a resource
- @Consumes – the accepted media type of a request
- @Produces – the response media type of a request

This API is developed by Oracle Corporation and is under CDDL License (Common Development and Distribution License). From version 1.1 it is an official part of Java EE 6. This means that JAX-RS can be used in any Java EE development without any configuration. JAX-RS 2.0 is the current major release of JAX-RS [29] [30].

The next sub sections contain a high-level overview of some of JAX-RS implementations and a comparison between them.

3.3.1 Apache Wink

Apache Wink is a fully compliant java based solution for implementing and consuming REST based web services. This provides a set of reusable and extensible classes and interfaces that can be used by the developer to build REST based web services. This provides two modules: a server module developing web services and a client module for consuming web services [31].

Apache wink reflects the design principles of a REST web service, by providing the developer with a set of java classes that enable the implementation of “Resources” and “Representations” and the association between them [31].

Apache Wink supports a wide variety of industry standard data formats such as XML, Atom, Json etc. Also this provides easy spring integration through an additional module that comes with the core framework, providing features such as registering resources and providers as classes or spring beans [32].

3.3.2 Project Jersey

Project Jersey is an open source, dual license JAX-RS implementation developed by Sun for building RESTful web services. This is more than a reference implementation and provides API that extends JAX-RS toolkit with additional features and utilities to further simplify RESTful service and client development. This is shipped as part of Glassfish application server download [32] [33].

The client API of Jersey provides the ability to invoke any RESTful web service, not just JAX-RS compliant web services. This also allows a pluggable HTTP implementation and supports common data formats such as Atom, Json and MIME Multipart data. Jersey also supports extension based support for Spring framework and Google Guice framework [32].



University of Moratuwa, Sri Lanka.
Electronic Theses & Dissertations
www.lib.mrt.ac.lk

3.3.3 JBoss RESTEasy

This is a JAX-RS compliant framework developed by Red Hat and can be used in any serverlet-based environment. This is licensed under the GNU Lesser General Public License (LGPL). Embeddable server implementation for junit testing, client framework that leverages JAX-RS annotations so that it is easy to write HTTP clients, server in memory cache, local response cache are some of the features of RestEasy [32] [34].

RESTEasy has a built-in client proxy framework that is a slightly different way of writing RESTful Java clients. It works by reusing the JAX-RS annotations on the client side, using the annotations to turn a method call into an equivalent HTTP request [32].

RESTEasy supports most of the popular data formats such as XML, Json, Atom and multipart. This also supports integration with frameworks and standards such as Enterprise Java Beans (EJB) technology, spring and Google Guice [32].

3.3.4 Restlet Framework

This framework is slightly different from other JAX-RS implementation, as it existed before JAX-RS is implemented and finalized. This was designed as a light weight REST based java framework with pluggable extensions for different functionalities. An extension exists as part of the Restlet framework that implements JAX-RS specification. This is fully open source and can be used under Apache Software License [32] [35].

This provides a client API that makes it easy to consume any HTTP-based services, not just JAX-RS services. The Restlet Framework is based on a connectors and components architecture, where a connector enables communication between components, usually by implementing a network protocol [32].

This framework too supports many of the common data formats such as XML, Json, Atom etc. In addition, Restlet supports integration with frameworks and standards such as Spring, Jetty, Grizzly, JAXB etc. [32].

3.3.5 Comparison



University of Moratuwa, Sri Lanka.
Electronic Theses & Dissertations
www.lib.mrt.ac.lk

When comparing the above four JAX-RS implementation with respect to the suitability of using in the implementation of web services for OBCIE, one of the main aspects considered is the performance of the implementation. The following performance test done in [32] to measures the relative transactional throughput, measured in transactions per second, of these implementation. This test can be used to get a general idea of their relative performance characteristics.

Figure 10, which is taken from the study performed in [32], plots the number of transactions per second for each JAX-RS implementation. It can be seen that Apache wink, RESTEasy and Jersey have similar performance while RESTlet framework lags behind a little.

However, for OBCIE web services extensive data format support adds more value and with sending a large amount of text between web services, a high throughput is important. Therefore, Apache wink is chosen as the JAX-RS implementation to develop the web services.

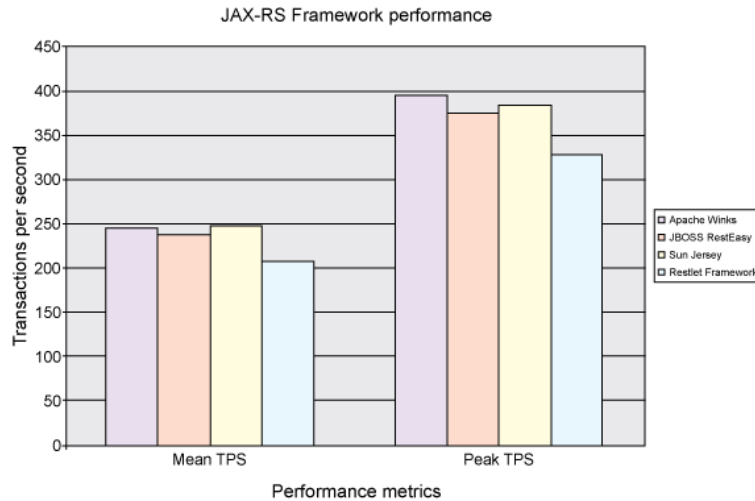


Figure 10 – Performance Metrics for different JAX-RS implementation [32]

3.4 Implementation

The three main web services, preprocessor, extractor and the front end, in this application as described in section 3.1 communicate using REST methods. The front end web services accepts user input and send this information to each web service to process the string and extract the information based on the given metadata file. A unique id, based on the metadata file name, information extraction method and the current time is used to identify which metadata should be used to extract information from the given string. This ID is returned from the extractor when the metadata files are sent to be processed.

Also throughout the application, each document is assigned an integer as the file number from the front end, and this number is passed with the string containing the content of the document, along with the total number of documents. This is useful in two phase classifier method, as described under section 3.2.1.2. Even though this is not used in the information extractor method, as each document is fully processed independent of each other, if the sending application sends these two values, the extractor accepts these values to maintain uniformity between the two methods.

Since all of these components are developed as web services, anyone can use both preprocessor and extractor separately or together to develop their own applications.

3.4.1 Front End

This is the front end of the application and contains the user interface as well as the base logic that handles the pipeline for each information extraction method. This also provides the ability to view a populated and unpopulated ontology. For the information extractor method, the front end accepts three inputs from the user; an ontology (.owl file), corpus (this can contain many files) and metadata files.

All the communication between the web services is done using Apache Wink. In REST web services all the resources are accessed through URIs. Apache Wink provides the functionality to call the desired operation on the resources using the given URL, whether it is POST, DELETE, PUT or GET.

When the user gives the input, the front end will first send the metadata files to the extractor and accepts the returned unique ids. Then for each document, it will start the pipeline. First, the text from the document is send to the preprocessor, along with the extraction method for the string to be preprocessed. Then the returned string from the preprocessor is send to the extractor, with all the unique ids, to extract the information from the string. Then the returned string is used to add the information to the ontology using GATE API [27]. Once all the documents followed the pipeline, the ontology is fully populated, and it will be available for download.

When using the two-phase classifier as the extraction method, it accepts a training corpus, keys for the training corpus, test corpus, keys for the test corpus, metadata files and the ontology. A single document may contain multiple key files, therefore all the text from these key files are aggregated in to one string for both test corpus and training corpus. This method will also follow the same pipeline as information extractor method, with the addition of sending the aggregated key string along with the preprocessed string and the unique ids for metadata to the extractor.

Figure 11 below depicts how the messages are passed between the web services.

A limitation introduced in this method is that, in order to identify which keys files belong to which document, the key files should follow a standard method to name the files. It should be named as “<name of the document>_<key file number>”. As an example, consider a training document is named as “training_doc_1.txt”, then the key files should be names as “training_document_1_1.txt”, “training_document_1_2.txt”.

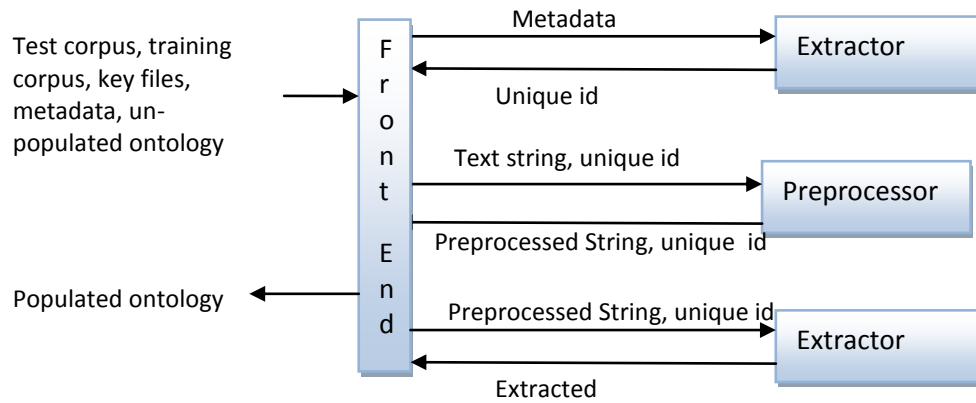


Figure 11 – Message passing between web services

The front end also allows ontologies to be viewed as an additional functionality. To provide this, it uses OWL API to convert and render the ontology file (.owl) file in to html format.

3.4.2 Preprocessor

The main functionality of this web service is to process a string and format it based on the given information extraction method and return the preprocessed string. For each information extraction method, the output string will be slightly different. The information extraction method is sent as an integer value, where “1” indicates information extractor method, and “2” indicates two-phase classifier method.

For the information extractor method, since the strings are given as input to GATE method calls in the extractor, it should be made GATE compatible by removing the “@” and replacing it with a space character.

For the two-phase classifier method, in addition to making the string GATE compatible, it is further processed by splitting the text in to sentences.

For each information extraction method, there is a unique URI in preprocessor to perform POST operation. Only the POST operation is used in preprocessor, as this web services does not maintain any resources and will return the output for each input. Given below are the two URIs for the two information extraction methods implemented in this system.

- Information Extractor : “http://localhost:8080/obcie-preprocessor/rest/corpus-preprocessor/extractionrules”
- Two-Phase Classifier : “http://localhost:8080/obcie-preprocessor/rest/corpus-preprocessor/two-phase-classifier”

This web service accepts a json object as the input, which contains the string to be processed and the file number, and the total number of files. A sample of an input json object is given below in figure 12.

```
<"fileName":"Afganistan.txt","partString":"country- Afghanistan ; capital- Kabul ; languages- Pashto , Dari ; religion- Islam ; currency- Afghani ; population- 30822848 ; ","partNumber":1>
```

Figure 12 – A sample request json object sent to preprocessor

Once the string is preprocessed, the output string is returned from the preprocessor as a json object. This output differs based on the information extraction method used. A sample of each output string is given below in figure 13 and 14.

```
<"gatePreprocessed":"country- Afghanistan ; capital- Kabul; languages- Pashto , Dari ; religion- Islam ; currency- Afghani ; population- 30822848 ; ","fileName":"Afganistan.txt","partNumber":1>
```

Figure 13 – Information Extractor method: a sample response json object from preprocessor



3.4.3 Extractor

This web service does the main processing in the information extraction process. It accepts the information extractor metadata files, processes them, stores the information in the web service, and returns a unique id to identify and use these stored information during the actual information extraction process. As in the pre-processor, the information extraction method is sent as an integer value when processing the metadata, where “1” indicates information extractor method, and “2” indicates two-phase classifier method.

As described in section 3.4.2, the extractor web service uses URIs to give access to it as well. As this web service provides two separate functionalities: process the metadata files and extract the information using the processed metadata information and preprocessed string, there are unique URIs to access each of these functionalities.

```

"preprocessed":":1::SAN SALVADOR, 3 JAN 90 -- [REPORT] ARMED FORCES PRESS C
OMMITTEE, COPREFAI [TEXT] THE ARCE BATTALION COMMAND HAS REPORTED THAT ABOUT 50
PEASANTS OF VARIOUS AGES HAVE BEEN KIDNAPPED BY TERRORISTS OF THE FARABUNDO MARTI
NATIONAL LIBERATION FRONT [FMLN] IN SAN MIGUEL DEPARTMENT.\n::2::ACCORDING TO
THAT GARRISON, THE MASS KIDNAPPING TOOK PLACE ON 30 DECEMBER IN SAN LUIS DE LA
REINA.\n::3::THE SOURCE ADDED THAT THE TERRORISTS FORCED THE INDIVIDUALS, WHO
WERE TAKEN TO AN UNKNOWN LOCATION, OUT OF THEIR RESIDENCES, PRESUMABLY TO INCOR
PORATE THEM AGAINST THEIR WILL INTO CLANDESTINE GROUPS.\n::4::MEANWHILE, THREE
SUBVERSIVES WERE KILLED AND SEVEN OTHERS WERE WOUNDED DURING CLASHES YESTERDAY
IN USulutAN AND MORAZAN DEPARTMENTS.\n::5::THE ATONAL BATTALION REPORTED THAT
ONE EXTREMIST WAS KILLED AND FIVE OTHERS WERE WOUNDED DURING A CLASH YESTERDAY A
FTERNOON NEAR LA ESPERANZA FARM, SANTA ELENA JURISDICTION, USULUTAN DEPARTMENT.\n
::6::IT WAS ALSO REPORTED THAT A SOLDIER WAS WOUNDED AND TAKEN TO THE MILITARY
HOSPITAL IN THIS CAPITAL.\n::7::THE SAME MILITARY UNIT REPORTED THAT THERE WAS
ANOTHER CLASH THAT RESULTED IN ONE DEAD TERRORIST AND THE SEIZURE OF VARIOUS
KINDS OF WAR MATERIEL NEAR SAN RAFAEL FARM IN THE SAME TOWN.\n::8::IN THE COUN
TRY'S EASTERN REGION, MILITARY DETACHMENT NO.4 REPORTED THAT A TERRORIST WAS KIL
LED AND TWO OTHERS WERE WOUNDED DURING A CLASH IN LA RAMERA STREAM, SAN CARLOS,
MORAZAN DEPARTMENT.\n::9::AN M-16 RIFLE, CARTRIDGE CLIPS, AND AMMUNITION WERE
SEIZED THERE.\n::10::MEANWHILE, THE 3D INFANTRY BRIGADE REPORTED THAT PONCE BATA
LION UNITS FOUND THE DECOMPOSED BODY OF A SUBVERSIVE IN LA FINCA HILL, SAN MI
GUEL.\n::11::AN M-16 RIFLE, FIVE GRENADES, AND MATERIAL FOR THE PRODUCTION OF
EXPLOSIVES WERE FOUND IN THE SAME PLACE.\n::12::THE BRIGADE, WHICH IS HEADQUAR
TERED IN SAN MIGUEL, ADDED THAT THE SEIZURE WAS MADE YESTERDAY MORNING.\n::13::
NATIONAL GUARD UNITS GUARDING THE LAS CANAS BRIDGE, WHICH IS ON THE NORTHERN TR
UNK HIGHWAY IN APOPA, THIS MORNING REPELLED A TERRORIST ATTACK THAT RESULTED IN
NO CASUALTIES.\n::14::THE ARMED CLASH INVOLVED MORTAR AND RIFLE FIRE AND LASTED
30 MINUTES.\n::15::MEMBERS OF THAT SECURITY GROUP ARE COMBING THE AREA TO DE
TERMINE THE FINAL OUTCOME OF THE FIGHTING.\n", "gatePreprocessed": " SAN SALVAD
OR, 3 JAN 90 -- [REPORT] ARMED FORCES PRESS COMMITTEE, COPREFAI [TEXT] THE ARCE
BATTALION COMMAND HAS REPORTED THAT ABOUT 50 PEASANTS OF VARIOUS AGES HAVE BEEN
KIDNAPPED BY TERRORISTS OF THE FARABUNDO MARTI NATIONAL LIBERATION FRONT [FMLN]
IN SAN MIGUEL DEPARTMENT. ACCORDING TO THAT GARRISON, THE MASS KIDNAPPING TOOK
PLACE ON 30 DECEMBER IN SAN LUIS DE LA REINA. THE SOURCE ADDED THAT THE TERROR
ISTS FORCED THE INDIVIDUALS, WHO WERE TAKEN TO AN UNKNOWN LOCATION, OUT OF THEIR
RESIDENCES, PRESUMABLY TO INCORPORATE THEM AGAINST THEIR WILL INTO CLANDESTINE
GROUPS. MEANWHILE, THREE SUBVERSIVES WERE KILLED AND SEVEN OTHERS WERE WOUNDED
DURING CLASHES YESTERDAY IN USULUTAN AND MORAZAN DEPARTMENTS. THE ATONAL BAT
ALION REPORTED THAT ONE EXTREMIST WAS KILLED AND FIVE OTHERS WERE WOUNDED DURIN
G A CLASH YESTERDAY AFTERNOON NEAR LA ESPERANZA FARM, SANTA ELENA JURISDICTION,
USULUTAN DEPARTMENT. IT WAS ALSO REPORTED THAT A SOLDIER WAS WOUNDED AND TAK
EN TO THE MILITARY HOSPITAL IN THIS CAPITAL. THE SAME MILITARY UNIT REPORTED
THAT THERE WAS ANOTHER CLASH THAT RESULTED IN ONE DEAD TERRORIST AND THE SEIZUR
E OF VARIOUS KINDS OF WAR MATERIEL NEAR SAN RAFAEL FARM IN THE SAME TOWN. IN
THE COUNTRY'S EASTERN REGION, MILITARY DETACHMENT NO.4 REPORTED THAT A TERRORIS
T WAS KILLED AND TWO OTHERS WERE WOUNDED DURING A CLASH IN LA RAMERA STREAM, SAN
CARLOS, MORAZAN DEPARTMENT. AN M-16 RIFLE, CARTRIDGE CLIPS, AND AMMUNITION WER
E SEIZED THERE. MEANWHILE, THE 3D INFANTRY BRIGADE REPORTED THAT PONCE BATA
LION UNITS FOUND THE DECOMPOSED BODY OF A SUBVERSIVE IN LA FINCA HILL, SAN MIGUE
L. AN M-16 RIFLE, FIVE GRENADES, AND MATERIAL FOR THE PRODUCTION OF EXPLOSIVES
WERE FOUND IN THE SAME PLACE. THE BRIGADE, WHICH IS HEADQUARTERED IN SAN MIGUEL
, ADDED THAT THE SEIZURE WAS MADE YESTERDAY MORNING. NATIONAL GUARD UNITS G
UARDING THE LAS CANAS BRIDGE, WHICH IS ON THE NORTHERN TRUNK HIGHWAY IN APOPA, T
HIS MORNING REPELLED A TERRORIST ATTACK THAT RESULTED IN NO CASUALTIES. THE ARM
ED CLASH INVOLVED MORTAR AND RIFLE FIRE AND LASTED 30 MINUTES. MEMBERS OF THAT
SECURITY GROUP ARE COMBING THE AREA TO DETERMINE THE FINAL OUTCOME OF THE FIGHTI
NG. ", "fileName": "test_page-1.txt", "partNumber": 2}

```

Figure 14 – Two-phase classifier method: a sample response json object from preprocessor

Again, the POST operation is used when processing the metadata file and extracting information from a preprocessed string, and the DELETE operation is used to remove the metadata information from the extractor web service. Given below are the URIs that can be used to access the two functionalities in the extractor web service.

- Information Extractor: “http://localhost:8080/obcie-extractor/rest/corpus-extractor/extractionrules”
- Two-phase Classifier: “http://localhost:8080/obcie-extractor/rest/corpus-extractor/two-phase-classifier”
- To process the metadata: “http://localhost:8080/obcie-extractor/rest/platform”

Based on the URI used, the extractor chooses the extraction method. A detailed description of these two extraction methods is given in section 3.2. Given below in

figure 15 and 16 are the sample json object used to send a metadata file to the extractor.

```
<"platformType":1,"fileName":"CapitalX.xml","rules":"<?xml version="1.0"?><InformationExtractor xmlns="http://aimlab.cs.uoregon.edu/obie/StandardXMLSchema/1.0/"><Language>EN</Language><IETechnique>http://aimlab.cs.uoregon.edu/obie/IETechniques/LinguisticExtractionRules.xml</IETechnique><Concept>Capital</Concept><ConceptType>Class</ConceptType><IdentifierName>hasName</IdentifierName><Feature>rule-input:Lookup</Feature><Feature>rule:<<Lookup.minorType == city>>:loc-></Feature></InformationExtractor>"}
```

Figure 15 – Information Extractor method: request json object for processing the metadata file

```
<"platformType":2,"fileName":"HumanTarget.xml","rules":"<?xml version="1.0"?><InformationExtractor xmlns="http://aimlab.cs.uoregon.edu/obie/StandardXMLSchema/1.0/"><Language>EN</Language><IETechnique>http://aimlab.cs.uoregon.edu/obie/IETechniques/TwoStepClassification.xml</IETechnique><Concept>HumanTarget</Concept><ConceptType>Class</ConceptType><IdentifierName>hasName</IdentifierName><Feature>sentence_no</Feature><Feature>wordcount</Feature><Feature>artice_sentence_num</Feature><Feature>noun_count</Feature><Feature>stanford_noun_count</Feature><Feature>verb_count</Feature><Feature>stanford_verb_count</Feature><Feature>adjective_count</Feature><Feature>stanford_adjective_count</Feature><Feature>adverb_count</Feature><Feature>stanford_adverb_count</Feature><Feature>number_count</Feature><Feature>stanford_number_count</Feature><Feature>other_count</Feature><Feature>stanford_other_count</Feature><Feature>firstname</Feature><Feature>lastname</Feature><Feature>military_rank</Feature><Feature>job_title</Feature><Feature>terrorist_organization</Feature><Feature>instrument_type</Feature><Feature>word:attack</Feature><Feature>word:kidnap</Feature><Feature>word:kill</Feature><Feature>word:threaten</Feature><Feature>word:burn</Feature><Feature>word:civilian</Feature><Feature>word:peasant</Feature><Feature>word:leader</Feature><Feature>word:politician</Feature><Feature>word:officer</Feature><Feature>word:military</Feature><Feature>word:innocent</Feature><Feature>word:terrorist</Feature><Feature>word:soldier</Feature><Feature>word:guerrilla</Feature><Feature>word:armedforces</Feature><Feature>word:armed</Feature><Feature>word:indivudual</Feature><Feature>word:suspected</Feature><Feature>word:commando</Feature><Feature>word:officers</Feature><Feature>word:army</Feature><Feature>word:murderer</Feature><Feature>word:leftist</Feature><Feature>word:rightwing</Feature><Feature>word:kidnapper</Feature><Feature>word:navy</Feature><Feature>word:airforce</Feature><Feature>word:revolutionary</Feature><Feature>word:movement</Feature><Feature>word:spokesman</Feature><Feature>word:spokeswoman</Feature><Feature>word:clandestine</Feature><Feature>word:arson</Feature><Feature>word:shoot</Feature><Feature>word:dynamite</Feature><Feature>word:tnt</Feature><Feature>word:explosive</Feature><Feature>word:grenade</Feature><Feature>word:bullet</Feature><Feature>word:axe</Feature><Feature>word:fuse</Feature><Feature>synset:attack</Feature><Feature>synset:kidnap</Feature><Feature>synset:kill</Feature><Feature>synset:threaten</Feature><Feature>synset:burn</Feature><Feature>synset:civilian</Feature><Feature>synset:peasant</Feature><Feature>synset:leader</Feature><Feature>synset:politician</Feature><Feature>synset:officer</Feature><Feature>synset:military</Feature><Feature>synset:innocent</Feature><Feature>synset:shoot</Feature><Feature>word_no</Feature><Feature>sentence_word_no</Feature><Feature>gazetteer:firstname</Feature><Feature>gazetteer:lastname</Feature><Feature>gazetteer:military_rank</Feature><Feature>gazetteer:job_title</Feature><Feature>gazetteer:terrorist_organization</Feature><Feature>gazetteer:instrument_type</Feature><Feature>pos_tag</Feature></InformationExtractor>"}
```

Figure 16 – Two-phase classifier method: request json object for processing the metadata file

The figure 17 and 18 below gives sample response json objects from the extractor after processing the metadata file, containing the unique id and the other relevant information extracted from the metadata file.

```
<"platformType":1,"uniquePlatformId":"ER_Capital_1424798716118","extractionConceptName":"CapitalX","identifierName":"hasName","extractionConcept":"Capital">
```

Figure 17 – Information Extractor method: response json object for processing the metadata file

```
<"platformType":2,"uniquePlatformId":"TPC_HumanTarget_1424804636001","propertyName":"hasName","className":"HumanTarget">
```

Figure 18 - Two-phase classifier method: response json object for processing the metadata file

This web service accepts a json object as the input which contains the preprocessed string from which the information to be extracted and the file number, and the total number of files. Total number of files is not mandatory when using the information extractor method. A sample of an input json object for each information extraction method is given below. When using both preprocessor and the extractor as a pipeline, the response json object from the preprocessor can be used as the request object to the extractor.

```
<"gatePreprocessed":"country- Afghanistan ; capital- Kabul; languages- Pashto , Dari ; religion- Islam ; currency- Afghani ; population- 30822848 ; ","fileName":"Afghanistan.txt","partNumber":1>
```

Figure 19 – Information Extractor Method: A sample request json object to extractor



University of Moratuwa, Sri Lanka
Electronic Theses & Dissertations
www.lib.mrt.ac.lk

When accepting a string to be processed by the extractor using two-phase classifier methods, it should contain the text string and the relevant key strings in the same object. As described in the section 3.2.2, for two-phase classifier, all the training files and test files should be provided at once for Weka to train. Therefore, since the extractor has the ability to receive multiple files, it needs to collect all the files before sending to Weka. The document number and the number of documents parameters in the input json object is used for this purpose. The extractor assumes that the document numbers are sent in order, and when the document number matches the number of documents, it performs the rest of the steps in the two-phase classifier extraction process and returns the extracted results. Until the last document is sent, for two-phase classifier, the extractor will return a null json object. This does not happen in the information extractor, since it can fully process each input string separately and return the extracted results.

```

{"preprocessed":"::1::LIMA, 15 JAN 90 (EFE) -- [TEXT] THE PERUVIAN POLICE HAVE REPORTED THAT AN ATTACK WITH EXPLOSIVES IN FRONT OF A BUILDING WHERE U.S. DIPLOMATS RESIDE IN THE MIRAFLORES NEIGHBORHOOD IN LIMA HAS LEFT ONE PERSON INJURED.
\n::2::THE TERRORIST ATTACK WAS LAUNCHED FROM A MOVING VEHICLE, FROM WHICH UNIDENTIFIED INDIVIDUALS THREW THREE DYNAMITE CHARGES THAT INJURED A PASSERBY AND DAMAGED THE FRONT OF THE BUILDING, ACCORDING TO THE POLICE.\n::3::THE POLICE SOURCE INDICATED THAT THOSE RESPONSIBLE FOR THE CRIME HAVE NOT YET BEEN IDENTIFIED.\n::4::THE SOURCE NOTED THAT IN 1984, THE FIRST ATTACK ON U.S. INTERESTS WAS CARRIED OUT BY THE GUEVARIST TUPAC AMARU REVOLUTIONARY MOVEMENT, WHEN ACTIVISTS FROM THAT GROUP FIRED AT THE FRONT OF THE U.S. EMBASSY IN DOWNTOWN LIMA.\n::5::A CAR BOMB EXPLODED IN FRONT OF THE U.S. EMBASSY RESIDENCE IN THE PERUVIAN CAPITAL 3 YEARS LATER, AND AFTER THAT, EXPLOSIVES WERE HURLED AT THE DOOR OF A HOUSE IN THE SAN BORJA NEIGHBORHOOD, WHERE U.S. MARINES RESIDED.\n", "documentType": "TRAINING", "fileNumber": 1, "keys": {"Location": {"hasName": "PERU"}, "Location": {"hasName": "LIMA"}, "Location": {"hasName": "MIRAFLORES"}, "MAINSUBCLASS": "BOMBING", "Incident": {"hasStageOfExecution": "ACCOMPLISHED"}, "Instrument": {"hasName": "THREE DYNAMITE CHARGES"}, "Instrument": {"hasInstrumentType": "DYNAMITE"}, "Perpetrator": {"hasPerpetratorType": "TERRORIST ACT"}, "Perpetrator": {"hasPerpetratorIndividual": "UNIDENTIFIED INDIVIDUALS"}, "PhysicalTarget": {"hasName": "BUILDING"}, "PhysicalTarget": {"hasPhysicalTargetType": "DIPLOMAT OFFICE OR RESIDENCE"}, "PhysicalTarget": {"hasNumber": 1}, "PhysicalTarget": {"hasNationality": "UNITED STATES"}, "PhysicalTarget": {"hasIncidentEffectOnPhysicalTarget": "SOME DAMAGE"}, "HumanTarget": {"hasDescription": "PASSERBY"}, "HumanTarget": {"hasDescription": "CIVILIAN"}, "HumanTarget": {"hasNumber": 1}, "HumanTarget": {"hasIncidentEffectOnHumanTarget": "INJURY"}, "HumanTarget": {"hasName": "DIPLOMATS"}}, "gatePreprocessed": "LIMA, 15 JAN 90 (EFE) -- [TEXT] THE PERUVIAN POLICE HAVE REPORTED THAT AN ATTACK WITH EXPLOSIVES IN FRONT OF A BUILDING WHERE U.S. DIPLOMATS RESIDE IN THE MIRAFLORES NEIGHBORHOOD IN LIMA HAS LEFT ONE PERSON INJURED. THE TERRORIST ATTACK WAS LAUNCHED FROM A MOVING VEHICLE, FROM WHICH UNIDENTIFIED INDIVIDUALS THREW THREE DYNAMITE CHARGES THAT INJURED A PASSERBY AND DAMAGED THE FRONT OF THE BUILDING, ACCORDING TO THE POLICE. THE POLICE SOURCE INDICATED THAT THOSE RESPONSIBLE FOR THE CRIME HAVE NOT YET BEEN IDENTIFIED. THE SOURCE NOTED THAT IN 1984, THE FIRST ATTACK ON U.S. INTERESTS WAS CARRIED OUT BY THE GUEVARIST TUPAC AMARU REVOLUTIONARY MOVEMENT, WHEN ACTIVISTS FROM THAT GROUP FIRED AT THE FRONT OF THE U.S. EMBASSY IN DOWNTOWN LIMA. A CAR BOMB EXPLODED IN FRONT OF THE U.S. EMBASSY RESIDENCE IN THE PERUVIAN CAPITAL 3 YEARS LATER, AND AFTER THAT, EXPLOSIVES WERE HURLED AT THE DOOR OF A HOUSE IN THE SAN BORJA NEIGHBORHOOD, WHERE U.S. MARINES RESIDED. ", "fileName": "training_page-1.txt", "totalNoOfFiles": 1, "rules": [{"propertyName": "hasName", "platformType": 2, "uniquePlatformId": "TPC_HumanTarget_1424970900693", "className": "HumanTarget"}], "partNumber": 1}

```

Figure 20 – Two-phase classifier Method: A sample request json object to extractor



University of Moratuwa, Sri Lanka.

```

{"fileNumber": 1, "extractedResults": [{"Capital": {"hasName": "Kabul"}, "Country": {"hasName": "Afghanistan"}, "fileNumber": 1, "fileName": "afghanistan.txt", "totalNoOfFiles": 0, "partNumber": 1}], "rules": [{"propertyName": "hasName", "platformType": 2, "uniquePlatformId": "TPC_HumanTarget_1424970900693", "className": "HumanTarget"}], "partNumber": 1}

```

www.lib.mrt.ac.lk

Figure 21 - Information Extractor Method: a sample response json object to extractor

Chapter 4

Results and Conclusion



University of Moratuwa, Sri Lanka.
Electronic Theses & Dissertations
www.lib.mrt.ac.lk

4.1 Implementation

Using the web services described in methodology, an application is developed to populate a given ontology. This provides the ability to extract information using both methods: information extractor method and two-phase classifier method.

In information extractor method, there are three input fields to accept the input from the user: Corpus location, information extractor metadata and the ontology. Both corpus location and information extractor fields can accept multiple files. Figure 22 depicts the image of the UI developed for this.

The screenshot shows the 'OBCIE Ontology Populator' web application. At the top, there is a navigation menu with links for 'Home', 'View Ontology', 'Information Extractor', 'Two Phase Classifier', and 'About'. Below the menu, the main heading reads 'Please select the Ontology, Corpus and the Information Extractors'. There are three input fields: 'Ontology:' with a 'Choose File' button and 'No file chosen' text; 'Corpus Location:' with a 'Choose Files' button and 'No file chosen' text; and 'Information Extractor:' with a 'Choose Files' button and 'No file chosen' text. An 'upload' button is positioned to the right of the input fields. Below the input fields, there is a watermark for the University of Moratuwa, Sri Lanka, with the text 'Electronic Theses & Dissertations' and the website 'www.lib.mrt.ac.lk'. A 'download file' link is also visible. At the bottom, a footer contains the text 'Copyright © OBCIE Distributed Pipeline. 2015'.

Figure 22 – UI developed to populate ontology using information extractor method

In two phase classifier method, there are six input fields to accept user input: Test corpus location, key files location for test corpus, training corpus location, key files location for training corpus, metadata files and the ontology. All input fields except the ontology field accepts more than one file. Figure 23 depicts the image of the web page developed for this.

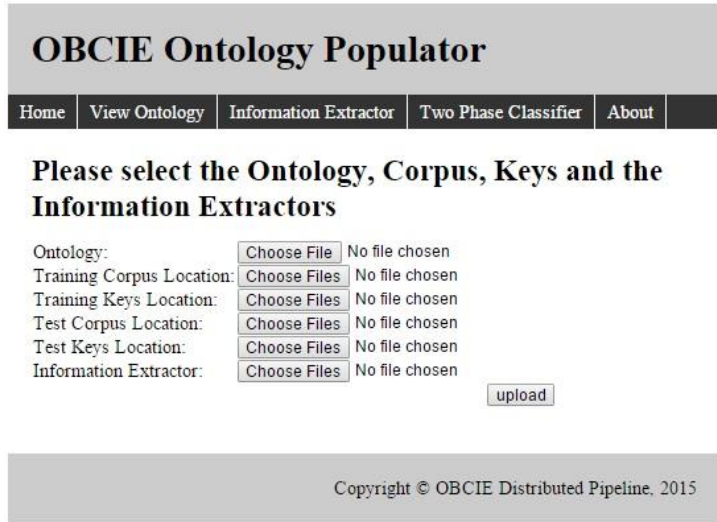


Figure 23 – UI developed to populate ontology using two-phase classifier method

In addition to the above two main functionality, this system also provides the user to view an ontology. This will list down the classes, properties, instances and the values for each property. In order to display the ontology, a third party library called “owl api” is used. This will accept the ontology file and render the information provided in this ontology file in HTML format. A sample output of a rendered ontology is given in figure 24.



University of Moratuwa, Sri Lanka
 Electronic Theses & Dissertations
www.lib.mrt.ac.lk

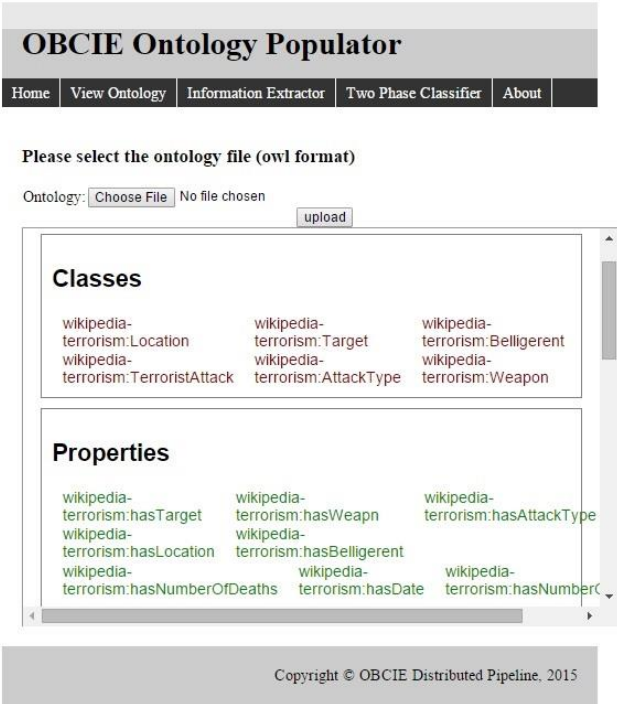


Figure 24 - UI developed to view a given ontology

4.2 Conclusion and Future Work

As the amount of available data grows, it becomes a challenge to extract the useful information easily from them. As a result, fields such as ontology based information extraction have emerged. However, most of the applications developed in this area are domain dependent. Therefore, if anyone to use information extraction methods in a new domain, then the amount of work that needs to be done is very large. By developing domain independent web services to extract information, the amount of time and effort required can be reduced greatly.

The web services developed here allows anyone to extract information from a set of documents according to a given set of rules. The rules provide the required domain dependent information and anyone can develop applications in any domain using them to give information to the users in a meaningful way.

The application developed here is one such instance. It accepts a set of documents and a set of rules and populates a given ontology using the extracted information. This populated ontology can be used in other applications developed for different purposes.

As future work, the application developed can provide the functionality for a user to design an ontology from scratch, rather than just displaying an already created ontology. Also the preprocessor can be improved to accept different type of documents, and return the content as simple text based documents. For example, when an HTML page is given as the input to the preprocessor, it should process it, remove the HTML tags and return the content as a simple text file. Also the message passing among the web services is done using json. As future work, the web services can be modified to support other data formats as well.

When extracting information using the two-phase classifier method, in order for it to give better output by training, it should have a large amount of data. Also this data is accumulated in the web service until they are used to train and analyze the information. This will take a long time to run and will lead to performance issues such as high usage of memory and other resources. Therefore as future work, this developed system can be improved to have good performance, while giving accurate information as the output.

REFERENCES

- [1] Amit Singhal, "Modern Information Retrieval: A Brief Overview," *BULLETIN OF THE IEEE COMPUTER SOCIETY TECHNICAL COMMITTEE ON DATA ENGINEERING*, 2001.
- [2] J Cowie and Y Wilks, "Information Extraction," , 1996.
- [3] Daya C. Wimalasuriya and Dejing Dou, "Ontology-based information extraction: An introduction and a survey of current approaches," *Journal of Information Science* 2010, vol. 36, no. 3, pp. 306-323, June 2010.
- [4] R Studer, V.R Benjamins, and D Fensel, "Knowledge Engineering: Principles and Methods," in *Data and Knowledge Engineering* 25, 1998, pp. 161-197.
- [5] Daya C. Wimalasuriya and Dejing Dou, "Components for Information Extraction: Ontology-Based Information Extractors and Generic Platforms," in *CIKM '10 Proceedings of the 19th ACM international conference on Information and knowledge management*, 2010, pp. 9-18.
- [6] M. Gruninger and M. Uschold, "Ontologies: Principles, Methods and Applications," *Knowledge Engineering Review*, vol. 11, no. 2, pp. 93-136, June 1996.
- [7] N. Guarino, D. Oberle, and S. Staab, "What Is an Ontology?," in *Handbook on Ontologies, International Handbooks on Information Systems.*: Springer-Verlag Berlin Heidelberg, 2009.
- [8] Noy, Natalya F and McGuinness, Deborah L.. (2001, March) *Ontology Development 101: A Guide to Creating Your First Ontology*. [Online]. Available: http://protege.stanford.edu/publications/ontology_development/ontology101-noy-mcguinness.html
- [9] Carlos Vicient Monllaó, "Ontology-based Information Extraction," Interuniversity Master in Artificial Intelligence (UPC-URV-UB), LSI Department, Technical University of Catalonia, Master of Science Thesis 2011.
- [10] Tim Berners-Lee, James Hendler, and Ora Lassila. (2001, May) The Semantic Web. Document.
- [11] Daya C. Wimalasuriya and Dejing Dou, "Using Multiple Ontologies in Information Extraction," in *18th ACM conference on Information and knowledge management*, 2009, pp. 235-244.

- [12] W3C. (2004, February) *Web Service Glossary, W3C Working Group Note 11*. [Online]. Available: <http://www.w3.org/TR/2004/NOTE-ws-gloss-20040211/#webservice>
- [13] W3C. (2004, February) *Web Service Architecture, W3C Working Group Note 11*. [Online]. Available: <http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/#gengag>
- [14] Einar Landre and Harald Wesenberg, "REST versus SOAP as Architectural Style for Web Services," in *ACM SIGPLAN International Conference on Object-Oriented Programming, Systems, Languages and Applications*, Montréal, Québec, Canada, 2007.
- [15] Hao He. (2003, September) What Is Service-Oriented Architecture. Document.
- [16] Xinyang Feng, Jianjing Shen, and Ying Fan, "REST : An Alternative to RPC for Web Services," in *ICFIN 2009, First International Conference on Future Information Networks.*, Beijing, 2009, pp. 7-10.
- [17] W3C. (2000, May) *Simple Object Access Protocol (SOAP) 1.1*, *W3C Note 08 May 2000*. [Online]. Available: <http://www.w3.org/TR/2000/NOTE-SOAP-20000508/>
- [18] P.A. Castillo, J.L. Bernier, M.G. Arenas, J.J. Merelo, and P. Garcia-Sanchez, "SOAP vs REST: Comparing a master-slave CA implementation," in *First International Workshop of Distributed Evolutionary Computation in Informatics*, New Orleans, 2011.
- [19] Roy Thomas Fielding, "Architectural Styles and the Design of Network-based Software Architectures," University of California, Irvine, Doctoral dissertation 2000.
- [20] M. zur Muehlen, J.V. Nickerson, and K.D. Swenson, "Developing Web Services Choreography Standards – The Case of REST vs. SOAP," *Decision Support Systems - Special issue: Web services and process management*, vol. 40, no. 1, pp. 9-29, July 2005.
- [21] G. Mulligan and D. Gracanin, "A Comparison of SOAP and REST Implementation of a Service Based Interaction Independence Middleware Framework," in *Winter Simulation Conference*, Austin, TX, 2009, pp. 1423-1432.
- [22] Hamish Cunningham, Valenti Tablan, Angus Roberts, and Kalina Bontcheva, "Getting More Out of Biomedical Documents with GATE's Full Lifecycle Open Source Text Analytics," *PLOS Computational Biology*, vol. 9, no. 2, February 2013.
- [23] René Witte, Ninus Khamis, and Juergen Rilling, "Flexible Ontology Population from Text: The OwlExporter," in *Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta, 2010.

- [24] Paul Buitelaar, Philipp Cimiano, Stefania Racioppa, and Melanie Siegel, "Ontology-based Information Extraction with SOBA," in *LREC*, Genoa, Italy, 2006.
- [25] (2015) *GATE General Architecture for Text Engineering, Chapter 8, JAPE:Regular Expressions over Annotations.* [Online]. Available: <https://gate.ac.uk/sale/tao/splitch8.html>
- [26] Hamish Cunningham, Kevin Humphreys, Robert Gaizauskas, and Yorick Wilks, "GATE - A General Architecture for Text Engineering," in *Proceedings of the Fifth Conference on Applied Natural Language Processing: Descriptions of System Demonstrations and Videos*, Washington, DC, USA, 1997, pp. 29--30.
- [27] Ian H Witten et al., "Weka: Practical Machine Learning Tools and Techniques with Java Implementations," , 1999.
- [28] A. K. McCallum. (2002) *MALLET: A Machine Learning for Language Toolkit.* [Online]. Available: <http://mallet.cs.umass.edu>
- [29] (2015) *Java API for RESTful Web Services.* [Online]. Available: http://en.wikipedia.org/wiki/Java_API_for_RESTful_Web_Services
- [30] Java.net. (2014) *Java API for RESTful Services (JAX-RS).* [Online]. Available: <https://jax-rs-spec.java.net/>
- [31] (2010, April) *Apache Wink 1.4.0 user guide.* [Online]. Available: https://wink.apache.org/1.4.0/Apache_Wink_User_Guide.pdf
- [32] Vishnu Vettrival. (2010, April) *RESTful Web services with Apache Wink, Part 3: Apache Wink and the REST - A comparison of Apache Wink and other open source JAX-RS implementations.* [Online]. Available: <http://www.ibm.com/developerworks/library/wa-apachewink3/>
- [33] (2015) *Jersey: RESTful web services in java.* [Online]. Available: <https://jersey.java.net/>
- [34] *RestEasy:A jboss project.* [Online]. Available: <http://resteasy.jboss.org/>
- [35] *Restlet Framework.* [Online]. Available: <http://restlet.com/products/restlet-framework/>
- [36] Kalina Bontcheva, Valentin Tablan, Diana Maynard, and Hamish Cunningham, "Evolving GATE to Meet New Challenges in Language Engineering," *Natural Language Engineering*, vol. 10, no. 3/4, pp. 349-373, 2004.

APPENDIX A: Sample Metadata Files – Information Extractor

Extract Currency Information

```
<?xml version="1.0"?>
<InformationExtractor
xmlns="http://aimlab.cs.uoregon.edu/obie/StandardXMLSchema/1.0/">
<Language>EN</Language>
<IETechnique>http://aimlab.cs.uoregon.edu/obie/IETechniques/LinguisticExtractionR
ules.xml</IETechnique>
<Concept>Currency</Concept>
<ConceptType>Class</ConceptType>
<IdentifierName>hasName</IdentifierName>
<Feature>rule:input:Lookup</Feature>
<Feature>rule:
    ({Lookup.minorType == post_amount})
    : curr -->
    {
        gate.AnnotationSet curSet = (gate.AnnotationSet)bindings.get("curr");
        FeatureMap features = Factory.newFeatureMap();
        outputAS.add(curSet.firstNode(), curSet.lastNode(),"CurrencyX", features );
    }
</Feature>
</InformationExtractor>
```



University of Moratuwa, Sri Lanka.
Electronic Theses & Dissertations
www.lib.mrt.ac.lk

Extract the names of Capitals mentioned in the corpus

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

<InformationExtractor
xmlns="http://aimlab.cs.uoregon.edu/obie/StandardXMLSchema/1.0/">

<Language>EN</Language>

<IETechnique>http://aimlab.cs.uoregon.edu/obie/IETechniques/LinguisticExtractionR
ules.xml</IETechnique>

<Concept>Capital</Concept>

<ConceptType>Class</ConceptType>

<IdentifierName>hasName</IdentifierName>

<Feature>rule-input:Lookup</Feature>

<Feature>rule:
({Lookup.minorType == city}).
: loc -->
{
    gate.AnnotationSet curSet = (gate.AnnotationSet)bindings.get("loc");
    FeatureMap features = Factory.newFeatureMap();
    outputAS.add(curSet.firstNode(), curSet.lastNode(), "CapitalX", features );
}
</Feature>

</InformationExtractor>
```



University of Moratuwa, Sri Lanka.
Electronic Theses & Dissertations
www.lib.mrt.ac.lk

APPENDIX B: Sample Metadata File – Two-Phase Classifier

To identify the Perpetrators given in the corpus

```
<?xml version="1.0" encoding="UTF-8"?>
<InformationExtractor
xmlns="http://aimlab.cs.uoregon.edu/obie/StandardXMLSchema/1.0/">
<Language>EN</Language>
<IETechnique>http://aimlab.cs.uoregon.edu/obie/IETechniques/TwoStepClassification.xml</IETechnique>
<Concept>Perpetrator</Concept>
<ConceptType>Class</ConceptType>
<IdentifierName>hasName</IdentifierName>
<Feature>word:attack</Feature>
<Feature>word:kidnap</Feature>
<Feature>word:kill</Feature>
<Feature>word:threaten</Feature>
<Feature>word:burn</Feature>
<Feature>word:civilian</Feature>
<Feature>word:peasant</Feature>
<Feature>word:leader</Feature>
<Feature>word:politician</Feature>
<Feature>word:officer</Feature>
<Feature>word:military</Feature>
<Feature>word:innocent</Feature>
<Feature>word:terrorist</Feature>
<Feature>word:soldier</Feature>
<Feature>word:guerrilla</Feature>
<Feature>word:armedforces</Feature>
<Feature>word:armed</Feature>
<Feature>word:individual</Feature>
<Feature>word:suspected</Feature>
<Feature>word:commando</Feature>
```



University of Moratuwa, Sri Lanka.
Electronic Theses & Dissertations
www.lib.mrt.ac.lk

<Feature>word:officers</Feature>
<Feature>word:army</Feature>
<Feature>word:murderer</Feature>
<Feature>word:leftist</Feature>
<Feature>word:rightwing</Feature>
<Feature>word:kidnapper</Feature>
<Feature>word:navy</Feature>
<Feature>word:airforce</Feature>
<Feature>word:revolutionary</Feature>
<Feature>word:movement</Feature>
<Feature>word:spokesman</Feature>
<Feature>word:spokeswoman</Feature>
<Feature>word:clandestine</Feature>
<Feature>word:arson</Feature>
<Feature>word:shoot</Feature>
<Feature>word:dynamite</Feature>
<Feature>word:tnt</Feature>
<Feature>word:explosive</Feature>
<Feature>word:grenade</Feature>
<Feature>word:bullet</Feature>
<Feature>word:axe</Feature>
<Feature>word:fuse</Feature>
<Feature>synset:attack</Feature>
<Feature>synset:kidnap</Feature>
<Feature>synset:kill</Feature>
<Feature>synset:threaten</Feature>
<Feature>synset:burn</Feature>
<Feature>synset:civilian</Feature>
<Feature>synset:peasant</Feature>
<Feature>synset:leader</Feature>
<Feature>synset:politician</Feature>
<Feature>synset:officer</Feature>



<Feature>synset:military</Feature>
<Feature>synset:innocent</Feature>
<Feature>synset:shoot</Feature>
<Feature>gazetteer:firstname</Feature>
<Feature>gazetteer:lastname</Feature>
<Feature>gazetteer:military_rank</Feature>
<Feature>gazetteer:job_title</Feature>
<Feature>gazetteer:terrorist_organization</Feature>
<Feature>gazetteer:instrument_type</Feature>
</InformationExtractor>



University of Moratuwa, Sri Lanka.
Electronic Theses & Dissertations
www.lib.mrt.ac.lk