Enhancing
Information-Centric Web
Services Through Query
Expansion

Cher Min Gabriel Foo

This report is submitted as partial fulfilment
of the requirements for the Honours Programme of the
School of Computer Science and Software Engineering,
The University of Western Australia,
2005
Abstract

Information-centric Web Services are Web Services retrieving information rather than performing actions. To be able to retrieve the desired information, highly precise queries are needed because of the lack of semantics in normal Web Services. In particular, Web Services are intended for program consumption rather than human consumption. The lack of semantics exerbate the situation of low precision in information retrieval or database query through Web Services.

Humans are good at coming up with alternative query strings but programs, unless deliberately programmed to do so, it is almost impossible for them to try refining the query string automatically. So Web Services that support queries should be improved to return not just information containing the query keywords but also semantically related results. Although semantically enhanced Web Services are provided through the Semantic Web, drastic changes to existing approaches are often resisted by businesses.

This research looks into making use of available database schemas and domain terminologies to enhance the existing service-oriented Information Retrieval tasks to return semantically relevant information.

Thus, we will apply Query Expansion techniques to enhance the performance of information-centric Web Services. The use of ontology distance, incorporating lexical dictionary such as WordNet, and the use of domain terminology are integrated to achieve semantically enriched Query Expansion. A prototypical system, using ATDW (Australian Tourism Data Warehouse) as the domain terminology, is created to test out the Query Expansion ideas proposed.

**Keywords:** Web Services, Semantic Web, Ontology, Information Retrieval

**CR Categories:** H.3.3, I.2.7
Acknowledgements

Thank you God for everything.

I would like to thank Dr. Wei Liu for being patient and understanding throughout the research and writing of this dissertation. Without her assistance, this dissertation will not have been completed. I would like to thank my family and Siew Leng for their love and support throughout the year.
Contents

Abstract ii

Acknowledgements iii

1 Introduction 1

2 Information Retrieval 4
   2.1 Introduction .............................................. 4
   2.2 Search Engines .............................................. 4
   2.3 Retrieval and Storage Techniques .................... 5
      2.3.1 Web Crawlers ....................................... 6
      2.3.2 Web Indexing ....................................... 6
   2.4 Ordering Techniques used by search engines .......... 7
      2.4.1 Page Rank ........................................... 8
      2.4.2 Term Frequency / Inverse Document Frequency .... 8
      2.4.3 Metadata ............................................. 8
   2.5 Limitation of keyword-based queries .................. 9
   2.6 Query Expansion ......................................... 10
      2.6.1 Domain Taxonomy ................................... 10
      2.6.2 Ontology ............................................. 11
      2.6.3 Lexical Dictionary: WordNet ....................... 12
      2.6.4 Thesaurus ............................................. 12
      2.6.5 Summary .............................................. 13
   2.7 Discussion ................................................. 13

3 Web Services 14
   3.1 Introduction .............................................. 14
3.2 Web Services: An Overview ........................................ 14
  3.2.1 Communication: SOAP ........................................ 15
  3.2.2 Description: WSDL ........................................ 16
  3.2.3 Discovery: UDDI ........................................ 16
  3.2.4 Technical Description: tModel .............................. 17
3.3 Types of Web Services ........................................ 17
3.4 Limitations of Traditional Information-centric Web Services ... 18
3.5 The Semantic Web: an Overview ................................. 18
3.6 Overcoming Limitation Through Semantics ..................... 18
  3.6.1 OWL-S: Semantic Markup for Web Services ............... 19
  3.6.2 LARKS: Agent Capability Description Language ......... 19
  3.6.3 The Lack of Service Ontology in UDDI .................... 20
3.7 Query Expansion of information-centric Web Services .......... 21
  3.7.1 Ontology Distance ....................................... 21
  3.7.2 Vector Space Approach .................................. 22
  3.7.3 Edit Distance ........................................ 23
  3.7.4 Term Frequency/Inverse Document Frequency approach .... 24
3.8 Summary ....................................................... 24

4 The Implementation ............................................... 26
  4.1 Overview: Design Purpose .................................... 26
  4.2 External Schemas and Repository ............................. 26
  4.3 Architecture of Framework ................................... 27
    4.3.1 Data Repository ....................................... 28
    4.3.2 Semantic Level: Ontology Distance ...................... 28
    4.3.3 Fine Tuning: Vector Space Similarity ................. 29
    4.3.4 Popularity search: Search Engine ................. 30
    4.3.5 Additional Components ................................. 30
  4.4 Summary ....................................................... 31
List of Tables

1.1 Search Engines and Web Services ........................................ 3
2.1 Example of a hash table ......................................................... 7
5.1 Query Expansion of query string inn ..................................... 35
5.2 Query Expansion of query string inn ..................................... 36
List of Figures

2.1 A simple ontology ............................................. 11
3.1 The Web Service Interaction pattern ......................... 15
3.2 Vector Space representation .................................. 22
4.1 A Diagram of the System Architecture ....................... 27
4.2 Ontology Distance .............................................. 28
4.3 Vector Space ................................................... 29
5.1 Query on ATDW without Expansion ......................... 34
5.2 Result of Query Expansion on inn ........................... 35
CHAPTER 1

Introduction

The World Wide Web (Web) has become one of the most important technologies used in the world today. Whether we use it to discover information or to communicate information with another party, the Web has tremendously changed the way we use computers and how we interact with one another.

In today’s context of using computers, one would immediately think of the Web as a resource of networked documents. Over the past few years, businesses have used the Web as a way of disseminating information and facilitating business to customer interactions. An emerging trend in this area is the development of Web Services. Web Services are self-describing, self-contained modular units of application logic that provides to other applications through an Internet connection [9]. The aim of a service-oriented Web is to provide a systematic and extensible framework for business process integration. It is built on existing Web protocols and open XML standards [19]. The ultimate aim of Web Services is to turn the Web from document-centric toward process-centric.

There are largely two types of Web Services available. The first is action-based Web Services, where Web Services are created as end-points to allow interaction with programs to execute certain actions. Such Web Services usually provide limited descriptive information because its main task is to allow execution of actions. The other types of Web Services are information-centric Web Services. Information-centric Web Services are Web Services retrieving information rather than performing actions. Information-centric Web Services revolves around information retrieval type of tasks that involve queries. Queries for such Web Services are typically limited to keyword-based form of matching.

As Web Services are programs that are consumed by other programs, Web Services do not have the ability to refine its query for better retrieved results. Unlike a human, who is very good at disambiguating confusing words and coming up with alternative query strings other than a query’s surface representation, a computer program does not have any automatic way to alter the querying task. Two major solutions have been proposed to overcome the shortcomings of the
existing Web Services framework.

- The Semantic Web Initiative - a new framework, where Web content is annotated with, thus allowing more accurate results to be obtained.
- Leveraging on existing framework - apply additional techniques to ascribe semantics onto the existing Web Services framework.

The first approach, being the Semantic Web, is an ideal solution to expanding Web Service’s matching abilities using domain-specific ontologies. These ontologies will be developed and used to annotate information provided in the Semantic Web. These ontologies provide additional information about the Web Service and create a structure, where other relevant and similar Web Services are linked. Queries are no longer isolated to a Web Service description in WSDL as other information is readily available for comparison and matching. Although this solution seems viable, it has several implications that need to be addressed. Firstly, many businesses have adopted the existing Web Service framework, and in order to change, a large amount of work and cost may be involved to adopt the new framework. Secondly, the new framework proposed so far, is experimental and has only been tested on a relatively small scale. Thus, the adoption of a new framework is still resisted by the mainstream businesses whose top priorities are usually cost related.

The second approach to expand the Web Service’s information searching is through enriching Web Services with semantics. In order to enrich an otherwise purely syntactical query results, we would need techniques that can provides us with additional semantics on queries. Words and description should no longer just be taken at the face value. In order to achieve that, techniques found in the Information Retrieval literatures such as automatic annotation and query expansion are needed. Automated annotation include extra information associated with a particular point in a document or other piece of information [21]. This is usually achieved through the use of a natural language toolkit, where the toolkit will automatically annotate words that have additional meaning or information available to it. Query Expansion, on the other hand, expands queries by using techniques that allow us to retrieve additional semantics without needing to change the existing framework. It enhances the results returned from the queries through discovering similarities and relevance between similar words and concepts. The use of annotations, however, requires a separate storage for indexing of these annotations. Therefore our focus will be on Query Expansion as it can be applied without the need to store additional items.

Before the use of Web Services became prominent, Query Expansion has already been in use to enhance queries from keyword-based search engines. In fact,
Table 1.1 shows that both search engines and information-centric Web Services have similar inputs which is a query. The difference is search engines queries usually return documents as a result while information-centric Web Services queries return semantically relevant database or information items.

<table>
<thead>
<tr>
<th></th>
<th>Search Engines</th>
<th>Information-centric Web Services</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>Query</td>
<td>Query</td>
</tr>
<tr>
<td>Output</td>
<td>Document</td>
<td>Semantically relevant Database Items</td>
</tr>
</tbody>
</table>

Table 1.1: Search Engines and Web Services

Thus, the aim of this research is to enhance information-centric Web Services through the use of Query Expansion. In this project, we will first study basic techniques used in information retrieval and show how Query Expansion on search engines improves results returned. Then we will try to integrate query expansion into traditional keyword based query of information-centric Web Services.

In Chapter 2, we introduce basic Information Retrieval ideas. Techniques are discussed to give an overview of how they are applied in the area of Query Expansion in document-based search engines and illustrate the similarities with Query Expansion of information-centric Web Services.

In Chapter 3, we introduce the technologies behind Web Services. Both existing and Semantic Web Services technologies are discussed to provide a comprehensive overview of the differences of the technologies. We propose several Query expansion techniques to be integrated into discovery of Web Services.

In Chapter 4, we discuss the implementation of experiments that incorporate Query Expansion techniques discussed in prior chapters to show that query expansion does provide a semantically richer result for information-centric types of Web Services. A framework is proposed and details of its structure and implementation are discussed.

In Chapter 5, results obtained from the experiments are evaluated and explained to show that query expansion does create additional semantics in an otherwise keyword-based retrieval of queries from information-centric Web Services. A comparison is made between the initial query and the expanded query.

In Chapter 6, we conclude our findings and discuss some of the future works and direction of the research. Some open problems of the research are also discussed.
CHAPTER 2

Information Retrieval

2.1 Introduction

Information retrieval (IR) is the art and science of searching for information in documents, searching for documents themselves, searching for metadata which describes documents, or searching within databases, whether relational stand alone databases or hypertext networked databases on the Internet or intranets, for text, sound, images or data [10]. There is a common confusion, however, between data retrieval, document retrieval, information retrieval, and text retrieval, and each of these have their own bodies of literature, theory, praxis and technologies [10]. However, the use of the term Information Retrieval can usually be attributed to retrieval of information.

Dominant areas of research include document classification, text-retrieval, designing of information and the organization of information.

2.2 Search Engines

One of the major areas of research in IR is the research of search engines. Indeed apart from the existence of hyperlinks which establish connections between documents on the Web, the other main valuable and indispensable tools on the Web are search engines. Without search engines, the Web would not have been such a huge success as it is now. Even so, search engines do have several problems that are associated with their keyword based document indexing. Some of the major ones include:

- **Low or no recall** - Important and relevant pages are not retrieved, or no answer from query
- **High recall, low precision** - Main relevant pages are retrieved along
with many mildly relevant or irrelevant documents. This is usually caused by queries that are general and not specific enough to retrieve.

- **Results highly sensitive to vocabulary** - Documents has relevant keywords but use different terminology from the original query will not be retrieved. This is usually resolved by retrieving synonyms of the query.

- **Results are single Web pages** - If original query exist over several various documents, several sub queries are needed to collect the relevant documents, and put together manually.

Ideally, results returned by search engines should have high recall and precision. To achieve these, search engines have adopted a variety of methods and techniques from Information Retrieval area. A typical search engine works in the following manner:

1. **Crawling**
2. **Indexing**
3. **Searching**

We will give a discussion on crawling techniques used by search engines for retrieval, and how it indexes the documents that have been retrieved and finally how search engines search and orders its results.

### 2.3 Retrieval and Storage Techniques

For a search engine to function, it needs both good retrieval and storage techniques to extract data from web documents. Retrieval techniques revolve around the search engines attempting to retrieve valuable keywords that appear on the Web document through software. Such techniques are usually automated as the number of Web documents is large and it is impossible for a human to manually extract the keywords.

Storage techniques, on the other hand, are techniques that are employed by search engines to store keywords obtained by the retrieval techniques used. As such, retrieval and storage techniques go hand in hand in the successful use of search engines. In the following sections, we attempt to give a comprehensive overview of these techniques used by search engines. This will give us an important understanding of how search engines work.
2.3.1 Web Crawlers

A Web crawler (also known as a Web spider or ant) is a program or software agent which browses the World Wide Web in a methodical, automated manner [16]. Web crawling caches copies of all the visited pages so that search engines can index the pages for fast searches as well as provide statistical information to the user. Statistics can include number of pages that are relevant to the query, the number of times a Web page was accessed, etc. There are two major types of crawling techniques employed by search engines.

- **Deep Crawling** - Uses depth first search algorithms for retrieval of information.
- **Fresh Crawling** - Uses breath first search algorithms for information retrieval.

Deep crawling, using depth first search algorithms digs deep within individual sites in order to discover all the Web pages within this Web site for indexing. Such form of Web crawling enables search engines to be able to retrieve thorough information for querying later. Fresh crawling, on the other hand, usually employ breath first search algorithms to cache Web pages that are visible on the top level of a Web site. This allows the search engines to discover the Web pages that have been updated from the last Web crawl. In both uses of crawling, retrieval of information about documents is performed automatically by software programs without the need of human interaction.

2.3.2 Web Indexing

Web indexing (or “Internet indexing”) includes back-of-book-style indices to individual websites or an Intranet, and the creation of keyword metadata to provide a more useful vocabulary for Internet or onsite search engines. With the increase in the number of periodicals that have articles online, Web indexing is also becoming important for periodical websites [10]. Categorising of documents and pages retrieved from the Web ensures that queries made by search engines later are quick for retrieval and comparison.

Indexing of documents and pages from the Web varies and include extraction of important keywords from metatags, header and titles to extraction of the entire document. Difference in indexing differs due to the difference of focus or importance held by search engines and their programmers. Regardless of the indexing criteria, indexing can be done through the use of a common form of
data structure, known as the hash table, which is a data structure that divides all elements into (preferably) equal-sized categories, or buckets, to allow quick access to the elements.

<table>
<thead>
<tr>
<th>uid</th>
<th>Web Page</th>
<th>url</th>
<th>keyword1</th>
<th>keyword2</th>
<th>keyword3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>index.html</td>
<td>./230.124</td>
<td>Java</td>
<td>Programming</td>
<td>...</td>
</tr>
<tr>
<td>2</td>
<td>index.html</td>
<td>./230.104</td>
<td>Software</td>
<td>Engineering</td>
<td>...</td>
</tr>
</tbody>
</table>

Table 2.1: Example of a hash table

Table 2.1 shows an example of a possible hash table that search engines uses to index web pages. A unique identifier is used to separate all indexes while each entry has associated keywords that have been extracted from the web pages itself. Using the hash table, a search engine can find relevant web pages against the query made by looking at its index database. Without a proper index of retrieved Web documents, search engines would have a hard time processing queries made to it and even provide results back.

2.4 Ordering Techniques used by search engines

Despite the use of Web indexing to categorise Web documents retrieved by search engines, query results return by search engines are usually not absolute and in most of the time, more than one relevant documents are returned. Without a proper technique to order the results returned, search engines would not be able to obtain the necessary high precision and recall expected of a search engine. As such, ordering techniques are used by search engines to order results retrieved from queries made to search engines, in order to provide high precision results back to the users.

Some search engines employ a single method to order the results while others employ a variety of methods to return results. Regardless of the number of methods employed by any single search engines, the significance of the ordering techniques shows that no information retrieval types of task is complete without techniques to order the results obtained.

As a majority of search engines uses proprietary algorithms and closed databases, we discuss techniques used by search engines that are known for ordering of Web documents.
2.4.1 Page Rank

Page Rank is a family of algorithms for assigning numerical weightings to hyper-linked documents (or web pages) indexed by a search engine [10]. Google™ is one of the most popular Internet search engines available now. One major feature that it has is the Page Rank algorithm, where the set of most relevant Web documents are arranged according to the number of links which points to it [29]. But Google looks at more than the sheer volume of votes, or links a page receives; it also analyzes the page that casts the vote. Votes cast by pages that are themselves “important” weigh more heavily and help to make other pages “important”.

The advantage of Page Rank over other algorithms is that document on the Web is ranked based on the number of links towards it, rather than based on the occurrence of content. This gives us assurances that pages that have multiple instances of a query may not always appear highest on the list. However, a disadvantage of using Page Rank is that it favors older pages, because a new page, even a very good one, will not have many links unless it is part of an existing site.

2.4.2 Term Frequency / Inverse Document Frequency

Term Frequency / Inverse Document Frequency (TF/IDF) is a popular weighting algorithms used in Information Retrieval. The term frequency in the given document gives a measure of the importance of the term within the particular document. The document frequency is a measure of the general importance of the term (it is the log of the number of all documents divided by the number of documents containing the term) [13].

TD/IDF is used for determining the importance of a word in a document. The importance of a word is determined by its occurrences within a document as well as how common it is within the collection of documents it is in. TF/IDF is commonly used in search engines because a direct formulation can be used to determine the result where the frequency of a word within a document and collection can be determined.

2.4.3 Metadata

A good method of linking weakly structured information is through the use of the metadata. Metadata defined as machine understandable information for the web [26]. It is a common approach to the problem of information heterogeneity.
In the context of documents on the Web, a user has to find the information needed, before it can be used. However, the information is usually presented in different kinds of data format and structures. Metadata can be used to tackle this problem:

- **Search** - By providing important keywords and topic areas, metadata can be used to identify information sources on the Web without having to search every single Web page. As the Internet grows rapidly, this ability is very important.

- **Access** - Metadata can reduce the effort to process all available information in the document through by providing users with important technical properties of a source.

- **Interpretation** - Metadata helps to reduce misinterpretation of content and helps both human users and intelligent systems to understand the content of an information source.

In the case of search engines, manually-created metadata adds value because it ensures consistency. If one Web page about a topic contains a word or phrase, then all Web pages about that topic should contain that same word. It also ensures variety, so that if one topic has two names, each of these names will be used. For example, an article about Sports Utility Vehicles would also be given the metadata keywords 4 wheel drives, 4WDs and four wheel drives, as this is how they are known in some countries.

### 2.5 Limitation of keyword-based queries

Having discussed about search engines and the techniques used for retrieval, storage and ordering, we realised these techniques are keyword-based and there are no semantics involved in using these techniques. Although in most cases, we are able to retrieve relevant documents using keywords. However, limitation with using keywords is that it relies on exact or semi-exact matches with literal words in the content of documents. This means if a keyword has multiple meanings, the query will probably return results that are relevant in keywords, rather than its intended meaning. This poses huge problems, when attempting to retrieve documents where results are returned. Even among the results returned, relevance of results varies largely.

As search engines are software based, results obtained from the query are largely based on the algorithms and techniques used and in most cases, in order
for users to get better results, users have to manually amend their query in order to get better results. Unless deliberately programmed to do so, it is almost impossible for search engines to try refining the query string automatically. Ideally, refinement of results should not be just information containing the query keywords but also semantically related results. Thus, to overcome, or rather, to improve the results return, an automatic expansion of the keywords is needed.

2.6 Query Expansion

Keyword, on its own, does not represent any semantics nor does it have any meaning other than its surface representation. In addition, Query formulation, especially the selection of terms, is crucial for search success. The limitation of keyword-based queries has prompted many researches in the area of Information Retrieval. One such method is query expansion. Query expansion is the process of adding relevant terms to a user’s weighted search. The intent is to improve precision and/or recall of the query [14].

As there are usually little to no semantics obtainable from the surface representation of a word, additional tools are used for Query Expansion.

Through Query Expansion of keywords, we can hopefully apply and support semantics to a word by expanding the underlying meaning of the word. Many search engines employ Query Expansion to increase the relevance of results returned to user. Currently, there are many methods proposed or being researched to expand queries of keywords. The methods discussed are not exhaustive but rather, some of the more important ones being used for query expansion.

2.6.1 Domain Taxonomy

A domain taxonomy is a hierarchical categorization or classification of entities within a domain [3]. It can also be thought of as a clustering of entities based on common ontological characteristics. It allows us to represent entities and concepts of a domain in a manner where we can identify related relationships of a particular entity with respect to another. By using the domain taxonomy, we can expand queries via related concepts on the domain, ensuring that these expansions are related to the original query, thus ensuring relevance in the queried results. An example of domain taxonomy is the Australian Tourism Data Warehouse (ATDW) which provides a central distribution and storage facility for tourism product and destination information for all Australian States and Territories.
A domain taxonomy can be defined via several methods such as through data mining, textual analysis, and statistical techniques. With well defined domain taxonomy, we can classify different words together via common ontological characteristics. A good example would be domain taxonomy of animals. Assuming we have three animals that are tiger, lion and parrot. Based on the characteristics that both tiger and lion have four legs and a parrot only has two legs, we can see that tiger and lion are more closely related terms than parrot.

2.6.2 Ontology

An ontology is the attempt to formulate an exhaustive and rigorous conceptual schema within a given domain, typically a hierarchical data structure containing all the relevant entities and their relationships and rules (theorems, regulations) within that domain [10]. Unlike domain taxonomy, which is a clustering of entities based on domain characteristics, ontology is a full specification of a domain.

Ontologies provide a convenient mechanism for expansion of words where they can be consistently compared in a semantic fashion, by providing concepts and relationships. It can be represented in a hierarchical data structure as shown in Figure 2.1, an example ontology of subsumption relationships. Constructing ontology is always considered as the most challenging and time-consuming task in knowledge engineering.

Figure 2.1 shows a simple ontology that represent part of an animal ontology where each animal name represents a class of similar animals.

```
    Animal
     /|
    / |\n   /  |  \n  Fish  Mammal
     /|
    / |\n   /  |  \n  Shark  Eel  Leopard  Lion
```

Figure 2.1: A simple ontology

Thus through an ontology, we can determine the closeness of relationships
between words or concepts. This is useful in expanding queries because through a similar concept, we can identify other keywords that we can use to expand the original keyword. However, one of the major drawbacks of using this method to expand queries is that defining ontology is difficult and even more so for a huge database of words. In addition, ontology needs to be well defined in order to be able to leverage on the ontology distance method.

2.6.3 Lexical Dictionary: WordNet

Besides applying domain taxonomy and ontology, which can be applied by adding them onto existing search spaces, we can expand queries through third-party tools to discover additional semantics over a particular word or concept.

WordNet\[^1\] [22] is an online lexical reference system whose design is inspired by current psycholinguistic theories of human lexical memory. English nouns, verbs, adjectives and adverbs are organized into synonym sets, each representing one underlying lexical concept. Different relations link the synonym sets.

Through WordNet, we can identify words that are related to the query either directly or indirectly in order to enhancing the query to make it more specific. For example, the query *fly*, it can exist as either a noun, which can represent an insect, or a verb where it represents an action. By using Wordnet, we can expand the query *fly* by adding the word *insect* if we want to query it as a noun or adding the the word *pilot* or *wing* if we want to expand the query as a verb.

2.6.4 Thesaurus

A thesaurus is defined as a taxonomy that also includes associated and related terms. It is the most complex type of controlled vocabulary, and is sometimes used to standardize an organization’s terminology and subsequently inform both navigation and search systems [7]. A thesaurus can assist expansion of queries by presenting by provide searchers with information about the equivalence and hierarchical relations of query terms.

An example is *ample*, which when queried by a thesaurus will return a list of words that include *abounding, abundant, big, bounteous, bountiful, broad*. In this example if the query *ample* through a search engine does not return relevant results, we can then attempt to use the associated words returned by the thesaurus to find relevant results. Unlike Wordnet, a lexical dictionary, a thesaurus

\[^1\]http://wordnet.princeton.edu/
restricts itself to synonyms and antonyms while Wordnet includes hypernyms, hyponyms, holonym and meronym as related words.

Thesaurus is not isolated to general use but has also seen its uses in domain specific areas. Domain specific thesaurus can be used to limit the vocabularies or to apply consistency of terms used around a query. Thus a query search of mouse in a computer hardware domain will not result in results retuning information about the animal mouse but could return information relating to a “human interface device”.

2.6.5 Summary

Having discussed some of the techniques and technology behind query expansion, we have observed that the query expansion of the keywords is done through determining relationships between the keywords and related words. By attempting to lengthen the query with additional words that are related to the original keyword, ideally, we will be able to return more relevant documents than before. Another observation made is that the techniques are applied on top of the existing search engine framework. Thus it is possible to use query expansion on a variety of applications, not just for search engines.

2.7 Discussion

We have discussed the ideas behind the query expansion in search engines and how they are able to produce semantically richer results as opposed to purely keyword-based searching. The techniques and methods applied in query expansion are largely generic and can be applied not just on search engine queries but in a variety of applications that has a need for querying information. One such similarity with search engine queries is the querying of information-centric Web Services. Web Services too, have had limitations in the ability to query. Ideally, integration of Query Expansion onto Web Services would return results which are semantically richer.

In the next chapter, we give an overview of Web Services and some of the researches done in creating more semantics for Web Services. In addition, we attempt to show some Query Expansion techniques that can be integrated into Web Services.
CHAPTER 3

Web Services

3.1 Introduction

Business have interacted using ad hoc approaches that take advantage of the basic Internet infrastructure over the past few years. However, Web Services is emerging as an approach to provide a systematic and extensible framework for application-to-application interaction that is built on top of existing Web protocols and based on current open XML standards. Web Services are self-describing, self-contained applications that perform a specific business task and conform to a particular technical format [9, 18].

Traditionally, the Web is geared towards delivering information visually, where the contents on the Web are usually human readable and not computer intelligible. A new idea, through the Semantic Web, is enabling both computers and humans to bring understanding between both parties. In addition, with the introduction of Web Services, the Web is now geared towards exchanging information as well as integrating business processes between applications on the Web. As information on the Web becomes larger and more complex, it is difficult to identify every service on the Web. Thus, the ability to incorporate both ideas will lead us to a more intelligible querying of Web Services.

3.2 Web Services: An Overview

Web services essentially aim to simplify the process of say, for example, that of purchasing a vacation package using an online travel agent. To locate the best prices on airline tickets, hotels, and rental cars, the agency will have to poll multiple companies, each of which likely uses different, incompatible, applications for pricing and reservations. Web services define a standardized mechanism to describe, locate and communicate with online application as essentially each application becomes an accessible Web service component that is described using a
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 [15].

A typical Web service architecture consists of three main components: service registry, service provider and service requester.

The service registry is usually provided by a directory known as the Universal Description Discovery and Integration (UDDI) registry. The UDDI records information about deployed Web services in a similar fashion as a yellow pages telephone directory.

The service providers are the Web services that publish about its service to the service registry so that its services can be made known on the Web.

The service requester is a querying service that finds the service that it needs by using a lookup against the service registry, and then binds to the service provider. In this way, the provider and the requester can interact in a meaningful way.

So in an essence, we would require protocols for communication, description as well as discovery in order to make up a Web service.

3.2.1 Communication: SOAP

Given the Web’s intrinsically distributed and heterogeneous nature, communication mechanisms must be platform-independent, international, secure and as
lightweight as possible. The Simple Object Access Protocol (SOAP) fulfills this task for Web services by providing itself as a protocol for messaging and remote procedure calls (RPCs) [24].

As it is written in XML, it supports platform independence and internationalization. In addition, SOAP, works on existing transport protocols, such as HTTP, SMTP and MQSeries [19]. At its core, SOAP specification defines a model that dictates how recipients should process SOAP messages. The messages model also includes actors, which indicate who should process the message. A message can identify actors that indicate a series of intermediaries that process the message parts meant for them and pass on the rest.

3.2.2 Description: WSDL

For Web services, SOAP offers basic communication, but it does not tell us what messages must be exchanged to successfully interact with another service. That role is filled by the Web Service Description Language (WSDL). WSDL is an XML formatted language that describes Web services as collections of communication end points that can exchange certain messages [17].

In WSDL, endpoints and messages are separated from their concrete network deployment which allows the reuse of abstract definitions. The concrete protocol and data format specifications for a particular port type in WSDL constitute a reusable binding. A port is defined by associating a network address with a reusable binding, and a collection of ports define a service.

3.2.3 Discovery: UDDI

The UDDI specifications offer users a unified and systematic way to find service providers through a centralized registry of services that is roughly equivalent to an automated online “phone directory” of Web services [11].

UDDI provides two basic specifications that define a service registry’s structure and operations:

- a definition of the information to provide about each service, and how to encode it; and
- a query and update API for the registry that describes how this information can be accessed and updated.

UDDI encodes three types of information about Web services:
• **white pages** - information includes name and contact details;

• **yellow pages** - information provides a categorization based on business and service types; and

• **green pages** - information includes technical data about the services

Each of the information in the white, yellow and green pages are logically organized as a business entity, business service, business template and a **tModel**. The business entity provides standard white-pages information, including identifiers (IDs), contact information and a simple description. The business service represents the services it provides while technical description is provided through the **tModel** [19].

### 3.2.4 Technical Description: **tModel**

A **tModel** is a data structure representing a service type (a generic representation of a registered service) in the UDDI registry [19, 8]. Each business registered with UDDI categorizes all of its Web services according to a defined list of service types. Businesses can search the registry’s listed service types to find service providers. The tModel is an abstraction for a technical specification of a service type; it organizes the service type’s information and makes it accessible in the registry database. Another UDDI data structure, the **bindingTemplate** organizes information for specific instances of service types. When businesses want to make their UDDI specification-compliant services available to the registry, they include a reference to the tModelKey for that service type in their bindingTemplate data.

### 3.3 Types of Web Services

The types of services provided by Web Services vary and there are two major know types of Web Services. Firstly, known as action-based Web Services, these Web Services provide end-points for performing certain actions. An example of an action-based Web Service is a Web Service that calculates temperature from Celsius to Fahrenheit. On the other hand, another known type of Web Service is information-centric Web Service. Unlike, action-based Web Services, this type of Web Services retrieve information rather than performing actions. They process user input queries and return relevant results back to the user. The Australian Tourism Web Service (ATWS), is an example of information-centric Web Services which returns relevant tourism information based on the query made by a user.
3.4 Limitations of Traditional Information-centric Web Services

Traditionally, information that is described by information-centric Web Services are usually represented with minimal categorisation. This means these information are related minimally and even these relations are usually syntactical rather than semantically. For example, similar to the cases in Information Retrieval tasks performed by search engines in Section 2.6.3, if you were querying a word fly, it can have two different meanings, one being the insect fly or the action verb fly. Due to unavailable semantics over the query, results returned by these Web Services make no distinction between the two different meanings. In fact, these Web Services will return results that have the term fly in it.

One of the initiatives that researchers have proposed to overcome the problem of non-existence of semantics over traditional Web Services is the Semantic Web.

3.5 The Semantic Web: an Overview

The Semantic Web is an extension of the current Web in which information is given well-defined meaning, better enabling computers and people to work in cooperation [6]. The Semantic Web provides a common framework that allows data to be shared and reused across application enterprise and community boundaries. It is a collaborative effort led by the World Wide Web Consortium \(^1\) (W3C) with participation from a large number of researchers and industrial partners. It is based on the Resource Description Framework (RDF), which integrates a variety of applications using XML for syntax and URLs for naming.

The creation of the Semantic Web is largely due to the limitations provided by current Web Service Standards. Some of these limitations include the lack of computer understandable semantics, purely keyword-based discovery and lack of mechanisms for describing business relations.

3.6 Overcoming Limitation Through Semantics

The Semantic Web has been carefully designed to provide semantics through a well-defined framework. This framework provides additional schemas and terminologies on top of existing description so that information are no longer defined

\(^1\)http://www.w3.org/
as a standalone word but rather relations can now be made between these words. Within the Semantic Web, new description languages are created and proposed to allow information to be extensible. The information is not only semantically represented, but also computer understandable. This allows further automation of querying information and semantically relating other words. Next we discuss two languages that provide ontology descriptions that aim to provide semantics to Web Services through the Semantic Web initiative.

3.6.1 OWL-S: Semantic Markup for Web Services

The Ontology Web Language for Services (OWL-S) seeks to provide the building blocks for encoding rich semantic service descriptions, in a way that builds naturally upon OWL, the Web Ontology language. OWL-S previously known as DAML-S [5] aims to describe the Semantic Web Services. In traditional service matchmaking, matching is done through the use of input and output messages. In OWL-S, in addition to inputs and outputs, we have preconditions and effects, known to OWL-S as the atomic process. An important aspect of the Semantic Web service description is the specification and constraints as well as the preconditions and effects of a process or service. Thus, OWL-S provides preconditions that a service requester needs to satisfy prior to the execution of the service. Through these building blocks, OWL-S has allowed Semantic Web Services to leverage on richer service descriptions that allows better service queries.

3.6.2 LARKS: Agent Capability Description Language

Language for Advertisement and Request for Knowledge Sharing (LARKS) is an agent capability description language where matching is the process of determining whether an advertisement registered in the matchmaker matches a request [27]. LARKS allows the use local ontology to describe a word or concept in its specification. Using the ontology, LARKS enable inferences to be done through these additional semantics provided by the ontology. In addition, LARKS provide various types of matching filters that uses the following notions of specification matches to enhance its matching capability.

- **Exact Match** - Highest level and most accurate form of match. Description is equivalent literally, through variables or logically through inferences. Most Restrictive form of matching. Eg. a keyword match.

- **Plug-In Match** - Useful but less accurate form of match. This form of match occurs when the request is able to fit into an advertisement. Eg. a
request to find a service for calculating monthly budget can be plugged into an service that can calculate both monthly and yearly budget.

- **Relaxed Match** - Most useful but least accurate from of match. Relationship between description are determined by a numerical value that is returned. Two description match if the value is smaller than a preset threshold.

Through these notions, we can see that term “matching” does not merely mean having the same text. Some processes are involved in determining matches. These processes vary, with some purely syntactical, while others more semantically. Regardless, these processes allow LARKS to provide good capability matching in an automated fashion.

### 3.6.3 The Lack of Service Ontology in UDDI

Although Service and Capability Ontologies such as OWL-S and LARKS do provide explicit semantics that enables Web Services to have richer matching, they do not work well together with UDDI. Firstly, OWL-S and LARKS are both based on framework proposed by the Semantic Web initiative while UDDI is currently based on traditional Web Service framework. The additional services provided by OWL-S and LARKS are not compatible with UDDI which provides simple querying through its SOAP interface.

Currently, most Web Services are relying on the current Web Service framework. While the Semantic Web initiative does provide additional semantic through explicit service ontology from OWL-S and LARKS, currently it is extensively used for academic researches. Secondly, businesses that have already adopted Web Services will be resistant to the change, as the cost of adopting a new framework is costly. Thus, we would ideally like to be able to enhance querying of Web Services while using the existing Web Service framework.

Through service ontologies, service discovery is being supported by the Semantic Web. The use of agent capability discovery through ontologies description has been an aid as well. Similarly, if content related domain ontology is available, similar approaches can be used for enhancing the performance of information-centric Web Services.
3.7 Query Expansion of information-centric Web Services

We have seen that although the Semantic Web is a good source of enhancing querying results, its uses are limited. Hence, the use of using Query Expansion from the Information Retrieval area seems to be the quite appealing for enhancing and enriching queries. Similar with Query Expansion from search engines, we can integrate some of the Query Expansion techniques used by search engines onto information-centric Web Services. Thus, in addition to enriching the Web Services’ query with additional semantics, we have also bridged the gap between Web Services and Information Retrieval. Next, we discuss some of the Query Expansion techniques that can be integrated into Web Services to enhance the results obtained.

3.7.1 Ontology Distance

Ontology distance is the most intuitive measurement of concepts in an ontology. The similarity is measured according to the distance between the two concepts within the ontology [13].

A directed acyclic graph (DAG) of concepts in ontology is used to represent and calculate the ontology distance between the two concepts. The ontology distance is defined as the shortest path through a common ancestor or as the general shortest path, connecting two objects through common descendants/specializations. The pseudo-code algorithm is as follows:

1. gen\_a = all transitive generalizations of the concept A
2. gen\_b = all transitive generalizations of the concept B
3. from get\_a ∩ gen\_b find the Most Recent Common Ancestor (MRCA)
4. ontology distance = length of the path from A to MRCA to B

An example based on Figure 2.1 will demonstrate the use of ontology distance. The ontology distance between leopard and lion would be two, while the ontology distance between leopard and shark is four. Clearly leopard is more similar than lion as opposed to shark.

The advantage of using ontology distance is that ontology distance can be easily determined and calculated when the ontology is properly constructed. The disadvantage of the measure is that it effectively means that the decisions made when creating the ontology will largely determine the distance obtained between two similar objects in ontology.
3.7.2 Vector Space Approach

A common measure used in information retrieval is the vector space approach [4]. This approach represents each concept as a vector of features in a k-dimensional space and computes the similarity by measures such as cosine, Euclidean distance and Manhattan distance [13].

An concept in an ontology can be represented as a k-dimensional vector, where k are the number of unique attributes/relations describing a particular concept. The length of the k-th component of the vector is associated with the frequency of attributes in the concepts. The similarity of \( \text{sim}(A,B) \) between two concept A and B is now simply defined as the cosine of the two vectors. The steps for calculating are:

1. Determine vector \( x \) from attributes of concept A
2. Determine vector \( y \) from the attributes of concept B
3. \( \text{sim}(A,B) = \frac{|xy|}{|x| \times |y|} \)

Consider a concept-chair, which has four legs and one back as well as a type, indicating whether it is an office chair or not. This can be represented in a vector \([4, 1, 1]\) that would represent the chair in a vector space with three dimension \([\text{num_legs}, \text{has_back}, \text{is_office}]\). A table vector \([4,0,1]\) will be similar to a chair vector but not exactly the same as the table does not have a back.
Such a vectorisation has the advantage of being computationally cheap. It would be an ideal similarity measure for use when a quick approach is needed. However, the space vector approach suffers in the area of capturing semantic relationships. Obviously, the vector space approach is unable to capture that num_legs and has_back are closer related to each other than to is_office, although the space vector approach would state otherwise. As a result this would mean that vector space approach would be a less ideal approach to use when high similarity is required.

3.7.3 Edit Distance

The edit distance also known as the Levenshtein Distance is the similarity between strings by the number of changes needed to turn one string to the other [23].

A change is typically defined as either the insertion of a symbol, the removal of a symbol, or the replacement of one symbol with another.

As we are comparing objects in ontology, we need to calculate the number of transformation steps needed to turn one object into another object. It would include counting the number of insert, remove, and replacement operations of attributes, attribute values, relationships, or relationships types. Although in reality, the cost of replacement outweighs that of deletion and insertion. Hence, the cost function $c$ would have the following behavior:

$$c(\text{deleting}) + c(\text{inserting}) \geq c(\text{replacing}) \quad (3.1)$$

The algorithm works by first determining parts of A and then B. Next, the number of transformation steps from A to B is calculated and the worst case for that procedure as well.

The edit distance is finally calculated by taking the number of transformation steps over the worst case cost.

The pseudo-code algorithm is as follows:

1. Determine parts (attributes/relationships) of A
2. Determine parts of B
3. Compute number of transformation steps (replace, insert, delete) from A to B
4. Compute worst case cost of procedure
5. Relative edit distance=(number of transformation steps)/(worst case costs)

The edit distance has the advantage of being completely objective, and its results are verifiable. However, the weighting placed on insertion, deletion and replacement will largely affect the results being obtained for the similarity between objects. Combinatorial explosion is another disadvantage of this approach as a big ontology will result in a huge number of possible changes.

In order for this approach to be useful, we need to be able to decide on a suitable weighting scheme that will closely reflect the cost of changes. Thus, this method requires more thorough investigation and experimentation in order for it to be used.

3.7.4 Term Frequency/Inverse Document Frequency approach

The Term Frequency/Inverse Document Frequency approach is probably the most often used similarity measure in information retrieval literature. It compares two documents by using a weighted histogram of the words they contained [13].

This approach, specifically known as the “term frequency and inverse document frequency” (TF/IDF), assumes that term importance is proportional to the standard occurrence frequency of each term k in each document i and inversely proportional to the total number of documents to which term is assigned.

The approach works by counting the frequency of occurrence of a term in a document in relation to the words occurrence frequency in a whole corpus of documents. The result word counts are then used to compose a weighted term vector describing the document.

In order to put the approach into the context of concepts, a text document is created for each concept in the ontology. Each of the documents will contain the concept’s name, its attributes, and a brief description of its relationships.

The approach has its advantage because it assumes that rare terms are more important than common terms so that rare terms are made more important. However, this may or may not hold true when put into the context of concepts/objects and more tests are needed in order to produce an objective result.

3.8 Summary

Researches have been done to enhance Web Service discoveries by adding semantical relationships through service ontologies such as OWL-S and LARKS via the
Semantic Web initiative. Similarly, ontology-enriched information-centric Web Services should also provide semantically relevant results rather than literally similar keywords. Although seemingly viable, the resistance of businesses using Web Services adopting a new framework makes these approaches not ideal. Another alternative is to enrich queries from the current Web Service framework through Query Expansion. Using Query Expansion, we can retain existing framework while at the same time provide semantically relevant information.

In the next chapter, we apply some of the Query Expansion techniques that has been discussed and integrate them to achieve semantically enriched Query Expansion. A prototypical system is created to test out the Query Expansion ideas proposed.
CHAPTER 4

The Implementation

In this chapter, we design a prototype system that integrates some of the Query Expansion techniques discussed in previous chapters. By integrating them, we aim to enhance the query to an information-centric Web Service and obtain results that are not just syntactically relevant but also semantically relevant.

4.1 Overview: Design Purpose

Query Expansion enhances the existing service-oriented Information Retrieval tasks to return semantically relevant information. Information Retrieval tasks such as querying information-centric Web Services are usually keyword-based. Currently, we are able to obtain results from such queries, albeit a lack of semantics. Such queries usually return only information that contains the query keywords. Thus, semantically relevant information is not displayed as a result. The purpose of designing a prototype is to leverage the use of Query Expansion techniques onto existing queries and thus returning semantically relevant results.

4.2 External Schemas and Repository

Before we are able to study and experiment with the proposed Query Expansion techniques, a repository of data is needed. For the purpose of this experiment, we have chosen the Australian Tourism Data Warehouse (ATDW) as our repository. However, use of any available schemas and repository is allowed. Schemas of the ATDW are also obtained so that we are able to replicate the structure of the ATDW, thus enabling the experiment to be more relevant. Due to the large size of the ATDW, we will only simulate the accommodation classification of ATDW.

The ATDW is developed to encourage and assist the development of tourism in Australia. This is pioneered by the Australian Tourism Organization, the Australian States and Territories, and Tourism Australia [1, 2].
4.3 Architecture of Framework

The prototype integrates several Query Expansion techniques that can be applied consecutively. To enable modularity between each technique, the prototype is broken down into several major systems components. While not every component is mandatory to the functioning of the prototype, each component, however, does provide additional enhancement to Query Expansion. The ability to fine tune is in hopes that by applying the Query Expansion techniques together, more relevant results may be obtained.

Diagram 4.1 below depicts the overall sequence and structure of the prototype. In the following sections, each component is carefully discussed on its design and its objective.

Figure 4.1: A Diagram of the System Architecture
4.3.1 Data Repository

Before the prototype can process a query, a data repository is needed for the retrieval of information. In our example, the ATDW was chosen as the data repository. However, the use of this architecture is not specific to that of the ATDW, any data repository that has its schemas and data made available to programmers can be plugged in. The schema of the data repository is needed in the construction of ontology for the ontology distance.

4.3.2 Semantic Level: Ontology Distance

At the beginning, a user is required to enter a query where it is put through to the first filter of the prototype. The first filter uses Ontology Distance as its Query Expansion technique. In addition to a required query, the user is also prompted for the threshold where related results will be displayed. The threshold is a number between one to four. One represent results closely related via ontology distance and four represent least closely related via ontology distance.

Iterating through the data provided by the ATDW, the first filter extracts all results that contain the query keyword. Upon obtaining the results, each result is checked against the threshold and the ontology. By examining the ontology, each result whose ontology distance is no greater than the threshold provided will be returned back to the user.

Figure 4.2 shows the internal flow of the Ontology Distance at work.

![Figure 4.2: Ontology Distance](image-url)
4.3.3 Fine Tuning: Vector Space Similarity

The next step attempts to narrow the results obtained through Ontology Distance. Results from the Ontology Distance may only provide results that are ontologically closer. The vector space approach, however, can fine tune these results with the same Ontology Distance by attempting to calculate their similarities. This results in a clear ranking after fine tuning of results.

![Vector Space Diagram]

Figure 4.3: Vector Space

We approach the vector space algorithm by first creating nodes to represent each result returned. In our example, the use of weightings has been discarded. Below shows a sample code for the creation of a single vector. The node class can be found in appendix B.

```java
HashMap vectorA = new HashMap();
for(int i=0;i<selected.length;i++){
    String text = selected[i];
    if(!vectorA.containsKey(text)){
        Weight w = new Weight(text);
        vectorA.put(text,w);
    }
}
for(int i=0;i<param.length;i++){
    String text = param[i];
    if(!vectorA.containsKey(text)){
        Weight w = new Weight(text);
        vectorA.put(text,w);
    }
}
```
4.3.4 Popularity search: Search Engine

Occasionally, results obtained through the vector space approach might not be accurate in determining similarities between concepts. The use of Vector Space similarity might not be accurate in ranking results obtained through the use of the prototype. This is due to the algorithm approaching the problem of similarity using a more syntactical approach.

An alternative approach from using vector space approach is verifying the ranking of results by putting them through a search engine. Each result’s popularity ranking in the search engine is matched against each result and the results are ranked according to their popularity in the search engine.

We use Google as our search engine to determine the ranking of results. It is obtained through the use of Google Web API by the insertion of the code below. Based on the inserted query, an estimated result count will be returned from Google.

```java
GoogleSearch s = new GoogleSearch();
s.setKey(clientKey); //required by Google to use Web API
s.setQueryString(string);
GoogleSearchResult result = s.doSearch();
int result_num = result.getEstimatedTotalResultsCount();
```

4.3.5 Additional Components

In some cases, queries may not return any relevant results. This is due to the prototype unable to determine any similarity between the query keyword and the domain of interest. If such cases occur, the prototype can apply additional external techniques, finding other similar words that could match to the original query keyword.

Incorporating WordNet

There are scenarios where a term known to a user may not be used in the domain of interest. Such instances could occur when domain specific terms are used or terms that are used specifically by certain nationalities of people. In such cases, the most ideal alternative is to find alternate terms through lexical relationships.

To be able to identify lexical relationship between concepts in our prototype, we use WordNet as our database. To be able to retrieve results from WordNet, we need methods for accessing WordNet-style relational dictionaries. This comes
in the form of Java Wordnet (JWNL\(^1\)). With JWNL, we can extract terms such as synonyms or hypernyms to expand our Web queries. For example, if you have a query hotel, some related words that will be returned are inn and lodge which are related to the original query. In our prototype, this will allow alternate terms to be obtained. Therefore we can apply these terms back to the query.

```java
private void getTerms(IndexWord word) throws JWNLException {
    PointerTargetNodeList hypernyms = PointerUtils.getInstance()
        .getDirectHypernyms(word.getSense(1));
    hypernyms.print(21);
    PointerTargetTree syntree = PointerUtils.getInstance()
        .getSynonymTree(word.getSense(1), 1);
    System.out.print("\t\t ");
    syntree.print();
}
```

The above shows a section of source code that display alternating terms that is related to the initial query.

**Google Web APIs**

Google Web APIs\(^2\) allow users or developers to query more than 8 billion Web pages directly from their own computer programs. In addition, we are able to discover the total occurrence of any query. This will allow us to return to the user the occurrence of other terms identified by WordNet. This can be useful as the user can then decide whether which additional terms can be used for redefining their query. Using the example of inn and lodge, we discover through Google that inn (80 million) is a far more common term than lodge (76.6 million), and we will indicate to the user that inn should be higher ranked than lodge.

On top of that we can also use the suggested spelling by Google if we have made a mistake when entering our query.

The Google Web APIs is integrated with our prototype through its Java interface where it has a pre-defined set of classes and methods that allow us to return the total number of occurrence of a term.

### 4.4 Summary

The implementation of the prototype is a collection of techniques used for Query Expansion. Integrating them into a single system allows us to be able to enhance

\(^1\)http://jwordnet.sourceforge.net/
\(^2\)http://www.google.com/apis/
keyword based searches with semantic enrichment. Use of ATDW allows live data experiments while use of WordNet and Google show that existing technologies can be use for Query Expansion as well. The experiment results of the implementation are discussed in the next chapter.
CHAPTER 5

Experimental Analysis

In this chapter, the prototype proposed in Chapter 4 is developed. An experiment was done to obtain query strings for querying the ATDW Web Service. The experiment provides evidence that Query Expansion does provide some enhancement to the performance of information-centric Web Services. The experiments involve testing queries on the ATDW with and without the use of Query Expansion. A comparison is made to distinguish the effects of Query Expansion based on the results obtained.

5.1 Experiment

To demonstrate the use of Query Expansion, two sets of query strings were tested. They were tested using the ATDW Web Interface with and without the use of the Query Expansion prototype. The terms inn and hotel were chosen to demonstrate the experiment because we were only using the “accommodation” category of the ATDW for querying. Before the end, five individuals participated the use of the prototype to determine the satisfaction level of the Query Expansion prototype.

5.2 Query on ATDW without Expansion

Using the query string inn, we ran the query through the ATDW Web Interface. In Figure 5.1, we see a total of 704 relevant results were returned as a result of keyword match within the name of the accommodations. The relatively large number of results returned were a result of a large number of accommodations that had the word inn within their names. To confirm that this was consistent with other queries, we did another query using the string hotel, and we found that a total of 763 results were returned. This is consistent with the previous
query as both queries were both popular queries based on the number of results returned by Google.

5.3 Query on ATDW with Expansion

Using Query Expansion developed through our prototype, we inserted the query string `inn` into our application and through Query Expansion, derived the following alternate keyword terms that can be used. In Figure 5.2, we can observe the additional terms and they are `hotel`, `hostel`, `hostelry` and `lodge`.

As our prototype, discussed in Chapter 4, is unable to directly insert the additional terms discovered directly into the ATDW Web Interface, we had to manually insert these terms directly into the ATDW Web Interface. The results are tabulated in Table 5.1.

In all cases, other than the term `hostelry`, the ATDW was able to narrow
Figure 5.2: Result of Query Expansion on **inn**

<table>
<thead>
<tr>
<th>Query Expansion</th>
<th>hotel</th>
<th>hostel</th>
<th>hostelry</th>
<th>lodge</th>
</tr>
</thead>
<tbody>
<tr>
<td>Result</td>
<td>97</td>
<td>4</td>
<td>0</td>
<td>15</td>
</tr>
</tbody>
</table>

Table 5.1: Query Expansion of query string **inn**

down the search space significantly. Results returned were more relevant, despite the additional terms having no syntactical similarities.

5.4 Comparison and Discussion

Having compared the differences between the query applied on the ATDW without expansion and with expansion, we can see that the initial query results of 704 have been reduced. In most cases, the results returned after applying Query Expansion had less than 100 results returned. These query expanded results retained relevance to the initial query string.

We see that through the use of Query Expansion, we are able to derive additional terms that are semantically related to the original query string. Despite the ATDW Web Interface having no support expanding query automatically, we have shown that through Query Expansion, we are able to significantly reduce the number of results returned by ATDW, while still maintaining relevance. The result will be more significant if we use semantically ambiguous words.

Before we concluded the experiment, we had five student evaluators participating in the use of the prototype to determine if they were satisfied with the new results obtained through Query Expansion. They were all asked whether the prototype was useful in obtaining more specific results through Query Expansion.
A scale of 1 to 5 was chosen where 1 represented not useful and 5 represented very useful. The results are tabulated in Table 5.2.

<table>
<thead>
<tr>
<th>Participant</th>
<th>No.1</th>
<th>No.2</th>
<th>No.3</th>
<th>No.4</th>
<th>No.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Result</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

Table 5.2: Query Expansion of query string inn

Out of the 5 participants, 3 participants expressed that the prototype did proved to be useful in obtaining more specific results through Query Expansion. However, 2 participants found that although the prototype did obtain much more specific results, the use of single terms as query strings made it limited in use.
CHAPTER 6

Conclusion and Future Work

In this chapter, we conclude our work and discuss the open problems as well as future works that can be done to improve the effectiveness of Query Expansion of information-centric Web Services.

6.1 Conclusion

Information-centric Web Services are Web Services retrieving information rather than performing actions. We have observed that due to the lack of semantics in these Web Services, we need highly precise queries to retrieve the desired information. Improving queries to obtain better results require more than just returning information containing query keywords but also semantically related results. Alternative solutions include the use of semantically enhanced Web Services provided through the Semantic Web. We have examined the problems of using semantically enhanced Web Services, and they are not suitable for use due to drastic changes needed to existing approaches.

We have looked at existing Information Retrieval types of tasks such as search engines and how they are similar with retrieval of information from information-centric Web Services. These similarities are compared and the result is the use of Query Expansion techniques to enhance the performance of information-centric Web Services. The achievements of this dissertation include:

- identifying limitations of current queries to information-centric Web Services.
- identifying similarities between search engine and Web Service queries.
- proposed and implemented a framework for integrating some Query Expansion techniques to query information-centric Web Services.
Despite having developed a prototype that uses Query Expansion for enhancing the query to an information-centric Web Service, there is still much research needed to be done. Extensive work would be needed if we wanted to fully automate Query Expansion into information-centric Web Services. Though not exhaustive, we hope that this dissertation can serve as a basis for future development of Query Expansion techniques for querying information-centric Web Services.

6.2 Limitation of Work

In our experiment, although the schema of the ATDW was provided, the data that is represented by the schema remains directly unavailable. Due to this constraint, much of the information had to be retrieved manually through the ATDW Web Service. This poses problems in future implementation in automation for querying as a full representation of data is not available. We suggest that future use of data repository to be made fully available so that automation can be made easier. In addition, the ATDW Web interface does not allow keywords to be applied as a conjunction nor a disjunction. This effectively means that if more than one keyword is present inside a query string, they are treated as a single entity for querying, thus limiting level of Query Expansion and the results returned.

6.3 Future Works and Open Problems

Due to time constraint of the honours project, we have not fully explored many areas that could make Query Expansion yield better results. The following are some future works and open problems that have been identified and could be thoroughly investigated in order to extend the current work.

Satisfaction Level of Results

In our experiment, we did a short questionnaire with a few individuals to obtain a satisfaction level of results with regards to Query Expansion over a lack of it. Due to time constraint, the size of the questionnaire was reduced. In addition, the use of simple questions in our experiment yielded results less valuable than desired. We suggest that a larger pool of individuals should participate in the system evaluation. This would yield more convincing results that we could use to compare the effectiveness of Query Expansion on information-centric Web Services.
Part of Speech Tagging

Many words are ambiguous in their part of speech. However when a word appears in the context of other words, such ambiguity is often reduced. For example, “tag” can be a noun or a verb. It could be a noun when you consider it as a label or a verb if you consider the action of tagging. In a “tag is a part of speech label”, the word “tag” can only be a noun.

In our prototype, the query string chosen were single term nouns as we were seeking to find terms which represent names. The use of multiple terms would yield limited results as our prototype had no knowledge of part of speech representation of the terms. As such, we suggest the use of a part of speech tagger, a system that uses context to assign part of speech to words, to further enhance Query Expansion. There are many researches done in the area of automated part of speech taggers using methods such as hidden Markov model [12], which could allow Query Expansion to yield better results.

Modifier vs Head Nouns

In our prototype, our query string has been limited to the use of a single noun term. The use of complex noun phrases did not yield any relevant result, because the prototype did not have the ability to differentiate between modifiers and head nouns. Usually the head noun can be expanded to provide alternate terms. An example would be the query string “Hyatt Hotel”, where Hyatt is the modifier and Hotel is the head noun. The word Hotel can be expanded to yield words such as inn and lodge while Hyatt, being the modifier is unable to be expanded. Without the ability to differentiate between modifiers and head nouns, it is not possible for any automated system to expand on a query because it simply does not know which term within the query can be expanded.

However, there are researches done in an attempt to automate recognition between modifiers and nouns. The use of lexical dictionary and partial matching of previous analysed noun phrases are but some of the proposed methods [20]. If we are able to identify modifiers and head nouns within our query strings, we could possibly increase the enhancement made to Query Expansion.
APPENDIX A

Original Honours Proposal

Background

Semantic Web [28] is the next generation of the Web which evolves toward semantic knowledge representations and intelligent services (e.g. information brokers, search agents) where information can be processed by machines. Web Services are emerging to provide a systematic and extensible framework for application-to-application interaction that is built on top of exiting Web protocols and based on open XML standards.

A Web service is defined as a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine processable format, typically using Web Services Description Language (WSDL). Other systems interact with the Web service in a manner prescribed by its description using Simple Object Access Protocol (SOAP) [24] messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards [15]. The Universal Description, Discovery and Integration (UDDI) registry server acts as a source of directory listing for Web services to locate one another.

Traditional Web service matchmaking is usually limited to keyword based matching as WSDL does not describe the Web services semantically. Although there are Web ontology languages such as LARKS and OWL-S (Web Ontology Language for Services), these languages are currently experimental and unable to leverage on existing protocols that are used in traditional Web service matchmaking.

In order to discover services on the Web, good similarity measures are needed to matchmake service providers and service requesters on the Web. Extensive research has been done in the area of finding good similarity assessment algorithm for use in ontologies. This is central to the functioning of techniques such as retrieval, matchmaking, clustering, data-ming, ontology translations, automatic database schema matching, and simple object comparisons. One of the
important types of similarity measures that have been identified has been determining the semantic similarity between objects in ontology. Humans typically have little difficulty in determining the intended meaning of ambiguous words, expressions, or even complex objects, whereas it is challenging to replicate this process computationally [13].

Object-based similarity measure, unlike keyword-based matching, has the capability to provide more details by defining entities with attributes, attribute values, and relationships which one might be a specialisation. This, subsumes explicit ontologies such as WordNet [25], where the specialisation relationship is explicitly defined, as well as ontologies, where this relationship is to be derived logically.

In order to facilitate the semantic Web; efficient, accurate and semantic rich matchmaking is an inseparable component. As mentioned, Web services are limited to keyword based matching, which severely limits the potential of finding a good match in response to a request. Object-based similarity measure that is based on semantic similarity in ontologies however, has the capability to have different levels of matching.

Research Aim

The main research aim of this project is to investigate current similarity measures in the information retrieval literature and using these to extend current Web Service matchmaking. And thus, this will hopefully allow us to incorporate semantic capability into the matchmaker to provide semantically rich matching.

Initially we need to have a broad understanding of the current similarity measures that is currently available or being developed in the information retrieval literature. The current work will be reviewed in order to give us a better perspective of the current capabilities of similarity measures. The review will then be followed by a critique of the measures to provide us with an objective view of the measures.

Having criticised the measures, we will attempt to evaluate in the context of service matchmaking and propose appropriate measures to incorporate into our Web service matchmaker. This is done to extend the matchmaker to incorporate semantic relationships. A working model of the matchmaking algorithm will then be put to test in a tourism domain and specifically Australian Tourism Data Warehouse (ATDW).

We expect to be able to present a service matchmaking algorithm that is object-based in the domain of tourism in order to show that object-based match-
making can be incorporated into Web services to exhibit semantic properties. Having said that, this provides a promising ground to allow us to exploit object-based similarity measures onto Web service matchmaking.

Methodology

The project will begin with a literature review on current similarity measures as well as a background on how Web services work in general. This will be followed by categorising the different similarity measures based on their matching criteria.

Each of these similarity measures will then be evaluated in the context of Web service matchmaking. This will be done through experimentation and reviewing. An objective result will be presented as well.

An appropriate measure will then be proposed and implemented into the Web service matchmaker. This implementation will make use of data made available from ATDW.

Finally, the project will compare the different object-based similarity measures through the implementation of the matchmaking algorithm.

Evaluation

The project will be evaluated against its aim of bridging object-based similarity measures and Web services. The project will be tested for performance of matchmaking service request and service provider provided by the data in ATDW Web service. Verification of output is done against output and the performance is evaluated against that of existing implementation.
Research Timeline

<table>
<thead>
<tr>
<th>Date</th>
<th>Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>17 Mar</td>
<td>Submit Proposal</td>
</tr>
<tr>
<td>18 Mar - 27 Mar</td>
<td>Literature Review</td>
</tr>
<tr>
<td>28 Mar - 1 Apr</td>
<td>Proposal presentation to research group</td>
</tr>
<tr>
<td>2 Apr - 29 May</td>
<td>Continue literature review and categorisation of measures</td>
</tr>
<tr>
<td>26 May</td>
<td>Submit revised proposal</td>
</tr>
<tr>
<td>30 May - 7 Aug</td>
<td>Implementation of object-based measure onto Web service</td>
</tr>
<tr>
<td>8 Aug - 4 Sep</td>
<td>Evaluation and criticism of work</td>
</tr>
<tr>
<td>5 Sept - 14 Sep</td>
<td>Prepare draft dissertation</td>
</tr>
<tr>
<td>15 Sep</td>
<td>Submit draft dissertation</td>
</tr>
<tr>
<td>29 Sep</td>
<td>Collect draft dissertation</td>
</tr>
<tr>
<td>30 Sep - 19 Oct</td>
<td>Review and refine dissertation, seminar and poster</td>
</tr>
<tr>
<td>20 Oct</td>
<td>Submit final dissertation</td>
</tr>
</tbody>
</table>

Software and Hardware Requirements

The computers in the Honours laboratory are more than sufficient to conduct the experiment and implement the matchmaker. The Java Development Kit is required to be installed in order to able to program in Java.
Bibliography


[5] *OWL-S: Ontology Web Languages for Services*  
  http://www.daml.org/services/owl-s/.


[7] *Thesaurus: Definition*  


[11] *Introduction to UDDI*  


[18] Colan, M. *International Business Machine (IBM)* 


[23] Gilleland, M. *Levenshtein Distance, in Three Flavors* 


APPENDIX B

Listings

B.1 Node.java - Vector Space

```java
import java.util.*;

public class Node {

    private Node parent;
    private ArrayList nodeList;
    private String[] attributes;
    private String name;

    //top level Node creation
    public Node(String nm)
    {
        name = nm;
        parent = null;
        nodeList = new ArrayList();
        String[] attributes = null;
    }

    public Node(String nm, Node parentNode)
    {
        name = nm;
        parent = parentNode;
        nodeList = new ArrayList();
        String[] attributes = null;
    }

    public String getName()
    {
        return name;
    }

    public String[] getAttributes()
    {
        return attributes;
    }

    public void setAttributes(String[] attr)
    {
        attributes = attr;
    }

    public void setParent(Node aNode)
    {
        parent = aNode;
    }

    public Node getParent()
    {
        return parent;
    }
```
public void addChild(Node aNode) {
    nodeList.add(aNode);
}

public Node getChild(int num) {
    return (Node) nodeList.get(num);
}

public int getChildrenSize() {
    return nodeList.size();
}

public ArrayList getList() {
    return nodeList;
}