2011

Dynamic generation of reduced ontologies to support resource constraints of mobile devices

Dan Schrimpsher

DYNAMIC GENERATION OF REDUCED ONTOLOGIES TO SUPPORT RESOURCE CONSTRAINTS OF MOBILE DEVICES

by

DAN SCHRIMPSHER

A DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
in
The Department of Computer Science
to
The School of Graduate Studies
of
The University of Alabama in Huntsville

HUNTSVILLE, ALABAMA

2011

In presenting this dissertation in partial fulfillment of the requirements for a doctoral degree from The University of Alabama in Huntsville, I agree that the Library of this University shall make it freely available for inspection. I further agree that permission for extensive copying for scholarly purposes may be granted by my advisor or, in his/her absence, by the Chair of the Department or the Dean of the School of Graduate Studies. It is also understood that due recognition shall be given to me and to The University of Alabama in Huntsville in any scholarly use which may be made of any material in this dissertation.

(student signature)

8/8/2011

(date)

DISSERTATION APPROVAL FORM

Submitted by Dan Schrimpsher partial fulfillment of the requirements for the degree of Doctor of Philosophy in Computer Science and accepted on behalf of the Faculty of the School of Graduate Studies by the dissertation committee.

We, the undersigned members of the Graduate Faculty of The University of Alabama in Huntsville, certify that we have advised and/or supervised the candidate on the work described in this dissertation. We further certify that we have reviewed the dissertation manuscript and approve it in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Computer Science.

4/25/11 (Date) Committee Chair

4-26-11

4/26/2011

4/26/2011

4-26-11

4-26-11 Department Chair

College Dean

8/7/11 Graduate Dean

ABSTRACT

The School of Graduate Studies
The University of Alabama in Huntsville

Degree Doctor of Philosophy College/Dept Science/Computer Science

Name of Candidate Dan Schrimpsher

Title Dynamic Generation of Reduced Ontologies to Support Resource Constraints of Mobile Devices

As Web Services and the Semantic Web become more important, enabling technologies such as web service ontologies will grow larger. At the same time, use of mobile devices to access web services has doubled in the last year. The ability of these resource constrained devices to download and reason across these ontologies to support service discovery will be severely limited. Since concrete agents typically only needs a subset of what is described in a web service ontology to complete their task, a reduced ontology can be created.

In this research, we define computable measures of quality and performance with respect to reduced ontologies. Using these measures, we developed a dynamic algorithm to generate a reduced ontology that contains the necessary knowledge content and performance speedup to be used by a mobile device for service discovery.

Abstract Approval

Committee Chair Letitia Pura 4/25/11

Department Chair Heggere S. Ranganath

Graduate Dean Rhonda Kay Hoede 8/7/11

ACKNOWLEDGMENTS

First, I want to express my gratitude to my advisor, Dr. Letha H. Etzkorn, for her encouragement and forethought throughout the course of this dissertation. I especially want to thank her for her advice and patience.

I also wish to thank the other members of my committee, Dr. Harry Delugach, Dr. Glenn Cox, Dr. Dan Rochowiak and Dr. Sampson Gholston, for their advice and suggestions.

I want to thank my wife, Ann, for putting up with long hours of research and writing and occasionally spending vacations at research conferences. In addition to raising two kids, she helped review my work and provided a sounding board for ideas. I also want to thank my children, Hannah and Josh, providing inspiration for this whole thing.

TABLE OF CONTENTS

page

LIST OF FIGURES

LIST OF TABLES

CHAPTER ONE

INTRODUCTION

Web services are a de facto part of daily life for computer users. They are used to access weather reports, reserve plane reservations, and find the best sushi while traveling. User tasks are becoming more complex, leading to a desire for the computer to do more of the underlying work. The Semantic Web is an attempt to build machine readable web services, so software agents can handle more of the decision making in completing a complex task. Functioning as a Web Service API, they allow users to use services that were independently developed to work a task, facilitate the use of agents to collect, process, and exchange information necessary to complete the task, and permit web service requests based on a user defined concept.

At the same time, the use of resource constrained mobile devices, such as mobile phones and netbook computers, is growing at an astounding rate. The International Data Corporation predicts that mobile devices shipments are expected to reach 82 million by 2011 as reported by Cellular-News [4]. These mobile devices allowed users to access web resources away from traditional computers. A 2009 study found 63 million mobile device users accessing web resources with 22 million users doing so on a daily basis [29].

Given these two directions the world is heading, it is important to consider how these two technologies will integrate in the future. What techniques need to be developed to allow the massive number of mobile devices being integrated into our society to effectively use the semantic web? How can we structure the semantic web to be usable by both mobile and traditional computers? The goal of our research was to begin finding techniques to allow the integration of our mobile lives and the coming Semantic Web.

The Semantic Web uses Ontologies to describe the relationships between a set of concepts in a domain. Simple, non-semantic web services might specify which operations are available in a web service, and would describe the structure of the data that is received and transmitted via these operations. However, non-semantic web services would not specify the semantic meaning of the operations or data, or their semantic interrelationships. A number of models to build web service ontologies have been developed, including OWL-S and the Semantic Web Services Ontology. For example, OWL-S has been used with the Amazon Web Service framework to provide services to users [28]. Functioning as a Web Service API, semantic web services allow users to use services that were independently developed to work a task, facilitate the use of agents to collect, process, and exchange information necessary to complete the task, and permit web service requests based on complicated semantic concepts rather than simple keywords.

As semantic web services become more widely used, it is natural that the ontologies they use will grow in size. Nov and Musen observed that there is a computational penalty if an agent has to perform reasoning across a large ontology. This

is particularly noticeable if the agent only needs a small subset of the entire ontology [32][33]. Software agents most often use a subset of concepts and properties, rather than a full ontology [1]. According to Seidenberg & Rector, most ontology reasoners currently available cannot even handle a medium-sized ontology [23].

Concrete agents do not usually need comprehensive domain descriptions. They most often use a subset of concepts and properties, rather than a full ontology [42]. In these cases, a subset or reduced ontology of the service ontology, geared to the agent's request, would be useful to allow the agent to access only what is necessary to complete its task. For mobile agents and the semantic web to reach their full potential, an efficient method for reducing an ontology or fragmenting an ontology needs to be defined.

In the case of mobile devices, such as cell phones and PDAs, this problem is exacerbated. Today, mobile devices are being used more and more like a portable computer. However, performance and communications limitations make it very difficult to handle a full ontology of services provided. Although mobile devices are continuing to evolving into more capable computers, they are still limited by power usage and size. These kinds of relative performance limits will continue for the foreseeable future, particularly since ontologies and web services are themselves becoming larger and more complex. For example, the full Suggested Upper Merged Ontology (SUMO) at present contains 20,000 terms, 70,000 axioms, and 3000 rules [34]. Seidenberg and Rector found most ontology reasoners currently available cannot handle a large ontology [23]. Even on a traditional computer large ontologies must be broken into modules to be processed on a personal computer. Clearly, it would be difficult or even impossible for a resource constrained mobile device to download and process a large ontology.

For example, it is unreasonable to expect an iPhone to process a large ontology, like SUMO, with 20,000 terms quickly. Similarly, it is impractical to expect a Blackberry to download and reason across an ontology with 10,000 nodes in a reasonable time; this is particularly unacceptable when only a small portion of the ontology is needed to complete the task. Even ignoring the download time, the computational penalty is prohibitive for an agent, located on an iPhone or a Blackberry, to perform reasoning across a large ontology, when it only requires a small subset of it [23]. A reduced ontology, focused on the agent's request, would enable the agent to access what is necessary to complete its task.

Creating a quality reduced ontology is a nontrivial task. Grau et al. demonstrated that creating a minimal module of an ontology is not algorithmically solvable in a reasonable amount of time [16]. Since an optimal subset is not algorithmically practical, what is needed instead is a useful/"good enough" subset that can be generated in a reasonable time. A subset must be complete enough to allow usefulness, but contain minimal extraneous information. It must also be small enough for the agent to handle. If we are seeking more automation in web service discovery and use, it makes sense to automate the reducing an ontology process. The first question is, when is a reduced ontology "good enough" to be used instead of the original complete ontology? Also, once we have the ability to judge a reduced ontology ontology's quality, the second question is, how do we deliver it to the software agent in a reasonable time? Creating all possible reduced ontologies a priori is not feasible, since it would require exponential time and space. Creating a subset of likely reduced ontologies may not work for all requests, in the worst case a request has overlap with all stored reduced ontologies and ends up with

the full ontology. Thus, the purpose of our research is the following: we wish to dynamically generate a high quality ontology reduced ontology based upon a software agent's request in a reasonable time. The background/literature review required to understand our approach is provided in Chapter Two, and we discuss our overall research approach in Chapter Three.

In order to build an algorithm to build a reduced ontology, we first needed a definition of what characteristics of an ontology meant the ontology was "good enough" to use. As a first step we developed a predictive model for information quality and performance. In this work, we created a group of metrics specific to reduced ontologies and used these along with traditional structure metrics for ontologies. These metrics were then compared to information recall and processing time over a set of mechanically generated reduced ontologies. From this, we identified predictive models that can be used to predict the informational quality and performance improvement of a reduced ontology (to be published as [41]). We discuss our predictive models and our validation procedure for these in Chapter Four.

Once our model of "goodness" was defined, we developed a dynamic reduced ontology generator (DROG) algorithm, to dynamically build reductions of the original ontology suitable for mobile devices to download and use for inference. DROG is a based on a user request for a particular concept, called the concept of interest (COI). This approach is dynamic, meaning it can operate at runtime, as clients make requests. It employs the easy-to-calculate metrics in the previously developed predictive model to generate reduced ontologies that will be a good substitutes for the original ontology. By using these smaller reduced ontologies for downloading and reasoning over, the overall

performance of the mobile devices are improved. We provide a full description of our DROG algorithm and our validation of this algorithm in Chapter Five.

Our final end to end testing of our DROG algorithm as it applies to mobile devices is discussed in Chapter Six. In this chapter, we discuss how we first developed a simulated web server using a blackboard architecture. In our simulation, a controller handled time constraints and chose the best solution available when a set time completed. In this simulation, we used a wide variety of ontologies, from small to very large. Second, we executed Monte Carlo runs of the simulation using RIM's Blackberry simulation for PC across currently available networks (these networks included 2G, EDGE, 3G, and Wi-Fi) for the small and medium sized ontologies used in previous experiments. The RIM's Blackberry simulation for PC that we employed is the Blackberry smart phone simulation, available from RIM's SDK [54]. For each ontology, we used a sample request and generated a reduced ontology which was then downloaded and queried by the smart phone. Finally we chose a large, publicly available ontology, Word Net, and executed it in the same environment. This last piece is a more representative case of where DROG would provide benefit in a real world application.

Our simulation was based on a blackboard architecture that would allow multiple algorithms to be run simultaneously as separate intelligent agents in a cooperative gaming scheme. Each of these agents ran its algorithm independently of other agents, but they all used the blackboard data structure to store intermediate results and communicate [25]. This architecture allows flexibility for the future to allow new algorithms to be tested by simply implementing the agent API. It also allows multiple algorithms to handle different kinds of ontologies and different constraints dynamically.

CHAPTER TWO

LITERATURE REVIEW

In this section, we will give an overview of ontologies, the semantic web, and research that led us to the idea and implementation of the dynamic reduced ontology generator.

2.1 Ontologies

A widely quoted definition of an ontology is Tom Gruber's [17]:

When the knowledge of a domain is represented in a declarative formalism, the set of objects that can be represented is called the universe of discourse. This set of objects, and the describable relationships among them, are reflected in the representational vocabulary with which a knowledge-base program represents knowledge. Thus, in the context of AI, we can describe the ontology of a program by defining a set of representational terms. In such ontologies, definitions associate names of entities in the universe of discourse (for example, classes, relations, functions, or other objects) with human-readable text describing what the names mean, and formal axioms that constrain the interpretation and well-formed use of these terms.

Computer programs can use ontologies for a variety of purposes. For example, semantic web searches can be enhanced by inductive reasoning -concepts within the ontology that have been identified as matching the search criteria would serve as a basis for the reasoning [2]. Ontologies have been used for the classification of email; in this work, the ontology captures rules that are used for feature identification [47]. Crubezy & Musen discuss how ontologies have been used in various problem solving methods, for example, to aid in heuristic classification and in comparing role limiting methods [6]. Ontologies have been used in various forms of inter-software communication, for example, Takeda et al. have used ontologies in agent communication [48].

The Resource Description Framework, or RDF, is a method for modeling information, which is often used on the World Wide Web. An RDF resource can be a local class, an external class, or a data type. The Web Ontology Language (OWL) is a semantic Markup language that extends RDF for describing an ontology on the World Wide Web.

We provide OWL/RDF examples here so that readers unfamiliar with these formats may understand the specific formulations of some of our metrics. RDF, in particular RDF-S, provides some support for the specification of ontologies, but more expressiveness was necessary for extensive ontology definition. According to Lacey, OWL was developed to add this additional expressiveness [24]. Sowa pointed out that ontologies may also be defined in other powerful knowledge representation languages such as conceptual graphs [44]. Analogues of these metrics could easily be developed for conceptual graphs or other knowledge representations, so this research should be

considered to apply to these other k knowledge representation formats as well. However, since the world wide web primarily employs RDF/OWL type ontology representations, it makes sense to illustrate our work in these commonly used representation languages.

OWL describes concepts and their relationships through the use of classes and relations between classes. OWL contains formal semantics that allow reasoning about concepts to a limited degree. OWL's syntax is a series of tags prefixed by *owl* and *rdf*. A class with CLASSNAME being a child of PARENTCLASS is defined as

<owl:Class rdf:ID="CLASSNAME">

<rdfs:subClassOf

rdf:resource="PARENTCLASS">

</owl:Class>

OWL also defines three types of properties, Object, Datatype, and Functional. Object properties have a range of another OWL class. Datatype properties have a range that is a datatype primitive (e.g., integer or string). Functional properties are either Object or Datatype properties where there is at most one value in the range for each value in the domain. Examples of these are shown below.

<owl:ObjectProperty rdf:ID="PROPERTY1">

<rdfs:domain rdf:resource="#CLASSNAME"/>

<rdfs:range rdf:resource="#CLASS2"/>

</owl:ObjectProperty>

<owl:DatatypeProperty rdf:ID=" PROPERTY2">

<rdfs:domain rdf:resource="#CLASSNAME"/>

<rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>

```
</owl:DatatypeProperty>
<owl:FunctionalProperty rdf:ID=" PROPERTY3">
        <rdfs:domain rdf:resource="#CLASSNAME"/>
        <rdfs:range
        rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
        <rdf:type
        rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID=" PROPERTY4">
        <rdfs:domain rdf:resource="#CLASSNAME"/>
        <rdfs:range rdf:resource="#CLASS2"/>
        <rdf:type
        rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
</owl:FunctionalProperty>
```

While our methods are not limited to ontologies defined in OWL, we chose OWL-based ontologies as our focus due to their extensive use in the World Wide Web and their availability for testing.

2.2 Web Service Ontologies

As the semantic web grows, the use of ontologies is changing from being employed by a semantic tool to illustrate examples of data to being an integral part of searching for and using multiple web services to achieve a single task. We define these kinds of ontologies as web service ontologies.

A web service ontology will define both the terminology and information required to access the web service for a user or an agent. Web service ontologies provide a machine-readable description of a group of services and their consequences [51].

The Semantic Web Service Ontology initiative [56] integrates the Process Specification Language with web services by integrating OWL-S style atomic services, WSDL like messages, and process and data flow models of the web services [19]. As web services and agents become more important, the use of semantics provided by ontologies will become more important.

Vrandecic & Sure define a web service ontology as both the terminology and information required to access web services by providing a machine-readable description of a group of services and their consequences [51]. Martin et al. demonstrated this ability through a specific implementations of OWL, known as OWL-S, which was designed to work with existing web service protocols, including the web service description language (WSDL) and the Simple Object Access Protocol (SOAP) [28]. OWL-S enhances WSDL with the ability to incorporate semantics into web service discovery and use. The utility of OWL-S has been demonstrated through its use in enhancing Amazon's WSDL based web services. In this project, it was demonstrated that through ontology-based semantics a client agent can query Amazon's database without requiring human interaction, as would have been necessary if using only WSDL.

Hull advanced the Semantic Web Service Ontology initiative given by DAML by integrating the Process Specification Language with web services [19][56]. He integrated OWL-S style atomic services, WSDL like messages, and process and data flow models of

the web services. This yields semantically accessible web service discover and translations between semantic concepts on the client and server sides.

These are some examples of how semantic web services are incorporating ontologies into their methods for discovery and communications. Our research is not tied to these technologies, however, and is flexible enough to work with emerging types of semantic web services that incorporate ontologies.

2.3 Ontology Reduction

Seidenberg & Rector demonstrated that very large ontologies are difficult to process and reason over [42]. In the database world, large data sets are filtered down into smaller sets through views. There has been some research into creating a database type "view" for ontologies (for example, [21][22] [33][50][46]). This idea treats the full ontology as a database and queries for one or more concepts of interest (COI).

To help solve this problem, Jimenez-Ruiz et al. [22] built a Protege based query language, called OntoPath. Much like a database query, OntoPath allows users to query an ontology for concepts and relationships to retrieve an ontology view, or "personalized modules." An OWL ontology is stored in a database where OntoPath can query it much like a traditional database. While OntoPath's XPath based query language is relatively easy to use, it requires the user to understand more about the ontology being queried than a typical semantic web user or agent would understand.

Volz et al. [50] acknowledge that the semantic web must support the "needs of specific user communities" and create particular information for them. To fill this need, they created a "view language." This view language is designed to show users a view of

an ontology based on a query or set of queries in an extension of the RDF Query Language (RQL). The views they create are designed to be indistinguishable from the source ontology by the agent. However, they do require the use of an ontology to describe their view language to a user. Updates of these views are also limited and the semantic characterizations of the views cannot be computed automatically. While this is a step in the direction of dynamic reduced ontologies, it still requires the user to build specific queries to get the view they want and does not say anything about the quality of the view created. The burden is placed on the user agent.

In order to limit the information a user has to process when interacting with large ontologies, Noy & Musen [33] created a traversal algorithm. This algorithm builds a "traversal view" from a COI or set of COIs by following property resources out a set distance defined by the user. They define a view built by a query, such as the Jimnez-Ruiz et al. [22] OntoPath method as an ontology view. However, they realize that this view does not allow a user to see all concepts closely related to their COI. They propose a traversal view to complement the query views to serve this need. Again, although this is a step in the direction of a dynamic algorithm to create reduced size ontologies, it is still targeted toward users interacting with a tool, such as Protégé, to access information contained within a classifying ontology, such as a genome.

There has also been research into creating modular rather than monolithic ontologies that could be merged by the agent when they are needed. Stuckenschmidt & Klien [46] defined an architecture to create modular ontologies during the design and development phase. Their goal is to create modular ontologies with loose coupling, self-containment, and integrity. The architecture uses queries between modules to group the

modular ontologies together when needed. This allows reasoning across multiple modules through these groupings. A necessary piece of this architecture is a method for detecting changes in modules that affect a grouping. However, these modules are defined at development time and do not address the specific needs of an agent dynamically.

Kusnierczyk [23] looked at partitioning the Gene Ontology by using the Taxonomy of Species to define how to automatically generate the slimmed modules. The framework relates concepts in the Gene Ontology to the Taxonomy of Species through validity, specificity, and relevance.

Grau et al. [16] developed a method to extract reusable fragments, or modules, from ontologies based on a given set of terms. They show that minimal module extraction is not algorithmically solvable. Their syntactic locality algorithm was compared with Noy & Musen's [32] PROMPT-FACTOR algorithm and Grau et al.'s Modularization Algorithm [15]. Grau et al. (2007) [16] found in every case, their locality algorithm created smaller modules than either of the other two algorithms using a series of complex and simple ontologies . This work is similar to ours in that Grau et al. [16] wanted to create fragments of large ontologies. It is, however, still a static model designed to create reusable modules rather than a dynamic method, such as our method, for creating reduced ontologies. However, their success with syntactic locality gives confidence that a dynamic locality based reduced ontology algorithm such as the algorithm we use in our method is a reasonable approach.

2.4 Ontology Metrics

Metrics are widely used in software engineering and information retrieval to judge quality. Ideas such as cohesion, coupling, complexity, recall, and precision are all familiar topics in these fields [10] [27]. Metrics have recently been applied to ontologies as well. Ontology metrics have generally been divided up into structural, semantic, and hybrid, each of which we will discuss below.

2.4.1 Structural Metrics

The most natural metrics to measure any system are structural metrics. These are measures of relationships between object and properties. This class of metrics is similar to what has been traditionally used to measure software quality.

In 1999 Menzies [57] looked at using structural metrics to show that prior knowledge can help build new Knowledge bases. They defined the metric *supportablity, s,* which measures the number of terms in an ontology, *k,* to be reused divided by the number of terms needed in the knowledge base (KB), *n*. They then took two systems, SRI and Cyc/Tek which were posed with three sets of questions, sample questions (SQ), test questions a and b (TQA & TQB). They then measured how much support the Upper Ontology (UO), and the previous system (Cyc) offered both systems over their evolution. What they found was that reuse of prior systems was useful in understanding the questions. The more general UO, however, offered less use to the engineers who were not familar with it. Prior knowledge seemed to be less useful for encoding axioms, but this was inconclusive in their study [57].

Hsi et al. [58] in 2003 looked at finding the core concepts of an ontology based application. They examined five social network metrics as applied to ontologies. *Degree*

Centrality measures the number of edges on a node. *Closeness Centrality* measures the average distance from a node to all other nodes. *Betweeness Centrality* measures the number of shorts paths between all pairs of nodes that include a particular node. *Informational Centrality* measures the information contained in all paths originating with a specific node. Finally, *Eigenvector Centrality* measures the centrality of a node relative to the importance of its surrounding nodes. They then compared the ontology models for three applications. They found some metrics were sensitive to the structure of the ontology, specifically *Degree Centrality* was largely affected by how many subtypes an ontology had, and *closeness* and *information centrality* did not change significantly over all the nodes in the ontologies. They found that they could identify like core concepts by looking at the slope of the metrics as it changed across nodes. A large jump shows a movement from peripheral nodes to core nodes. They believed these metrics could be useful in software maintenance.

In 2004, Lozano-Tello & Gomez-Perez [26] developed a multilevel framework of 160 characteristics. They further categorized these characteristics into five domains including *content, language, methodology, tools* and *costs*. Rather than confine themselves to one ontology language, they build a multilevel tree of characteristics (MTC) based on a reference ontology, which includes generic items, such as ontology, class, attribute, instance, relation and axiom. Their characteristics were validated by ten ontology experts through questionnaires. Then they used these characteristics to create a tool, ONTOMETRIC, to measure the quality and usefulness of an ontology relative to the system needs. ONTOMETRIC is a semi-automated system that works as a

decision aid to help developers choose the right ontology given the requirements for the system.

Sabou et al. [59] looked at an automated extraction tool to build domain ontologies for web service from the service documentation and a generic web-service description ontology, such as OWL-S. They used six metrics to answer the following questions:

1. What is the performance of the learning algorithm?
2. Is the ontology a good basis for ontology building?
3. Does the ontology cover the analyzed domain?
4. Does the ontology support a certain task?

They used metrics to find the answer to each of these questions. *Lexical Recall (LR)* and *Lexical Precision (LP)* were used to measure how well the learning algorithm correctly identified the constructs needed for the domain ontology. *Ontology Precision (OP)* is the ratio between concepts found by the algorithm that were useful to experts to all concepts found. *Lexical Overlap (LO)* measured the number of correct concepts found versus all the concepts found by experts. *Ontological Improvement (OI)* measured the ratio of the concepts useful to experts that were not indentified manually to all concepts found by experts. *Ontological Loss (OL)* measured concepts identified by experts that the algorithm missed versus all the concepts found by experts. They found that the performance of their learning algorithm was good with 82% for LR and 91% for LP. They had high values for OP, but they found from expert analysis that a "clean ontology" was not necessary. Often expressive information was helpful in domain insight for the developers. LO and OL were very bad (7% & 93%) due to concepts that

were not listed in the documentation that experts inherently understood. OI was good (56%) mainly due to the fact that experts did not diligently review the concepts to build from. Overall these metrics gave them guidance in how to improve their learning algorithm to support ontology web service development.

Guo et al. [60] looked at performance metrics they integrated into a Semantic Web benchmarking tool, Lehigh University BenchMark (LUBM). This tool was designed to help choose a knowledge based system (KBS) for a large OWL application. This tool included five performance metrics that were measured for fourteen queries performed on a sample KBS (Univ-Bench). *Load Time (LT)* measured the time it to load all required OWL files to the target system and processing time. Their results for LT varied from five seconds to over forty-six hours depending on the number of universities represented. *Repository Size (RS)* measures the size of the repository once the OWL files have been loaded. RS followed LT very closely ranging from 16MB to 1.2GB. *Query Response Time (QRT)* measures the average response time to each query over ten consective calls. QRT ranged from one ms to almost 1,000,000 ms (~sixteen minutes) depending on how many universities were represented and which query was used. *Query Completeness and Soundness (QCS)* measures the completeness of the query answers relative to the knowledge base. Most of the systems performed well for OCS on the easy queries, but the more complicated ones separated the KBS from one another with some systems finding all the answers and others missing them all. Overall they found that it is very important to choose the correct domain ontology for a semantic web service. Scalability is very important, since forty-six hours are generally to long to for a user to wait.

Ceruti et al. [6] looked at ontology merging to support agent based distributed radar tracking. They used two general classes of metrics, general statistics and disjunction metrics. General statistics include metrics such as *number of added ontologies, total concepts*, and *number of redundant concepts removed*. Disjunction metrics include *individual disjunction metric* and *overall disjunction metric*. Disjunction measures the difference in levels a concept is at within the ontologies used in the merging.

In an effort to measure quality of an ontology, Yao et al. [53] applied software engineering based cohesion metrics to ontologies. They defined three metrics, *Number of Roots (NoR), Number of Leafs (NoL),* and *Average Depth of Inheritance Tree - Leaf Nodes (ADIT-LN)* metrics. They evaluated these metrics on a set of ontologies developed by Creative Commons. These ontologies were evaluated by experts for quality. The quality scores were then correlated with the metric values. NOR had a correlation of 0.386, a moderate correlation. NOL and ADIT-LN had correlations > 0.6, a large correlation. They concluded that there is evidence that software engineering metrics could be used to measure quality of an ontology.

Tartir et al. [49] also created a tool for measuring the quality of an ontology, called OntoQA. This tool analyzes the ontology schema and the knowledge bases it describes through a set of metrics. These metrics are defined in two categories, schema metrics and instance metrics. Schema metrics include relationship richness, attribute richness and inheritance richness. Instance metrics include two more categories, knowledge base metrics and class metrics. Class richness, average population and cohesion are included in knowledge base metrics, while importance, fullness, inheritance

richness, relationship richness, connectivity and readability make up the class metrics. They use these OntoQA and these metrics to analyze a number of ontologies.

Continuing on the path of applying software engineering metrics to ontologies, Orme et al. [35] applied coupling metrics to ontologies. Coupling measures the dependency of an ontology on other ontologies. They defined three metrics for ontology based systems. These metrics include number of external classes (NEC), reference to external classes (REC), and referenced includes (RI). Using the metric validation framework of Kitchenham et al. [62], they showed that each of these metrics is valid for use in the domain of coupling. They then applied these metrics to 33 ontologies from Creative Commons. They asked 18 knowledge engineering experts to evaluate the ontologies coupling on a scale between 0.0 and 1.0. They measured how well the experts agreed with each other and found a 0.9160 interrater reliability, meaning that there was consistent agreement between the experts. Then, the metric results were compared with the expert analysis and found NEC and REC had a large correlation (0.685 & 0.532) and RI had a moderate correlation (0.481) to the experts. These metrics were applied to a framework to reduce coupling in bioinformatics by integrating ontologies rather than using a layered approach.

Most recently, Orme et al. applied complexity metrics to ontologies [36]. These metrics included NoC, number of Fanouts (NoF) and ADIT-LN. Once again these metrics were highly correlated with expert analyses of the ontologies. We can take from these studies a confidence that the quality of ontologies in general can be measured by metrics.

2.4.2 Semantic Metrics

Vrandecic and Sure extend ontology metrics from purely structural metrics into semantic metrics. They also introduce the concept of normalizing ontologies prior to computing the metrics, which they argue allows them to apply known structural metrics in a semantics-aware way [51]. A "stable metric" refers to a metric that maintains its meaning even as new axioms are added to the ontology. They argue that both of these properties allow them to measure semantic metrics in a dynamic environment.

In the same vein, Guarino and Welty [18] seek to find "analytic metaphysical" measures of ontologies that are valid regardless of the ontology domain or structure. These include ideas of essence, rigidity, identity, and unity. However, this framework deals with individual properties and not with the quality of the ontology of a whole, except perhaps by uniting the metrics for individual axioms.

Alani and Brewster created AKTiveRank, a system for ranking ontologies for their reuse potential [1]. AKTiveRank uses an idea similar to the page rank system of Google. Using the Java Universal Network /Graph framework (JUNG) query system to query the RDF ontol-ogies, AKTiveRank makes four measurements. The first of these metrics is the Class Match Measure (CMM) which measures the coverage of an ontology over a set of terms. Second, the Density measure calculates the information content of a class within an ontology. This includes ideas, such as number of subclasses and number of properties. Thirdly, the Semantic Similarity Measure (SMM) calculates, within an ontology, how close together classes are that match the query. Finally the Betweeness measure calculates the number of shortest paths that pass through each node in the graph. Alani and Brewster compared their results with expert analyses and found that the CMM

matched up best with the experts, but using a weighting of 0.4(CMM), 0.3(BEM), 0.2(SMM), and 0.1(DEM) on the four metrics yielded a total score that was highly correlated with expert opinion [1].

2.4.3 Hybrid Metrics

Gangemi et al. had similar subcategories of ontology metrics to include structural measures, functional measures, and usability-profiling measures. Structural measures are traditional metrics and focus on the "syntax and formal semantics" of an ontology [12]. Functional measures focus on the "intended use of a given ontology and its components" [12]. Finally, usability profiling focuses on annotation metrics of the ontology. They then compare these to logical types, with structural measures viewing the ontology as an information object, functional measures viewing it as a language, and usability-profiling viewing it as a meta-language. According to Gangemi et al., structural metrics include topological properties (e.g., depth, breadth, and fan-out), logical-adequacy properties (e.g., consistency, anonymous classes, and cycle ratios), and meta-logical-adequacy properties (e.g., qualified density). For functional metrics, they match recall, precision, and accuracy from information retrieval to expert analysis to yield metrics including agreement, user-satisfaction, task, topic and modularity. Usability-profiling includes such measures as, presence, amount, completeness, and reliability of ontology annotations. They finally developed three frameworks: oQual, O2, and Qood grid, to validate ontologies based upon these types of measures [11][12][13][14].

2.5 Blackboard Architecture

Blackboard architectures were first used in the HEARSAY-II speech system developed in the early 70s [9]. Nii later defined what was meant by a blackboard architecture [31]. The basic structure of this model is a group of knowledge sources, a single blackboard data structure, and a control which allows the knowledge sources to change based on the contents of the blackboard. Each knowledge source contributes some piece of an overall solution by storing the intermediate solutions on the blackboard [31].

Lalanda et al. [25] adapted the blackboard model to work with multi-agents in a real-time environment. Their model includes a traditional real-time system that accesses the blackboard to perform various calculations. Their blackboard model consists of one agent which acts a control node for the system. This control node evaluates the current environment and makes decisions on how to calculate the solution. It also handles unexpected events that occur during the computation. The control node also contains plans which speed up the decision making process for the heuristic and introduce time restrictions inherent in a real-time system [25]. This model is the basis for the architecture created in this research, although the real-time requirements are relaxed in our architecture.

CHAPTER THREE

OVERALL RESEARCH DESCRIPTION

The overall goal of this research was to develop an algorithm that, given an ontology, could create an reduced ontology (a sub-ontology of the larger original ontology) based on a semantic concept of interest provided by some client. Specifically, we were interested in reducing ontologies used to describe semantic web services for mobile devices.

This research was broken up into three phases, each with its own evaluation/validation. In phase one we created models to dynamically predict the information quality and performance gain of a candidate reduced ontology. In phase two we developed an efficient algorithm to dynamically construct a minimal ontology based on the previously-developed prediction models. Finally we created an evaluation framework and simulation to verify that our algorithm created useable reduced ontologies for mobile devices using currently available networks (these networks included 2G, EDGE, 3G, and Wi-Fi). We examined this algorithm's performance when applied to small and medium ontologies, and finally when applied to a large real world ontology (Word Net).

The details of each of the phases are described in the following sections.

3.1 Phase One

As was previously stated, phase one of this research involves developing and choosing ontology metrics, validating them versus quality and performance gains, and developing models that can be computed dynamically.

To begin phase one, we had to define what was meant by information quality (IQ) and performance gain (PG). Quantifying performance is straightforward, since there are objective measures of download and processing time. We defined performance gain as the ratio of the download and processing time of the candidate reduced ontology to the download and processing time of the original ontology.

Information quality is difficult to define without knowing what the user is searching for. A common way to comparing information content in information retrieval is recall and precision. Recall is defined as the fraction of the concepts that are relevant to the query that are successfully retrieved. Precision is the fraction of the concepts retrieved that are relevant to the user's information need. We use these concepts by executing queries on both the original ontology and the candidate reduced ontology and averaging the scores. In this case, since the reduced ontology is always a subset of the original ontology and contains no concepts not in the original, precision has little value. Therefore, we define information quality as the average recall of the candidate reduced ontology over a set of relevant queries.

Neither of these measures for IQ and PG can be reasonably calculated dynamically, since they are use dependent. Therefore, we chose to create a predicative

model of computable metrics for each measure. We selected a set of ontology metrics defined in the literature and defined a new suite of ontology metrics with reduced ontologies in mind. Once the metrics were chosen, we chose a sample of publically available ontologies in various domains. Next, we validated these ontology metrics against both IQ and PG, employing statistical methods. Using the metrics which best predicted quality and performance, we created models for both quality and performance gain that can be calculated dynamically. This models were then used later (in phase 2) to develop the Dynamic Reduced Ontology Generation (DROG) algorithm.

The following sections discuss each of these steps in detail. Section 3.1.1 describes the metrics chosen for this work, Section 3.1.2 describes the process used to validate these metrics, and Section 3.1.3 discusses the models.

3.1.1 Metrics

While metrics have been in use for software systems for many years, applying them to ontologies is a relatively new idea. There have been metrics previously defined to describe the structure of an ontology [35][36][37][53]. We will use these earlier ontology metrics as well as five new ontology metrics designed for this study.

The five metrics we have defined are divided into two categories of metrics (presented in detail in this proposal): informational and performance. Informational metrics are concerned with keeping as much data related to the COI as possible, and will be correlated with mean average recall. Performance metrics are concerned with both the bandwidth and reasoning time required for the sub-ontology.

3.1.1.1 Previous Metrics

Table 3.1 represents a description of structural metrics defined by Orme, Yao, and Etzkorn [35][36][37][53].

Table 3.1 Previously Defined Ontology Metrics

Metric	Description
NOEC	Number of external classes included in the ontology [35].
REC	References to external classes included in the ontology [35].
RI	Number of references included in the ontology [35].
RC	Number of root classes in the ontology [53].
LC	Number of leaf classes in the ontology [53].
ADIT-LN	Average depth of the inheritance tree in the ontology [53][36].
NoC	Number of classes in the ontology [36].
NoF	Number of fan-outs in the ontology [36].

3.1.1.2 Informational Metrics

This section defines a set of metrics created in this research

3.1.1.2.1 Relevance

An ontology that contains nodes which are not relevant to the COI is requiring the device to both receive and process extraneous data. This is a problem due to resource limitations on mobile devices. Classes which are unreachable from the COI in an ontology reduced ontology are not relevant. An unreachable node is a node where the path length from the COI to that node is infinite.

Relevance to a COI is defined as

$$Relevance_{sg} = 1 - \frac{N_{ur}}{N_{total}}, \quad (3.1)$$

where N_{ur} is the number of nodes whose path from N_{coi} is infinite and N_{total} is all the nodes in the reduced ontology.

If the number of unreachable nodes is equal to the total nodes in the ontology reduced ontology, Relevance = 0.0, and conversely, if there are no unreachable nodes, Relevance is 1.0.

3.1.1.2.2 Completeness

References in an ontology to a node that the mobile device has no access to can cause problems with reasoning about the available web services and require additional bandwidth resources. While it is likely that any ontology reduced ontology would have some references that are not contained internally, it is desirable and expected that most links closely related to the focus of the reduced ontology should be internal.

Completeness is a measure of how many undefined references there are versus the total number of references in the reduced ontology. A large number of undefined references relative to the total number of references implies that the user would likely need an undefined reference during reasoning. Completeness is defined as

$$Completeness_{sg} = 1 - \frac{R_{ur}}{R_{total}}, \qquad (3.2)$$

Where R_{ur} is the number of references that are undefined and R_{total} is all the references in the reduced ontology.

If there are no undefined references, then Completeness will equal 1. If all the references are undefined, then Completeness will equal 0.

3.1.1.2.3 Proximity

Using the rule of proximity, references that are closer to a COI are more closely related to it than a reference that is farther away. While this is not always true, it is a good "rule of thumb" that we can use to describe a reduced ontology.

Based on proximity, we define the Proximity metric to compute the path length for undefined references compared to the maximum possible length they could be. The minimum path length is simply the number of undefined references (assuming they are all reference by the COI) and is defined as

$$PL_{min} = R_{ur}. \tag{3.3}$$

The maximum path length is equal to the path length of the baseline ontology. Allowing us to define Proximity as

$$Proximity_{sg} = 1 - \frac{PL_{ur} - PL_{min}}{PL_{maxl} - PL_{min}}. \tag{3.4}$$

If the path length of the undefined references PL_{ur} equals PL_{min}, then Proximity is 1. If PL_{ur} equals PL_{max} then Proximity equals 0.

3.1.1.3 Performance Metrics

3.1.1.3.1 Compactness

The major goal of building a reduced ontology for mobile devices is to limit the amount of data transfers and processing time required to use the ontology and the web services it identifies. So the sub-ontologies we are analyzing need to be as compact as possible. By the rule of proximity, nodes that are closer to our COI will most likely be more related and, conversely, nodes that are far away are likely to be less related.

In order to define compactness, we first need to define a few intermediate values. The minimum path length (PL_{min}) from a COI of any ontology is given as

$$PL_{min} = \sum_{i=1}^{n} \begin{cases} 1 \; if \; Length(N_{COI,} N_i) < \infty \\ 0 \; otherwise \end{cases}. \quad (3.5)$$

This is true since the most compact an ontology could be is if the COI had a single edge between it and all other nodes in the ontology.

In the case of ontology reduced ontologies, since a reduced ontology can never have a path that doesn't exist in the original graph, the maximum path length is simply equal to the path length of the original ontology.

$$PL_{max} = PL_{org}. \quad (3.6)$$

The total path length of an ontology reduced ontology is defined as

$$PL_{sg} = \sum_{i=1}^{n} Length(N_{coi} N_i). \quad (3.7)$$

where Length($N_{coi,} N_i$) is the length of the path between the COI and node *i* such that $N_i \epsilon$ Nodes Reachable from COI.

The compactness of an ontology reduced ontology based on the COI is defined as

$$Compactness_{sg} = 1 - \frac{(PL_{sg} - PL_{min})}{(PL_{max} - PL_{min})}. \quad (3.8)$$

This yields a 1.0 if $PL_{sg} = PL_{min}$ and 0.0 if $PL_{sg} = PL_{max}$

3.1.1.3.2 Size

The final metric is size. With all other things being equal, it is more desirable to have a smaller reduced ontology to improve reasoning and download performance on a mobile device. Size is a ratio to the original ontology size and is simply defined as

$$Size_{sg} = 1 - \frac{Num_{sg}}{Num_{original}}. \quad (3.9)$$

Therefore, a reduced ontology that is identical to the original graph would have a Size of 0 and an ontology that is much smaller would have Size approach 1.

3.1.1.4 Validation of Metrics

In order to tell whether or not the metrics are valid, they must be validated against some measure. The two measures relevant to this research are informational quality, or the amount of information relative to the concept of interest retained, and performance gain, or the decrease in bandwidth and processing required.

3.1.1.4.1 Correlation versus IQ

IQ is a somewhat subjective idea. Previous work has developed various methods to create subsets or views of an ontology. However, the only attempt to validate the ontologies was with Gangemi et al., who matched information retrieval with expert opinions [14]. In order to develop a dynamic model, we need a computable measure of quality. In order to validate our metrics, we must first define what is meant by quality.

In this research we chose to define the quality of an ontology reduced ontology in terms of "recall," a concept which is normally used in information retrieval. Specifically, we define the recall achieved when applying a particular query to the original, base ontology as our gold standard for quality. Another information retrieval

concept, precision, is not applicable because we are only creating subsets of the base ontology. There is no way more information can be retrieved from the reduced ontology queries than the base query, so precision would not provide any information about the subsets.

Given a concept of interest, we create a series of queries for the base ontology and measure the resulting tables. We then run those same queries on the reduced ontologies and take the ratio of those tables with the base tables. This is defined as recall [27]. We take the average of the recall scores to compute information quality (IQ). IQ varies between 0 and 1 and is our measure of quality.

Each of the metrics defined in this chapter is then correlated with the IQ score. This correlation provides a filter on which metrics are actually useful at predicting quality. This will provide confidence in which metrics are useful to quality and which may not be. The results on this correlation are discussed in Chapter Four.

3.1.1.4.2 Correlation versus Performance

Performance is less subjective, but a still a broad term. There have been some studies into the performance of subsets versus full ontologies [16]. However, performance has typically been defined as simple size and has not been related to any metrics. We must first define what is meant by performance.

We choose to define performance gain in two parts, bandwidth and processing time. As with quality, the original ontology forms the basis for performance measurement. We measure the bandwidth requirements and processing time of the base ontology and then for the candidate ontology reduced ontology. The ratio of bandwidths and processing time form the Performance Gain (PG).

In order to make these measurements, a simulation of a mobile device, with various processing capability and bandwidths, will be created. This simulation will interact with a mock semantic web service provider. A series of Monte Carlo runs with capabilities of the device being varied will be used to compute average bandwidth and processing time required. These measurements will form the PG score.

Each of the metrics defined in this chapter is then correlated with the PG score. This correlation provides a filter on which metrics are actually useful at predicting performance. This will provide confidence in which metrics are useful to performance and which may not be.

3.1.1.5 Predictive Model

Once the metrics are correlated with quality and performance, a model is needed. This model will serve as the basis for the algorithm developed in phase 2. At the end of this phase, we used linear regression to build predictive models of both IQ and PG.

Linear regression is a well understood way to finding linear models between observed data and responses. However, it does suffer from sensitivity to intercorrelation, so the first step in creating this model is finding and removing metrics with intercorrelation. Once the metrics used in the regression were decided, a step-wise regression was performed to find out which metrics provided the best model.

Once the model was found, standard regression analysis was performed. As a method of verifying the model, data not used to build the model was used to validate the predictions the model makes. Our work detailing Phase one is fully described in Chapter Four.

3.2 Phase Two

Phase two of this research involved taking the predictive model for PG and IQ and developing an algorithm that dynamically creates a reduced ontology given a user's concept of interest (COI) and original ontology.

To begin phase one, we had to develop the basis of our algorithm. We then had to find the best way to apply the predictive model to score the current candidate reduced ontology for both PG and IQ. Since these two values are inversely related, a balance needed to be found. If we only used IQ, the original ontology would be returned with perfect IQ score. If only PG was used, we would return a single concept, likely useless to the client.

Once the metrics had been shown to predict IQ and PG, an algorithm to exploit this relationship was developed. Based on the work of Noy & Musen [33] and Grau et al. [16], we chose an algorithm framework based on traversal of syntactic locality.

Our plan to develop our Dynamic Reduced Ontology Generation (DROG) algorithm started with the node(s) semantically related to the COI. It then added nodes that gave the largest increase in IQ. It continued to add a node to the candidate reduced ontology each pass of the algorithm. At each possible addition to the reduced ontology, it will be decided if the increase in the quality score is enough to justify the decrease in the performance score. The algorithm will continue until told to stop by the controller or until it processes all nodes in the original ontology.

The initial thought was to use a brute force method to check the IQ and PG changes for each possible node. This proved to be impractical for even small ontologies due to time constraints. A way to quickly rank possible nodes to limit the number we had

to check was needed. We first considered locality, or path length to the COI node, but it took a large number of rounds to get to a good answer. This required extra overhead time as well as generating a sub-optimal PG scores. Next, we used degree centrality of a node, on the premise that adding highly connected nodes would lead to better decisions earlier. This gave better answers than locality and came close to the same answer as the brute force method. Finally, we tried a wholesale approach that added multiple nodes, each *k* distance from the COI, at each round. This gave a tremendous boost to performance, but overshot the best answer. In the end, we settled on a hybrid approach which combined the wholesale method with the degree centrality to produce a fast algorithm that gave similar answers to the brute force method. This work is detailed in Chapter Five.

3.3 Evaluation Plan

In order to evaluate how the DROG algorithm actually works when employed by mobile devices, we built a blackboard framework as a proof-of-concept semantic web service. Within the blackboard framework, we simulated an actual semantic web service using the DROG algorithm and getting requests from a client. This system accepted as input the concept of interest (COI) and executed the DROG algorithm, with the controlling taking the final reduced ontology and returning it to the client.

3.3.1 Blackboard Architecture

The blackboard architecture model has been around for almost thirty years. Our version of the blackboard architecture is based on a relaxed version of the Landa real-

time model [29], which includes a controller, blackboard data structure, and algorithms. The conceptual view is given in Figure 3.1.

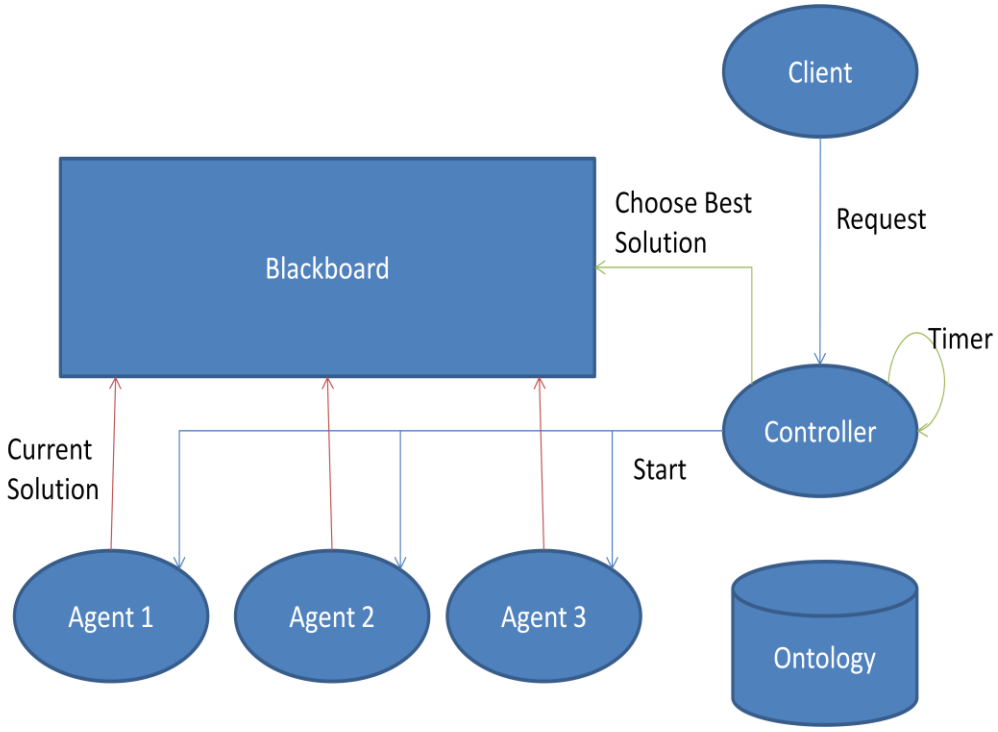

Figure 3.1 Conceptual Architecture

The client sends a request to the controller. This includes a concept of interest and a maximum wait time. The controller then starts the agents to work producing a reduced ontology and sets a time limit. Each agent operates on the full ontology with a specific algorithm. Periodically, the agents will write their newest solution to the blackboard, unaware of the time constraints. When the timer goes off, the controller stops work and chooses the best solution from the proposed solutions on the blackboard, based on the quality score we have defined. This reduced ontology is then sent to back to the client to perform reasoning and complete whatever task it had.

3.3.2 Evaluation Plan

Using this blackboard architecture, we first developed a simulated web server. In our simulation, a controller handled time constraints and chose the best solution available when a set time completed. In this simulation, we used a wide variety of ontologies, from small to very large. Second, we executed Monte Carlo runs of the simulation using RIM's Blackberry simulation for PC across currently available networks (these networks included 2G, EDGE, 3G, and Wi-Fi) for the small and medium sized ontologies used in previous experiments. The RIM's Blackberry simulation for PC that we employed is the Blackberry smart phone simulation, available from RIM's SDK [54]. For each ontology, we used a sample request and generated a reduced ontology which was then downloaded and queried by the smart phone. Finally we chose a large, publicly available ontology, Word Net, and executed it in the same environment. This last piece is a more representative case of where DROG would provide benefit in a real world application.

CHAPTER FOUR

A MODEL TO PREDICT INFORMATION QUALITY AND PERFORMANCE GAIN OF A REDUCED ONTOLOGY

This, the first phase of our research, was completed in three experiments. Two preliminary experiments gave us confidence on which metrics make sense for the predictive model. The final experiment used twenty ontologies in four domains to create a general model for predicting information quality (IQ) and performance gain (PG).

4.1 Preliminary Experiment #1

The first preliminary experiment was a small pilot study to make sure the research concept made sense. It used a single ontology and a subset of the metrics we ultimately used to create the predictive model.

4.1.1 Research Description

In this study, we selected a typical web service ontology available on the internet. A modified traversal algorithm based on that of Noy & Musen [33] was then coded in

Java and run on the ontology with each node as the COI and a cutoff value, $PL_{threshold}$, for the maximum length of any path with the COI as the source. (This modified traversal algorithm is described in detail in Section 4.1.1.3 below.) This generates N candidate reduced ontologies, where N is the number of nodes in the original ontology.

Seven queries were applied to both the original ontology and each reduced ontology. The results from the original ontology form the baseline for computing recall. Next, recall was computed for each reduced ontology and averaged over all seven queries, yielding the mean average recall (IQ) for each ontology reduced ontology. The IQ was used as the information quality (IQ) measure for the reduced ontologies.

This study was not concerned with the quality of the original ontology. It is assumed to be the "gold standard" and used as a baseline for the sub-ontology metrics we will use. We defined the primary indication of a good reduced ontology as that it gives the same answer to a query as would the original ontology.

For the purpose of this study, we defined recall as the fraction of the elements returned by the query relative to the elements returned for the original ontology. IQ was then defined as the average recall of all the queries for a given reduced ontology.

Likewise, precision was defined as the fraction of the elements returned by the query contained within the elements returned for the original ontology. Since each reduced ontology can only contain a sub-set of the information in the original ontology, by definition, precision does not make sense in this case.

The informational and performance metrics defined in Chapter Three were then applied to each reduced ontology. They were then correlated to the IQ score.

4.1.1.1 Ontology and Concept of Interest Selection

We selected a travel agency ontology based on OWL, with seventy-two named classes, to analyze. Flight reservations were chosen as a sample request a mobile device's agent might make. Based on this, "Flight" was chosen as the COI for metrics and seven queries were created by the authors that represent information a mobile device's agent might need to book an airline flight.

After generating each candidate reduced ontology and computing its IQ score, the metrics defined in Chapter Three were computed for each reduced ontology based upon the single COI, "Flight". Finally we correlated each metric's values with the IQ score in an effort to predict if a reduced ontology contains enough information to satisfy a web service request from a mobile device.

4.1.1.2 Creating the Reduced Ontologies with the Traversal Algorithm

The modified traversal algorithm was used to create a large number of reduced ontologies in a short time [33]. We did not claim this was the only way, or even the best way, to create reduced ontologies, but it was an efficient and mechanical way to create a large number of candidate reduced ontologies. The traversal algorithm starts at a selected COI node and follows all of its references (property, parents, and children). Each new node is added to the reduced ontology and the traversal algorithm is recursively called with that node as the COI. This process continues until either there are no more new nodes referenced or the path length passes $PL_{threshold}$. Child classes are the same as their parents and thus hold all the parents' references as their own. For that reason the is-a relationship is traversed, but the reference is not counted toward $PL_{threshold}$.

Each class defined in the travel agency ontology was selected as the COI for the traversal algorithm and a calculation was run with a threshold path length of two. Thus, from the 72 node original travel agency ontology, 72 reduced ontologies were created, each one with the COI set to a unique node in the original ontology.

4.1.1.3 Querying the Reduced Ontologies

The querying of both the travel agency ontology and all the reduced ontologies was performed using SPARQL [52] after loading each ontology into the Jena OWL parser [63]. A set of queries was applied to all 73 ontologies (the original and all the reduced ontologies).

The seven queries used for this pilot study were as follows:

- Get all flights
- Get all flights to New York and the return flights
- Get all flights departing in the August
- Get all flights leaving Athens, Greece
- Get all flights on a Boeing-747
- Get all flights out of Athens with at least five seats available
- Get all flights with a connection

The recall score for each query was computed and the IQ for the reduced ontology was computed over all seven queries. As we discussed earlier, we were only concerned with recall with respect to the original ontology.

4.1.1.4 Applying the Metrics

The metrics were applied to each reduced ontology and the original ontology. However, any reduced ontology that did not contain the COI "Flight" node, could not

have the metrics applied and was therefore tossed out. For this study, COIs are limited to a single choice of a node in the ontology. Since Jena provides an indexed list of the nodes in the ontology, checking for a single node is a trivial task.

If this algorithm was creating reduced ontologies for use with software agents, any possible ontology generated would contain the COI, due to the nature of the traversal algorithm. Therefore, it seems reasonable to ignore reduced ontologies that do not contain the COI in this study.

We then computed the metrics defined in Chapter Three on each of the remaining reduced ontologies. Each metric was automatically computed by a tool we developed in Java based on the Jena OWL parser.

Finally, we correlated the information metrics for each reduced ontology with the IQ scores described in Section 4.1.3.2 using the MiniTab statistical software package [ref]. The results of the correlation are in Section 4.1.4.

4.1.1.5 Ontology Metrics

While metrics have been in use for software systems for many years, applying them to ontologies is a relatively new idea. There have been metrics to describe the structure of an ontology [1][7]. While these may be useful for measuring the quality of an ontology overall, they aren't based on a particular concept that a mobile device may be looking for within the ontology. For this paper, we define metrics to describe properties of a reduced ontology relative to the full ontology and a particular COI.

There were two categories of metrics presented in Chapter Three, informational and performance. Informational metrics were concerned with keeping as much data

related to the COI as possible, and were correlated with IQ. Performance metrics were concerned with both the bandwidth and reasoning time required for the sub-ontology.

The metrics used were defined as equations 3.1 – 3.9 in Chapter Three.

4.1.2 Results

Using the modified traversal algorithm on the travel agency web service ontology, we generated seventy-two reduced ontologies using each OWL class as a starting point.

We ran preliminary tests on the travel agency ontology for $PL_{threshold}$ equal to one, two, and three. For $PL_{threshold} = 1$, the reduced ontologies did not contain enough information to process. The mean average recall score was 0 for almost all the reduced ontologies (with the exception of the one that actually began with the COI, flight).

When $PL_{threshold} = 3$, most of the reduced ontologies were virtually equal in size and structure to the original travel agency ontology. So for this study, $PL_{threshold} = 2$ was selected.

4.1.2.1 Reduced Ontologies

For $PL_{threshold} = 2$, twenty-two reduced ontologies were valid (i.e., contained the COI Flight). This means fifty-two reduced ontologies could not have the metrics calculated and were dropped. It should be noted, though that all fifty dropped reduced ontologies had information recall scores of 0.

4.1.2.2 Information Quality

Each reduced ontology had the standard seven queries applied to it. The mean average recall (IQ) score across all queries was computed and is show in Table 4.1 below.

Table 4.1 IQ for each Reduced Ontology

Reduced ontology Center	IQ
Airplane	1.000
AirTransportationMeans	1.000
BudgetPackage	0.461
Bus	0.461
BusOnly	1.000
FamilyPackage	0.461
Flight	1.000
FlightOnly	1.000
LandTransportationMeans	0.461
PackageToSamos	0.461
PublicTransportationMeans	1.000
Route	1.000
SeaTransportationMeans	0.461
Ship	0.461
ShipOnly	1.000
SpaPackage	0.461
Train	0.461
TrainOnly	1.000
TransportationMeans	1.000
TravelPackage	0.461
TravelPath	1.000
WinterPackageSale	0.461

4.1.2.3 Informational Metrics

We applied each of the informational metrics to the twenty-two reduced ontologies and correlated them with the information recall score given in Section 4.1.2.2. The metric score is shown as a percentage for readability. We performed a test for normality on each metric and the recall and none were normal. Therefore, we used Spearman's rank correlation. We used Cohen's [5] correlation magnitude scale (absolute value):

- <0.1 trivial
- 0.1-0.3 minor
- 0.3-0.5 moderate

- 0.5-0.7 large
- 0.7-0.9 very large
- 0.9-1.0 almost perfect

4.1.2.3.1 Relevance

Table 4.2 gives the relevance for each reduced ontology. Notice that Flight and Route have the highest relevance score.

Table 4.2 Relevance Metric for each Reduced Ontology

Reduced ontology Center	Relevance
Airplane	0.812
AirTransportationMeans	0.812
BudgetPackage	0.049
Bus	0.812
BusOnly	0.782
FamilyPackage	0.049
Flight	1.000
FlightOnly	0.782
LandTransportationMeans	0.823
PackageToSamos	0.049
PublicTransportationMeans	0.857
Route	0.933
SeaTransportationMeans	0.812
Ship	0.812
ShipOnly	0.782
SpaPackage	0.049
Train	0.812
TrainOnly	0.782
TransportationMeans	0.750
TravelPackage	0.046
TravelPath	0.692
WinterPackageSale	0.049

We correlated Relevance with the IQ. The result was 0.623 p=.002, a large correlation that is significant.

4.1.2.3.2 Completeness

Table 4.3 gives the completeness for each reduced ontology.

Table 4.3 Completeness Metric for each Reduced Ontology

Reduced ontology Center	Completeness
Airplane	0.639
AirTransportationMeans	0.639
BudgetPackage	0.301
Bus	0.639
BusOnly	0.622
FamilyPackage	0.301
Flight	0.800
FlightOnly	0.622
LandTransportationMeans	0.649
PackageToSamos	0.301
PublicTransportationMeans	0.683
Route	0.767
SeaTransportationMeans	0.639
Ship	0.639
ShipOnly	0.622
SpaPackage	0.301
Train	0.639
TrainOnly	0.622
TransportationMeans	0.636
TravelPackage	0.291
TravelPath	0.583
WinterPackageSale	0.301

We correlated completeness with the IQ. The result was 0.101 p=.646, a minor correlation that is not significant. We cannot say much about completeness from this result.

4.1.2.3.3 Proximity

Table 4.4 gives the proximity for each reduced ontology. We correlated Proximity with the IQ. The result was 0.641 p=.001, a large correlation that is statistically significant.

Table 4.4 Proximity Metric for each Reduced Ontology

Reduced ontology Center	Proximity
Airplane	0.633
AirTransportationMeans	0.633
BudgetPackage	0.380
Bus	0.633
BusOnly	0.636
FamilyPackage	0.380
Flight	0.809
FlightOnly	0.636
LandTransportationMeans	0.633
PackageToSamos	0.380
PublicTransportationMeans	0.633
Route	0.809
SeaTransportationMeans	0.633
Ship	0.633
ShipOnly	0.636
SpaPackage	0.380
Train	0.633
TrainOnly	0.636
TransportationMeans	0.633
TravelPackage	0.380
TravelPath	0.636
WinterPackageSale	0.380

4.1.2.4 Performance Metrics

4.1.2.4.1 Compactness

Table 4.5 gives the compactness score for each reduced ontology.

Table 4.5 Compactness Metric for each Reduced Ontology

Reduced ontology Center	Compactness
Airplane	0.948
AirTransportationMeans	0.948
BudgetPackage	1.000
Bus	0.948
BusOnly	0.912
FamilyPackage	1.000
Flight	0.641
FlightOnly	0.912
LandTransportationMeans	0.940
PackageToSamos	1.000
PublicTransportationMeans	0.912
Route	0.641
SeaTransportationMeans	0.948
Ship	0.948
ShipOnly	0.912
SpaPackage	1.000
Train	0.948
TrainOnly	0.912
TransportationMeans	0.912
TravelPackage	1.000
TravelPath	0.912
WinterPackageSale	1.000

4.1.2.4.2 Size

Table 4.6 gives the size score for each reduced ontology.

Table 4.6 Size Metric for each Reduced Ontology

Reduced ontology Center	Size
Airplane	0.780
AirTransportationMeans	0.780
BudgetPackage	0.164
Bus	0.780
BusOnly	0.684
FamilyPackage	0.164
Flight	0.424
FlightOnly	0.684
LandTransportationMeans	0.767
PackageToSamos	0.164
PublicTransportationMeans	0.712
Route	0.383
SeaTransportationMeans	0.780
Ship	0.780
ShipOnly	0.684
SpaPackage	0.164
Train	0.780
TrainOnly	0.684
TransportationMeans	0.671
TravelPackage	0.109
TravelPath	0.643
WinterPackageSale	0.164

We correlated Size with the IQ. The result was -0.283 p=.191, a minor negative correlation that is not statistically significant.

4.1.2.5 Discussion

Two of the three informational metrics showed promise as metrics to validate a reduced ontology for a given COI. Proximity and relevance hold promise, with a large correlation to IQ. Completeness could be useful, but the results from this study are not significant enough to say much about it.

The performance metrics required more careful analysis. Neither had a positive correlation with IQ and compactness actually had a medium negative correlation. This

makes sense, since the larger the reduced ontology is, the more likely it has all of the data in the original ontology. In fact, as can be seen from Tables 4.1, 4.5 and 4.6, the sub-ontologies with the best compactness and size scores have very bad IQ. For instance, the sub-ontology built from TravelPackage had a compactness score of 1.0, while having a IQ of 0.46. This would not have been a good choice for a sub-ontology since it had less than half the "flight" related information the original ontology had and would likely require the mobile device to request the full ontology to get the necessary information.

However, the performance metrics showed promise when taken alongside the informational metrics. Examining the top three sub-ontologies from the informational metrics, Flight, Route, and Public Transportation Means, we found that Public Transportation Means had the best performance metric scores. Notice that each of these sub-ontologies had an IQ of 1.0, so there was no data loss. This indicated that the Public Transportation Means sub-ontology was best choice for the mobile device.

Our results are especially interesting since previous work on ontology views, such as Noy and Musen's traversal algorithm [33], have focused on creating views based on a central COI. However, based on these findings, Flight was not the best center for a reduced ontology of the travel agency ontology for flight related information.

This was a pilot study on one web service in one domain, but more work needed to be done with these metrics across many more ontologies and many more domains if we are to generalize them as sub-ontology quality metrics. This led us to conduct the other two experiments in this chapter.

4.2 Preliminary Experiment #2

The results from preliminary experiment #1 were promising. In preliminary experiment #2 we expanded from one travel agency ontology to four to see if our results were consistent. We also attempted to create a predictive model of IQ and PG in the domain of travel agencies.

4.2.1 Research Description

In this study, we selected a group of four travel agency web service ontologies available on the internet. A modified traversal algorithm created in a preliminary experiment #1 and published in [39] was run on the ontology with each node as the center. The maximum path length for any path from the center was set to 3, based on empirical runs. When key was set to 1 and 2, the Each ontology generated N reduced ontologies, where N is the number of nodes in the original ontologies.

This study is not concerned with the quality of the original ontologies. Rather, they are assumed to be the "gold standard" and used as a baseline for the sub-ontology metrics we will use.

We define the quality of a reduced ontology to be whether or not it gives the same answer to a query as the original ontology. Recall is defined as the fraction of the elements returned by the query when applied to a reduced ontology relative to the elements returned for the same query when applied to the original ontology. Mean Average Recall (IQ) is then defined as the average recall of all the queries over a given reduced ontology.

Seven different queries were run on each of the original ontologies and all of their reduced ontologies. The results from the original ontologies form the baseline for

computing recall. Next, recall was computed for each reduced ontology and averaged over all seven queries, yielding the mean average recall (IQ) for each ontology reduced ontology. This IQ was used as the quality measure.

The informational and performance metrics we defined in Chapter Three were then applied to each reduced ontology. Ontology metrics from Orme, Yao, Etzkorn [35][36][37] and Yao, Orme, and Etzkorn [53] were also computed for each reduced ontology. Each the individual metrics was correlated with the IQ.

Linear regression on the metrics was then computed, using a stepwise regression to decide which variables give the greatest prediction value. Multicollinearity was also checked between each of the metrics and highly correlated metrics were reduced to one variable. From this a model of predicting the quality of an ontology reduced ontology was formed.

4.2.1.2 Ontology and Concept of Interest Selection

We selected four travel agency ontologies, implemented in OWL, with a total of 202 named classes, to analyze in this paper.

Flight reservations were chosen as a sample request a mobile device's agent might make. Based on this, "Flight" was chosen as the COI for metrics and twenty eight total queries were created by four unaffiliated volunteers to represent information a mobile device's agent might need to book an airline flight.

Candidate reduced ontologies were again generated with the modified Noy & Musen traversal algorithm [33] described in preliminary experiment #1. After generating a reduced ontology, it was queried and the recall was computed. Then the recall scores for each query were averaged to get the IQ score. Then metrics defined in Sections 4.1.2

and 4.1.3 of this chapter were computed for each reduced ontology based upon the single COI, “Flight”.

4.2.1.3 Querying the Reduced Ontologies

Four volunteers unrelated to this research were asked to create typical queries they would have about flight reservations. These volunteers had no knowledge of the ontologies being used for this study. These volunteers created twenty-eight queries to use in this study.

The querying of both the travel agency ontology and all the reduced ontologies was performed using SPARQL [15] after loading each ontology into the Jena OWL parser [5]. It should be noted that the queries were written in English and had to be mechanically converted to fit the naming conventions of each original ontology. However, no change to the structure or purpose of the query was done. The set of queries was applied to all 206 ontologies (the originals and all the reduced ontologies).

The recall score for each query was computed and the IQ for the reduced ontology was computed over all the queries. As we discussed earlier, we are only concerned with recall with respect to the original ontology.

4.2.1.4 Applying the Metrics

The metrics were applied to each reduced ontology and the original ontology. For this study, COIs are limited to a single choice of a node in the ontology. Since Jena provides an indexed list of the nodes in the ontology, checking for a single node is a trivial task.

If this algorithm was creating reduced ontologies for use with software agents, any possible ontology generated would contain the COI, due to the nature of the traversal

algorithm. Therefore, it seems reasonable to ignore reduced ontologies that do not contain the COI in this study.

We then computed the metrics defined in Chapter Three on each of the remaining reduced ontologies. Each metric was automatically computed by a tool the authors developed in Java based on the Jena OWL parser.

Next, we correlated the metrics for each reduced ontology with the IQ scores using the MiniTab statistical software package. The results of the correlation are in Section 4.2.2. We again used Cohen's [5] correlation magnitude scale.

Finally, using stepwise regression, we chose the metrics that created the best predictive model of quality.

4.2.2 Results

Using the traversal algorithm, we generated 202 reduced ontologies using each OWL class as a starting point. $PL_{threshold}$ was set to k, based on results seen in [12]. IQ and each metric's results are statistically described in Section 4.2.2.1. Correlation between each of the metric and IQ as well as Multicollinearity is described in Section 4.2.2.2. Finally the linear model is described in Section 4.2.2.3.

4.2.2.1 General Statistics of Measurements

The amount of data in this study makes it impractical to include it all in this paper. Instead, we included a general statistical description, including mean, standard deviation, and range for IQ and each of the metrics.

4.2.2.1.1 Information Quality (IQ)

Each query was executed and IQ was computed over 202 reduced ontologies. It had a range of [0.0, 1.0]. The mean was 0.3721 with a median of 0.4179. The standard deviation was 0.2608.

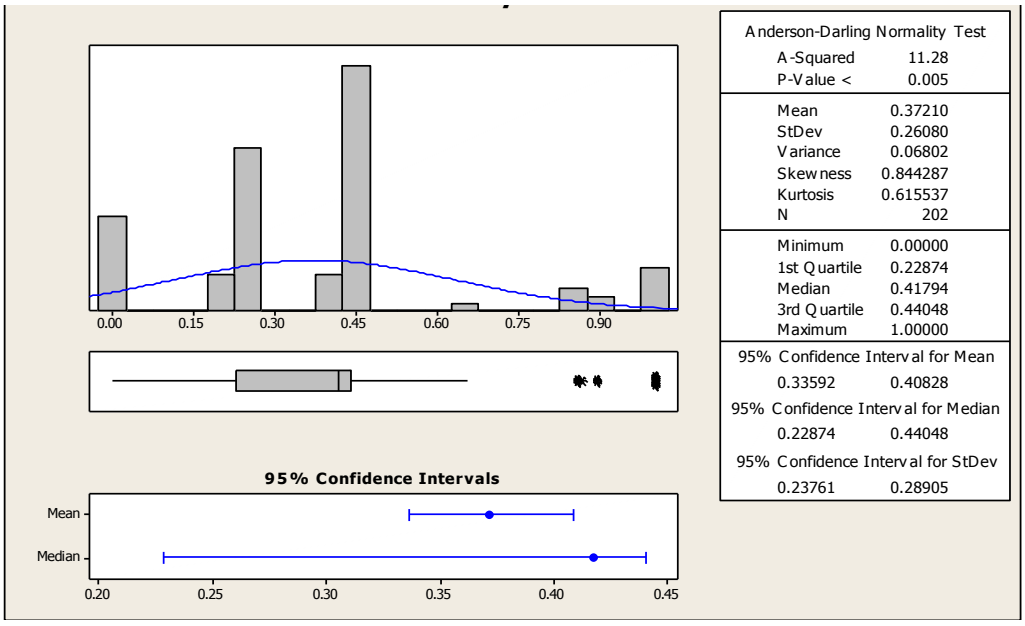

Figure 4.1 Statistical Summary of IQ

4.2.2.1.2. Relevance

Relevance was computed over 202 reduced ontologies. It had a range of [0.0, 1.0]. The mean was 0.1147. The standard deviation was 0.2789.

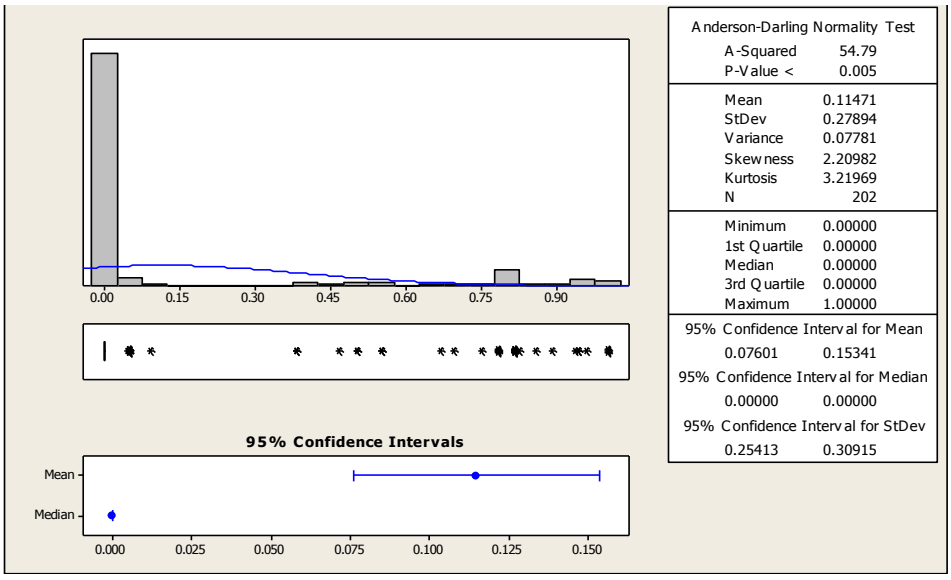

Figure 4.2 Statistical Summary of Relevance

4.2.2.1.3 Completeness

Completeness was computed over 202 reduced ontologies. It had a range of [0.0, 0.9211]. The mean was 0.6320. The standard deviation was 0.1858.

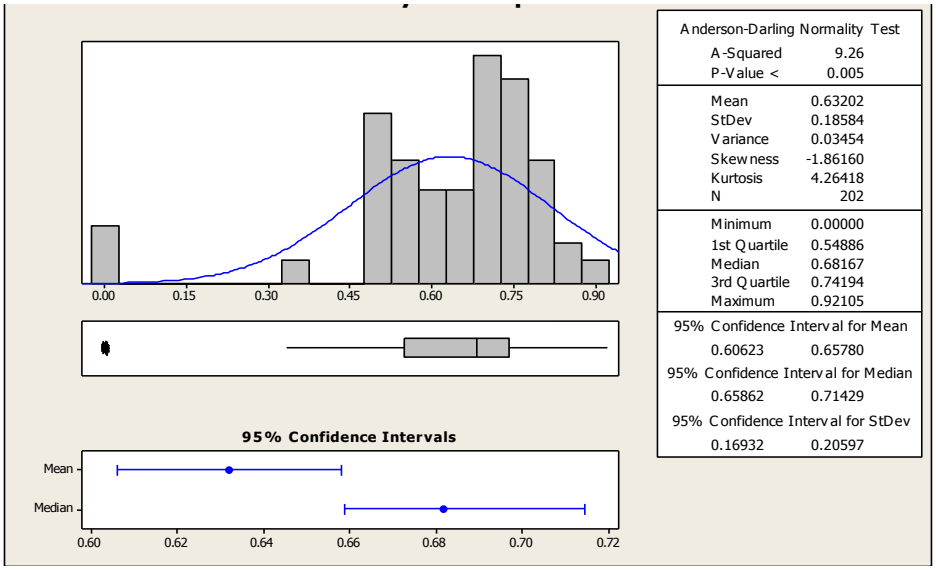

Figure 4.3 Statistical Summary of Completeness

4.2.2.1.4 Proximity

Proximity was computed over 202 reduced ontologies. It had a range of [0.0, 0.9524]. The mean was 0.0964. The standard deviation was 0.2246.

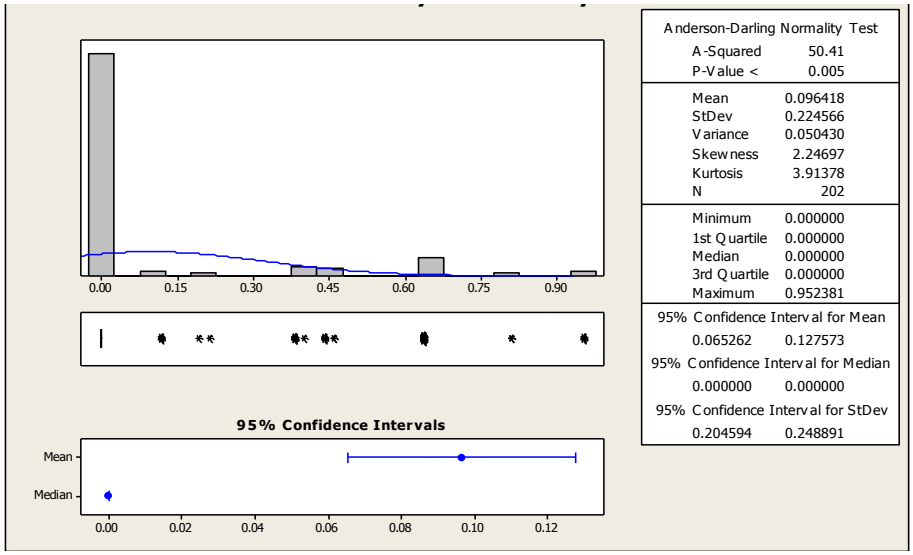

Figure 4.4 Statistical Summary of Proximity

4.2.2.1.5 Compactness

Compactness was computed over 202 reduced ontologies. It had a range of [0.0, 0.9922]. The mean was 0.0233. The standard deviation was 0.3310.

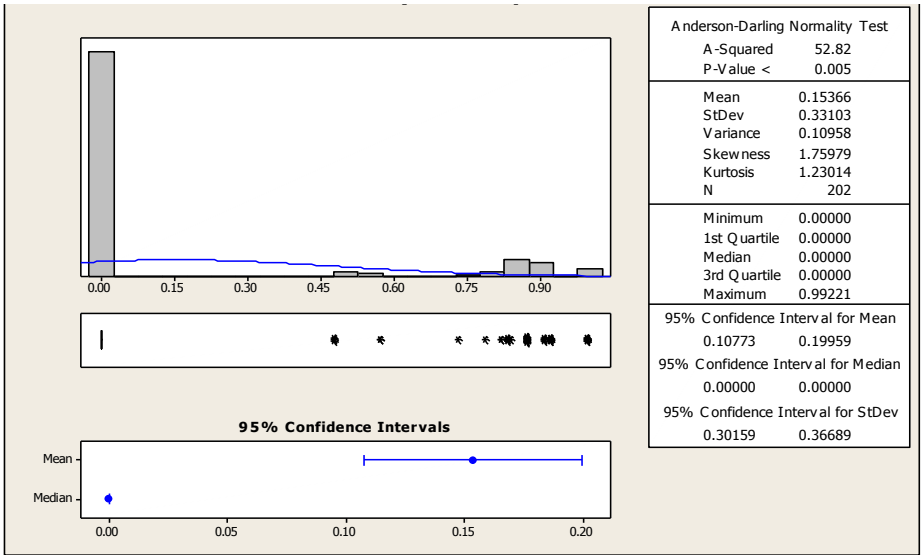

Figure 4.5 Statistical Summary of Compactness

4.2.2.1.6 Size

Size was computed over 202 reduced ontologies. It had a range of [0.0, 0.9529]. The mean was 0.7666. The standard deviation was 0.2081.

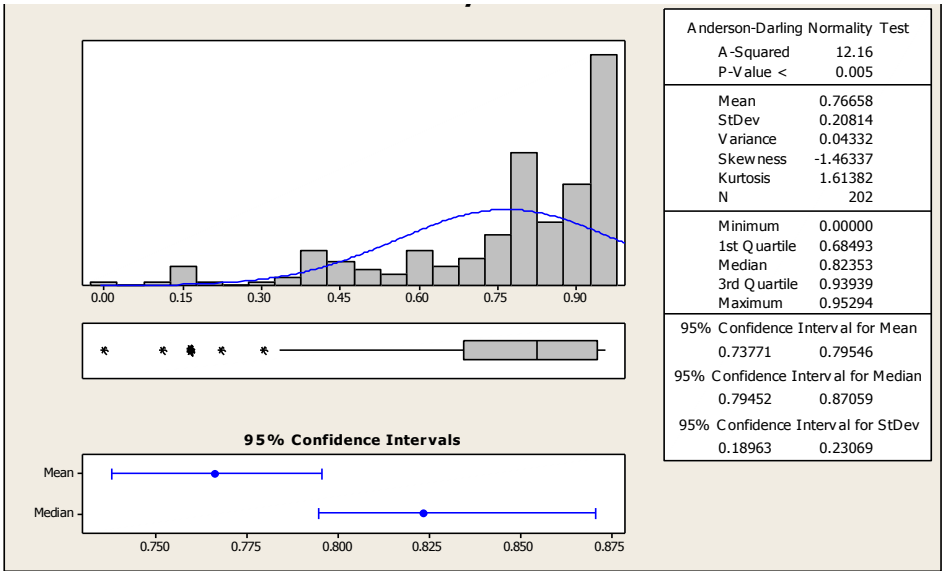

Figure 4.6 Statistical Summary of Size

4.2.2.1.7 NOEC

NOEC was computed over 202 reduced ontologies. It had a range of [0.0, 5.0]. The mean was 2.0149. The standard deviation was 1.3803.

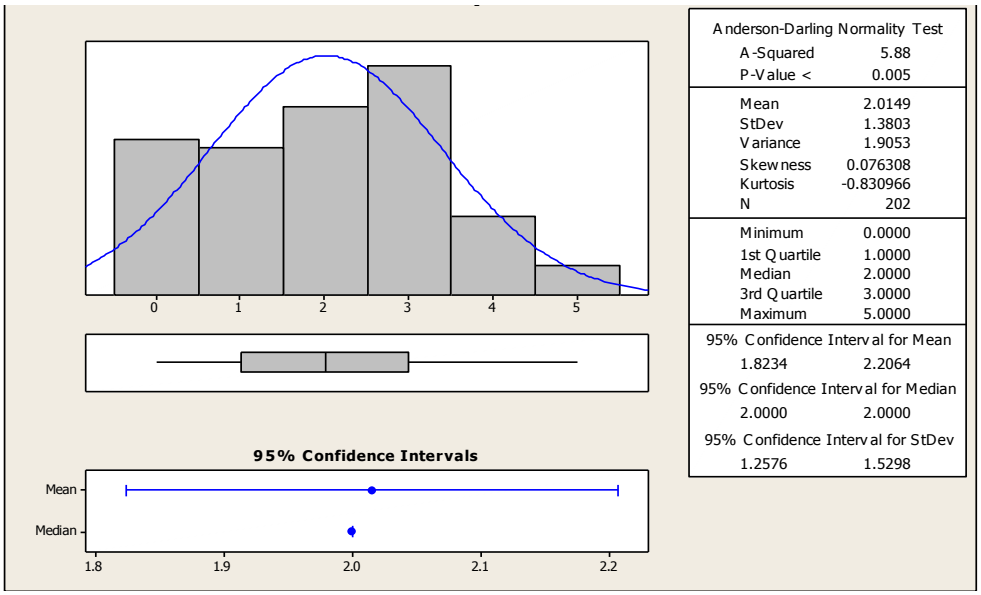

Figure 4.7 Statistical Summary of NOEC

4.2.2.1.8 REC

REC was computed over 202 reduced ontologies. It had a range of [1.0, 99.0]. The mean was 8.901. The standard deviation was 12.394.

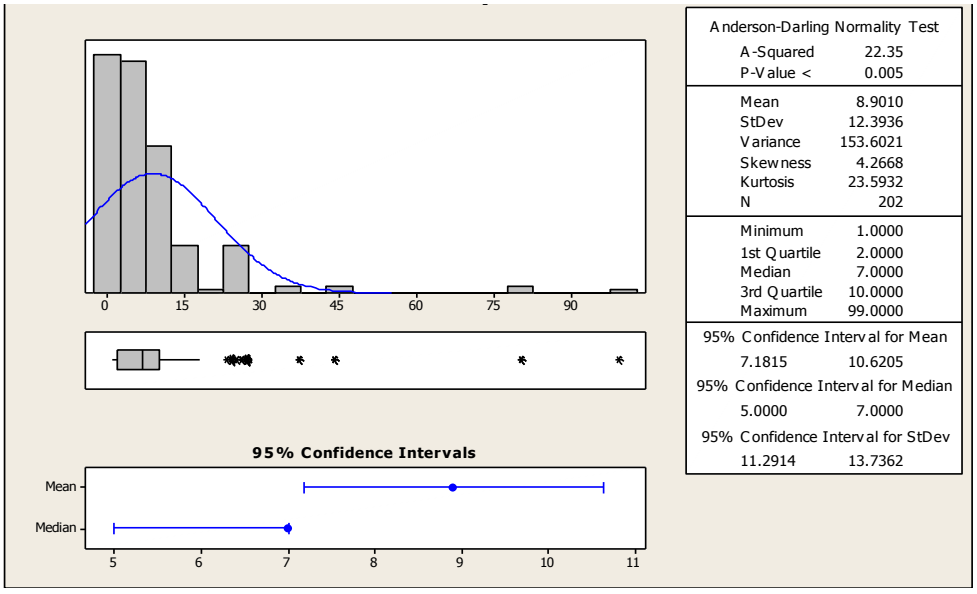

Figure 4.8 Statistical Summary of REC

4.2.2.1.9 RI

RI was computed over 202 reduced ontologies. It had a range of [3.0, 4.0]. The mean was 3.9505. The standard deviation was 0.2175.

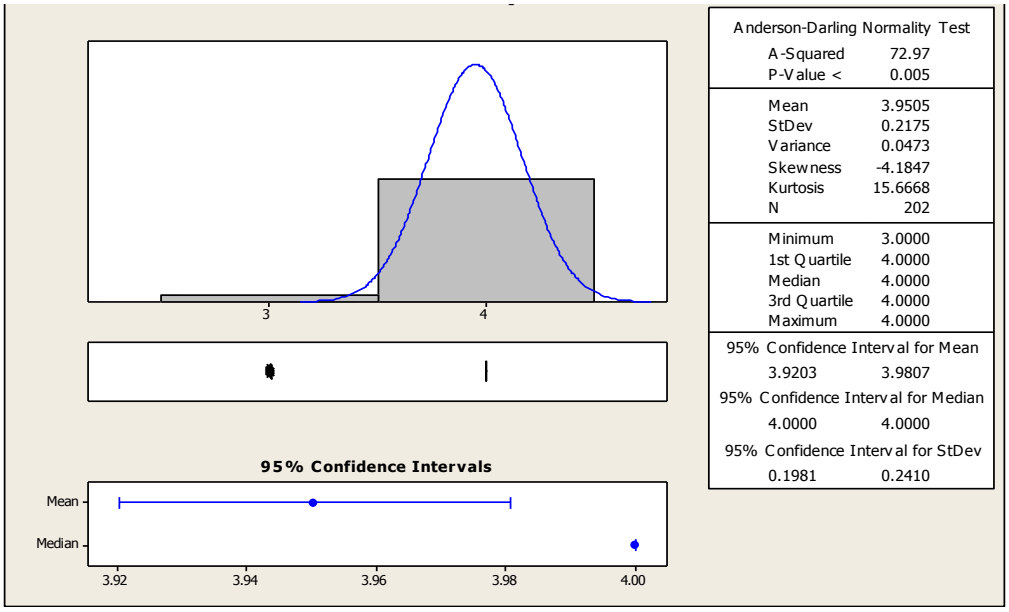

Figure 4.9 Statistical Summary of RI

4.2.2.1.10 RC

RC was computed over 202 reduced ontologies. It had a range of [1.0, 9.0]. The mean was 2.074. The standard deviation was 1.767.

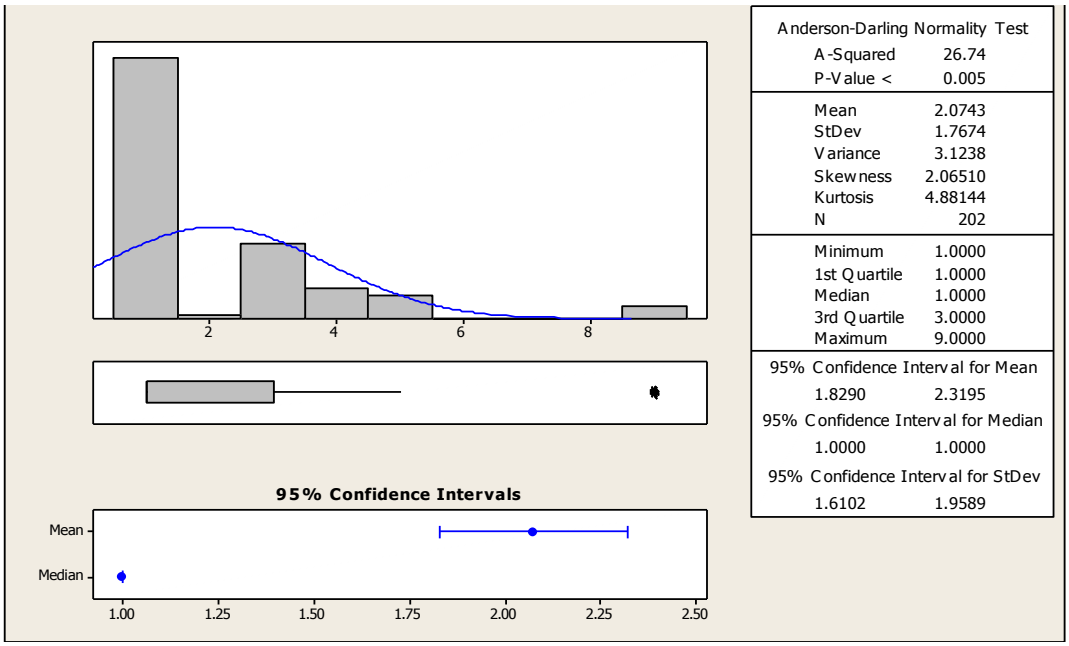

Figure 4.10 Statistical Summary of RC

4.2.2.1.11 LC

REC was computed over 202 reduced ontologies. It had a range of [2.0, 51.0]. The mean was 9.376. The standard deviation was 10.141.

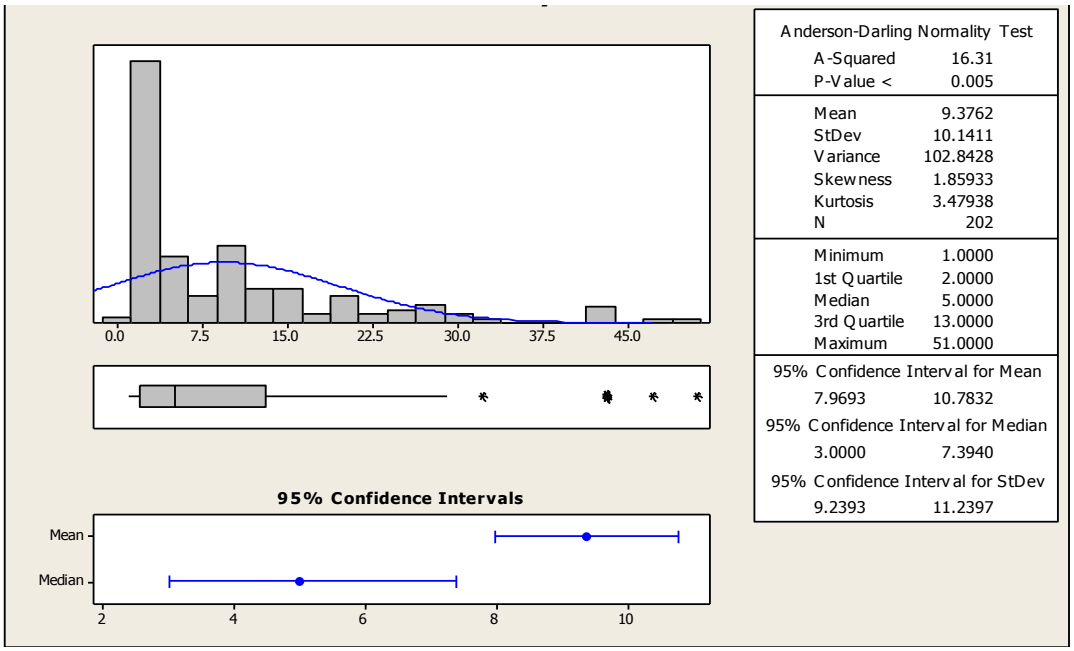

Figure 4.11 Statistical Summary of LC

4.2.2.1.12 ADIT-LN

ADIT-LN was computed over 202 reduced ontologies. It had a range of [1.0, 4.3751]. The mean was 2.3760. The standard deviation was 0.6623.

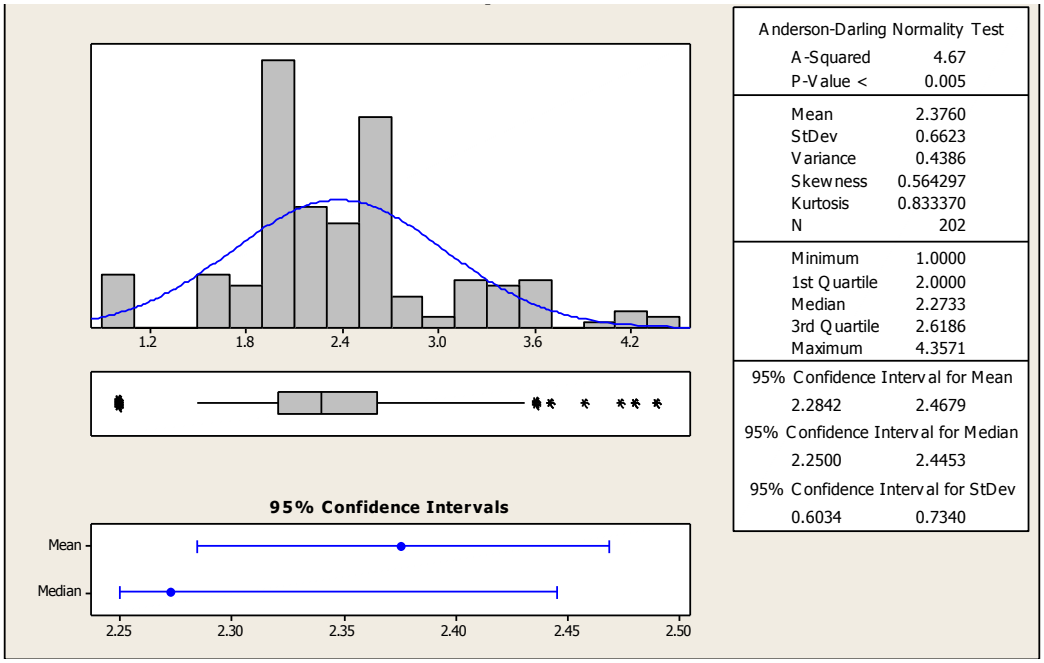

Figure 4.12 Statistical Summary of ADIT-LN

4.2.2.1.13 NoC

NoC was computed over 202 reduced ontologies. It had a range of [2.0, 66.0]. The mean was 15.01. The standard deviation was 14.36.

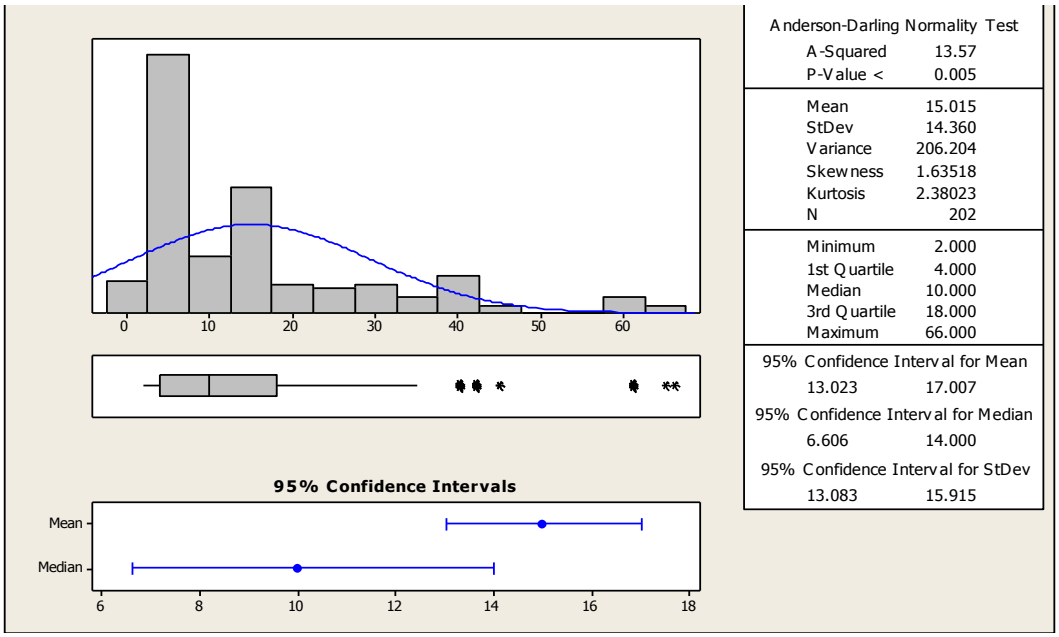

Figure 4.13 Statistical Summary of NoC

4.2.2.1.14 NoF

NoF was computed over 202 reduced ontologies. It had a range of [0.0, 64.0]. The mean was 0.906. The standard deviation was 12.879.

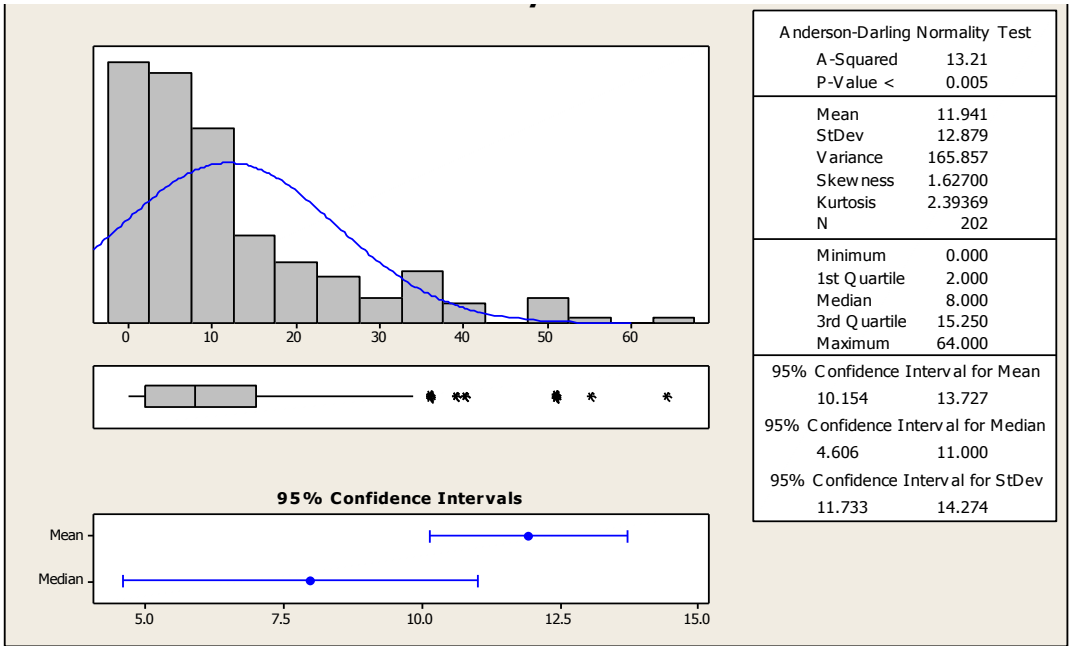

Figure 4.14 Statistical Summary of NoF

4.2.2.2 Correlations

We applied each of the metrics to the 202 reduced ontologies and correlated them with the IQ score. We also correlated them with each other metric to check for Multicollinearity.

4.2.2.2.1 IQ Correlation

Table 4.7 has a list of each metrics correlation to the IQ.

Table 4.7 Correlation Between Metrics and IQ

Metric	Correlation to IQ	Cohen's Magnitude Scale
Relevance	0.717 p<0.0001	Very Large
Completeness	0.398 p<0.0001	Moderate
Proximity	0.681 p<0.0001	Large
Compactness	0.653 p<0.0001	Large
Size	-0.258 p<0.0001	Minor (negative)
NOEC	0.386 p<0.0001	Moderate
REC	0.467 p<0.0001	Moderate
RI	0.326 p<0.0001	Moderate
RC	0.025 p<0.726	Trivial
LC	0.212 p<0.002	Minor
ADIT-LN	0.518 p<0.0001	Large
NoC	0.244 p<0.0001	Minor
NoF	0.269 p<0.0001	Minor

4.2.2.2.2. Multicollinearity

In order to prevent problems with the linear regression, we next performed a Multicollinearity between all the metrics. We considered any correlation above 0.5 with $\alpha > 0.05$ to be high enough to warrant removing one of the metrics from the regression model.

Table 4.8 Multicollinearities Between Metrics

Metric	**Multicollinearity(s)**
Relevance	Proximity
Completeness	
Proximity	
Compactness	
Size	
NOEC	RC, LC, NoC, NoF
REC	LC, NoF
RI	
RC	NOEC, LC, NoC, NoF
LC	NOEC, REC, RC, NoC, NoF
ADIT-LN	
NoC	NOEC, REC, RC, LC, NoC
NoF	NOEC, REC, RC, LC, NoC

Based on the above correlations, we chose in this study to use the better of two models of which one used Relevance and the other used Proximity. Since RC, LC, NoC, and NoF were inter-correlated with so many metrics, they were taken out of consideration and the stepwise regression was performed with the remaining metrics.

4.2.2.3 Choosing the Metrics for Regression

In order to choose the best metrics for regression, a stepwise regression was done with the Minitab statistical software package. The α to enter and remove was 0.15.

The resulting model included Relevance, ADIT-LN, REC, and RI with an R^2 = 71.59% and R^2_{adj} = 71.02%. The full Linear Regression was performed with these metrics.

4.2.2.4 Linear Regression Results

Using the Minitab statistical software package [45], a linear correlation was preformed with predictors Relevance, ADIT-LN, REC, and RI over all 202 reduced ontology results with IQ as the response.

4.2.2.4.1 Prediction Model

The prediction model generated is

IQ = - 0.475 + 0.598 Relevance + 0.124 ADIT-LN + 0.00281 REC + 0.116 RI

The t test with associated p values for each metric are given below in Table 4.9

Table 4.9 t & p Values for Metrics

Metric	**t value**	**p value**
Relevance	16.42	$p < 0.0001$
ADIT-LN	5.94	$p < 0.0001$
REC	2.77	$p < 0.006$
RI	2.19	$p < 0.029$

All of the Variance Inflation Factors (VIF) were < 2, so Multicollinearity doesn't appear to be a problem. The variance is 0.1404 and R^2 = 71.6% and R^2_{adj} = 71.0% so we have a high degree of confidence that future predictions will match the model.

The Analysis of Variance (ANOVA) with a p value < 0.0001 demonstrates that it is very likely there is a relationship between these variables and the IQ. The full ANOVA is given below:

Source	DF	SS	MS	F	P
Regression	4	9.7878	2.4469	124.12	0.000
Total		201	13.6715		

So given this analysis, we initially concluded that the model was a good predictor of IQ, and therefore predicts the IQ of the reduced ontology relative to retaining information on the COI.

There are 14 leverage points in the data. That seems reasonable, since there a number of cases where the COI was not even in the reduced ontology. Some of the metrics and IQ reflect that by having very small numbers.

4.2.2.4.2 Residual Analysis

In order to be more confident in our model, we conducted further analysis on the residuals. The normality plot of the residuals is given below. This plot checks the assumption that the residual errors are normal and random. While there appears to be some non-normal behavior in the residuals, they do follow the line fairly well, and for this early study, we accept it.

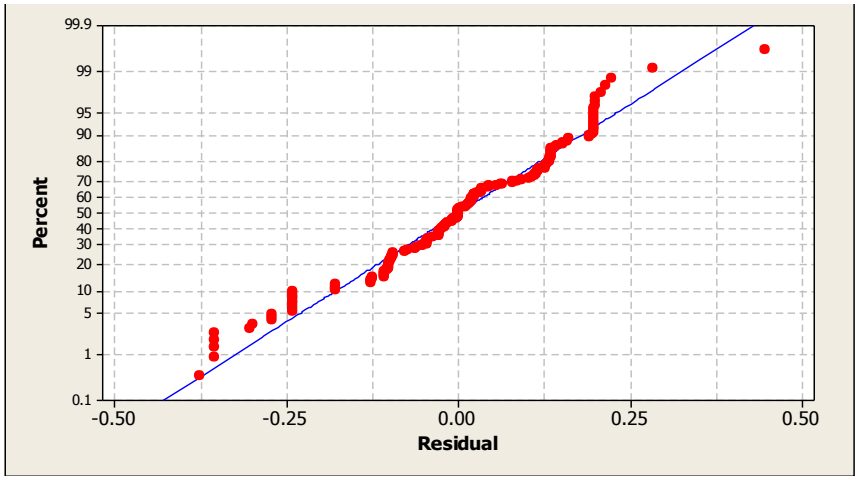

Figure 4.15 Normal Probability Plot of Residuals

The second assumption, randomness, is checked through the versus fit plot given in Figure 4.16. The errors seem to be fairly random and we see no obvious pattern.

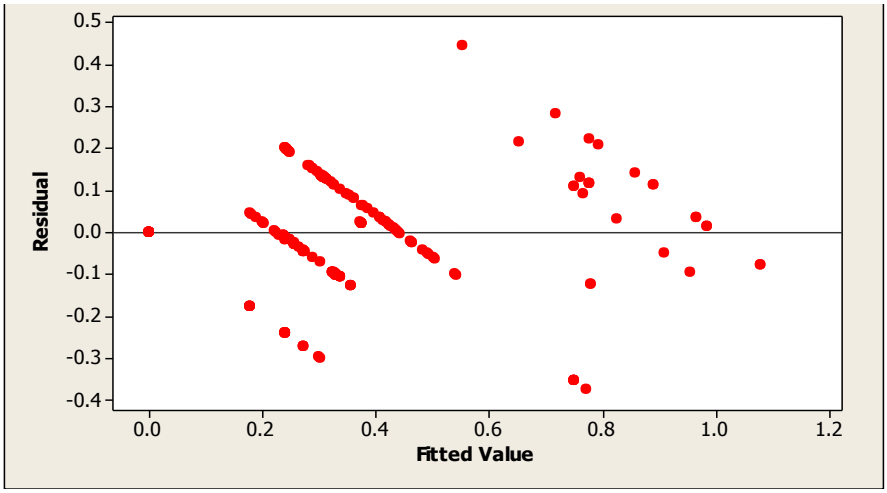

Figure 4.16 Versus Fit Plot for Residual Randomness

Given this analysis, we concluded that this model is a reasonably good predictor for IQ of a reduced ontology, and therefore is acceptable for use in mobile devices.

4.2.2.5 Discussion

This study expanded a previous study conducted on a single ontology that found a correlation between some of these metrics and the IQ. Here we increased the number of ontologies and the number of reduced ontologies. Also, instead of the simple individual metric correlations that we performed in our original study, we also developed regression models that employed several metrics, and thus improved our quality predictive capability.

Comparing to preliminary experiment #1, the informational metric's correlations all stayed roughly in the same direction and range. The performancd metrics became less important. Note that the metrics from Orme, Yao, Etzkorn and Yao, Orme, Etzkorn [35][36][37][53] were not included in the previous study. Both the previous study and

the current study show a decent, significant correlation between some of the metrics and IQ. Note that the size metric is the only metric without a reasonable correlation. See Table 4.10 below.

Table 4.10 Comparison of Correlations in Previous Study

Metric	Previous Correlation	New Correlation
Relevance	0.461 P=.002	0.72 p<0.0001
Completeness	0.101 P=.646	0.40 p<0.0001
Proximity	0.641 P=.001	0.68 p<0.0001
Compactness	-.425 P=.045	0.653 p<0.0001
Size	-0.283 P=.191	-0.258 p<0.0001

The linear model we created has a high R^2_{adj} at 71.0%. We believe we have created the beginnings of a model to allow the informational quality of an ontology reduced ontology to be measured dynamically.

4.3 Final Experiment

Once we had demonstrated that a predictive model was possible, we expanded our ontology base to include multiple ontologies in multiple domains to create a general predictive model of IQ and PG.

4.3.1 Research Description

The purpose of this study was to demonstrate that some subset of the structural metrics chosen can predict both the IQ of a reduced ontology compared to the original ontology, and the performance gain (PG) by the reduced ontology relative to the original ontology. First we chose the ontologies and queries for the study. Then reduced ontology candidates were generated automatically for each ontology. Next, IQ in the form of

average recall and PG were computed for each reduced ontology relative to its original ontology. Finally the selected metrics were applied to each reduced ontology candidate and used to generate a regression model to predict IQ and PG. Each of these steps is described in detail in the following sections.

4.3.1.1 Definition of Quality and Performance

We define a 'good' reduction in two ways, informational quality (IQ) and performance gain (PG). When creating a reduced ontology, the independent quality of the base ontology is ignored. Rather, the base ontology is considered the "gold standard" from an information retrieval perspective. IQ determines how well concept-of-interest (COI) queries were answered on the reduced ontology, relative to how those same queries previously were answered on the base ontology. On the other hand, PG measures the download and processing speed improvement for the client for the reduction versus the original ontology.

To measure IQ, we use average recall. From information retrieval, recall is defined as the ratio of the number of items returned from a query in the reduced ontology to the number of items returned in the original ontology. Recall is computed for each of the queries for an ontology and the mean of these recall scores forms the average recall. Thus, IQ measures the average amount of information retained in a reduced ontology relative to the COI.

To measure performance, we use performance gain (PG). Performance is computed by taking the ratio of time required for a mobile device to download and process a reduced ontology versus its base ontology. This measures the speedup of

handling the reduced ontology. This is performed in a simulated mobile device environment multiple times. The average speed up is the PG score

4.3.1.2 Ontology and Concept of Interest Selection

In this study, we selected four domains from which to select ontologies. There are twenty-one ontologies in total coming from the domains of food (5), travel (4), animals (4), and organizational structure (4). These ontologies were implemented in OWL, with a total of 829 named classes. A common concept of interest (COI) was chosen for each domain as a sample request a mobile device's agent might make. These COIs are shown in Table 4.11.

Table 4.11 Overview of Ontologies Selected for Study

Domain	Number	Ontologies COI
Animals	4	Birds
Food	4	Fish
Organizational Structure	4	Student
Travel	5	Flight

Based on the COI, seven queries for each domain were created by five unaffiliated volunteers. These queries represented information relative to a task a client using a mobile device might perform in that domain. These queries were applied to each ontology in the domain. The results of these queries were used to calculate IQ for each reduced ontology. A sample query from the travel domain is given below:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX ta: <http://www.owl-ontologies.com/TravelOnt08#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
SELECT ?start ?destination ?departure
WHERE {OPTIONAL { ?flight ta:toLocation ?destination .
        ?flight ta:fromLocation ?start .
        ?flight ta:departsAt ?departure .
        ?destination ta:locationName ?endName .
        ?start ta:locationName ?startName .
```

```
FILTER ( ?endName = "New York"^^xsd:string ||
        ?startName = "New York"^^xsd:string ) . } . }
```

4.3.1.3 Constructing Reduced Ontologies

An OWL ontology can be viewed as a graph, where the nodes are vertices and relationships, including both inheritance and property relations, are edges. A reduced ontology is defined as a subset of nodes and relationships from a base ontology that together form a smaller ontology. A reduced ontology should contain a subset of information contained in the base ontology specific to a COI and exclude non-relevant information.

A modified traversal algorithm created in our earlier study [39] (see Chapter Three and Section 4.1.1.3 in this chapter) was run on the ontologies with each node as the center. Each class defined in each of the ontologies was selected as the center for the traversal algorithm and a calculation was run with a threshold path length of two. Thus, from the 814 nodes in the original ontologies, 814 reduced ontology candidates were created, each one with the center set to a unique node in its original ontology.

4.3.1.4 Computing IQ and PG

Informational quality is a somewhat subjective idea. Previous work has developed various methods for create subsets or views of an ontology. However, the only attempt to validate the ontologies was with Gangemi et al., who matched information retrieval with expert opinions [11]. In order to develop a dynamic model, we need a computable measure of quality. For this study, we have chosen to use IQ to measure informational quality.

Five volunteers unrelated to this research were asked to create typical queries they would have about concepts in the domain. These volunteers had no knowledge of the specific ontologies being used for this study. These volunteers created twenty-eight queries to use in this study. It should be noted that the queries were written in English and had to be mechanically converted to fit the naming conventions of each original ontology. However, no change to the structure or purpose of the query was performed.

The querying of both the original ontology and all the reduced ontology candidates was performed using SPARQL [52] after loading each ontology into the Jena OWL parser [63]. The set of queries was applied to all 814 ontologies (the originals and all the reduced ontology candidates).

The recall score for each query was computed and the IQ for the reduced ontology candidates was computed over all the queries. As we discussed earlier, we are only concerned with recall with respect to the original ontology.

Performance is less subjective, but a still a broad term. There have been some studies into the performance of subsets versus full ontologies [16]. However, performance has typically been defined as simple size and has not been related to any metrics. We must first define what is meant by performance.

We choose to define performance gain in two parts, bandwidth and processing time. As with quality, the original ontology forms the basis for performance measurement. We measure the bandwidth requirements and processing time of the base ontology and then for the reduced ontology candidates.

In order to make these measurements, a simulation of a mobile device, with various processing capability and bandwidths, was created. This simulation interacts with a mock semantic web service provider. A series of runs with various capabilities for bandwidth and processing capabilities were completed. From the results we computed average bandwidth and processing time required for the original ontology and its reduced ontology candidates. The ratio of average time for the original ontology and each reduced ontology candidate was set to the PG for that reduced ontology.

4.3.1.5 Applying the Metrics

While metrics have been in use for software systems for many years, applying them to ontologies is a relatively new idea. There have been metrics previously defined to describe the structure of an ontology, these include Yao et al., 2005 [53], Orme et al., 2006 [35][36]. We chose the structural metrics because they are computable in a short time, making them useful for a dynamic model. Other, semantic type metrics, are difficult to compute in a short time and therefore not as useful for this model. None of the metrics defined for general ontology quality give a look at reduced ontologies specifically. So we define five new reduced ontology metrics that we used in a previous study (see Chapter Three) (we published this in Schrimpsher & Etzkorn, 2009 [39]) in order to look at how well a reduced ontology matches the original ontology. Table 4.12 represents a description of structural metrics chosen for this study.

The metrics were applied to each reduced ontology candidate and the original ontology. For this study, COIs are limited to a single choice of a node in the ontology. Since Jena provides an indexed list of the nodes in the ontology, checking for a single node is a trivial task.

Table 4.12 Ontology Metrics used in this Study

Metric	Description
NOEC	Number of external classes included in the ontology [35]
REC	References to external classes included in the ontology [35]
RI	Number of references included in the ontology[35]
RC	Number of root classes in the ontology [53]
LC	Number of leaf classes in the ontology [53]
ADIT-LN	Average depth of the inheritance tree in the ontology [53]
NoC	Number of classes in the ontology [36]
NoF	Number of fan-outs in the ontology [36]
Relevance	Ratio of unreachable nodes to the total nodes [39]
Proximity	Ratio of length to external references vs the original ontology [39]
Completeness	Ratio of external references to the total references [39]
Compactness	Ratio of the path length the maximum path length [39]
Size	Ratio of the number of nodes to the original ontology [39]

We then computed the metrics defined in Table 4.12 on each of the remaining reduced ontologies. Each metric was automatically computed by a tool we developed in Java based on the Jena OWL parser. Each of the metrics defined in Table 4.12 is then correlated with the IQ score. This correlation provides a filter on which metrics are actually useful at predicting quality. This will provide confidence in which metrics are useful to quality and which may not be.

Each of the metrics defined in Table 4.12 is then correlated with the PG score. This correlation provides a filter on which metrics are actually useful at predicting performance. This will provide confidence in which metrics are useful to measure

performance and which may not be. We used Cohen's correlation magnitude scale as in the previous experiments.

4.3.1.6 Building the Model

Linear regression is a well understood way for finding linear models between observed data and responses. However, it does suffer from sensitivity to Multicollinearity, so the first step in creating this a linear regression model is finding and removing metrics with Multicollinearity. Once the metrics to be used in the regression are decided, a step-wise regression is performed to find out which metrics provided the best model.

A preliminary attempt at creating a linear model between metrics and IQ in a single domain was performed previously (see earlier, Section 4.2) (we published this in Schrimpsher & Etzkorn [40]). This study expands those results to include models predicting PG as well as IQ and using multiple domains. The plan was create two linear regressions (one for IQ and one for PG) based on the 829 reduced ontology candidates in the four domains chosen. Once the two models were found, a standard analysis was performed.

4.3.2 Results

4.3.2.1 IQ and PG Correlations

In the first part of this study, each metric was correlated with the IQ and PG scores. These correlations were ranked according to Cohen's magnitude scale [5]. The IQ correlations are shown in Table 4.13. Two metrics have a very large correlation with IQ, proximity and size. Compactness, relevant nodes, NoC, NoF, and NOEC all have large

correlations. The rest have moderate correlations except for RI which has a minor correlation. This is promising for creating a useful model, since all but one of the metrics have at least a moderate correlation with IQ.

Table 4.13 The Correlations between Ontology Metrics and IQ

Ontology Metric	**Correlation to IQ**	**p value**	**Cohen's Magnitude Scale**
Proximity	0.773	$p < 0.0005$	Very Large
Size Metric	-0.708	$p < 0.0005$	Very Large
Compactness	0.689	$p < 0.0005$	Large
Relevant Nodes	0.593	$p < 0.0005$	Large
NoC	0.585	$p < 0.0005$	Large
NoF	0.536	$p < 0.0005$	Large
NOEC	0.520	$p < 0.0005$	Large
leafClasses	0.425	$p < 0.0005$	Moderate
Completeness	0.411	$p < 0.0005$	Moderate
rootClasses	0.408	$p < 0.0005$	Moderate
ADITN	0.344	$p < 0.0005$	Moderate
REC	0.341	$p < 0.0005$	Moderate
RI	0.147	$p= 0.057$	Minor

The PG correlations are shown in Table 4.14. RI is the only metric with a large correlation with PG. Completeness, ADITN, and NOEC all have moderate correlations. The rest of the metrics have minor correlations. While this is less promising than the IQ results, it is still possible we can create a good model to predict PG. However, it may be there are variables missing from this model that are needed to estimate PG.

Table 4.14 The Correlations between Ontology Metrics and PG

Ontology Metric	Correlation to IQ	p value Cohen's	Magnitude Scale
RI	0.533	$p < 0.0005$	Large
Completeness	0.471	$p < 0.0005$	Moderate
ADITN	0.386	$p < 0.0005$	Moderate
NOEC	0.309	$p < 0.0005$	Moderate
Size Metric	-0.255	$p < 0.0005$	Minor
NoC	0.217	$p < 0.0005$	Minor
NoF	0.192	$p < 0.0005$	Minor
Proximity	0.192	$p < 0.0005$	Minor
rootClasses	0.185	$p < 0.0005$	Minor
Relevant Nodes	0.175	$p < 0.0005$	Minor
Compactness	0.165	$p < 0.0005$	Minor
leafClasses	0.141	$p < 0.0005$	Minor
REC	0.123	$p < 0.0005$	Minor

4.3.2.2 Multicollinearity

In order to create a linear regression model, we must first identify and remove any intercorrelated metrics, since these will bias the results and give us an unreliable model. As a threshold, we set a correlation coefficient of 0.5 with a p value < .05 to say two metrics were multicollinear. Table 4.15 shows each metrics and which other metrics it is intercorrelated with.

Table 4.15 List of Multicollinear Metrics

Metric	Multicollinearity
Compactness	Relevant Nodes, Size, Proximity, NOEC
Relevant Nodes	Compactness, Proximity
Completeness	Size, ADITN, NoC, NoF
Size	Compactness, Completeness, , Proximity, NOEC, rootClasses, leafClasses, NoC, NoF
Proximity	Compactness, Relevant Nodes, Size, NoC, NoF
NOEC	Compactness, Size, rootClasses, NoC, NoF
REC	rootClasses, leafClasses, NoF
RI	
rootClasses	Size, NOEC, REC, leafClasses, NoC, NoF
leafClasses	Size, REC, rootClasses, NoC, NoF
ADITN	Completeness
NoC	Size, Proximity, NOEC, rootClasses, leafClasses, NoF
NoF	Completeness, Size, Proximity, REC, rootClasses, leafClasses, NoC

Of particular interest are RI an ADITN. RI isn't multicollinear with any other variable. This may be due RI lack of interaction with the actual structure of the ontology, since it only contains the external owl files. ADITN had a low correlation with other variables. Only Completeness was above 0.5 and had a significant p value.

In order to ensure that no multicollinear metrics are in the models, we created a list of possible variables for the models by choosing all combinations of metrics. We started with the highest correlation to IQ and PG and then added metrics that were not correlated with any variables already in the model.

4.2.3 IQ Model

For the IQ model, a number of variable options were analyzed using a stepwise regression approach. Montgomery, Peck & Vining [30] defines a stepwise regression as a way of evaluating a number of regression models automatically by evaluating their

partial F statistics at each step in order to find the best linear regression model. The stepwise regression was done with the Minitab statistical software package [54] and used a threshold of 0.15 to enter and remove. Based on the results of this stepwise regression, the variables selected were Proximity, NOEC, leafClasses, Completeness, and RI. The model selected was

$$IQ = -0.0616 + 0.709\ P + 0.0748\ NOEC + 0.0025\ LC + 0.0650C. \quad (4.10)$$

The results of the t-test on the predictors are shown in Table 4.16. A threshold of 0.10 was chosen. The p values are all < 0.005 with the exception of Completeness (p=0.098), but they all fall within our chosen threshold. All of the Variance Inflation Factors (VIF) are < 2, so Multicollinearity doesn't appear to be a problem. The R2 values are given in Table 4.17. These were the highest values of all the variable combinations examined. These values give us a high confidence in future predictions.

Table 4.16 T-test Results Whether There is a Relationship between the Metrics and IQ

Predictor	**t**	**p**	**VIF**
Constant	-2.80	<0.0005	N/A
Proximity	24.63	<0.0005	1.36
NOEC	9.36	<0.0005	1.43
leafClasses	3.25	<0.0005	1.99
Completeness	1.66	0.098	1.60

Table 4.17 R2 Statistics for IQ Model

R Type Statistic	**Value**
R2	64.6%
R2adj	64.4%
R2pred	63.91%

Source	DF	SS	MS	F	P
Regression	4	57.884	14.471	368.95	< 0.0005
Residual Error	809	31.731	0.039		
Total	813	89.614			

ANOVA analysis [30] shown in above, demonstrates that there is definitely a relationship between the metrics and IQ.

In order to be more confident in our model, we conducted further analysis on the residuals. The standard residual plots are given in Figure 4.17. The first assumption, normality of the residual data, is checked through the normality plot in the upper left of Figure 3. While there appears to be some non-normal behavior at 0 in the residuals, they do follow the line fairly well and we accept it. This can also be seen in the histogram plot of the residuals, showing a bell shape of the residuals. The second assumption, randomness, is checked through the versus fit plot given in the upper right of Figure 4.17. The errors seem to be fairly random and no obvious pattern is seen by the authors. The third assumption, that residuals are independent of order, is shown in the versus order plot in the lower right of Figure 4.17. The residuals seem to be fairly block shaped and there appears to be no order dependence.

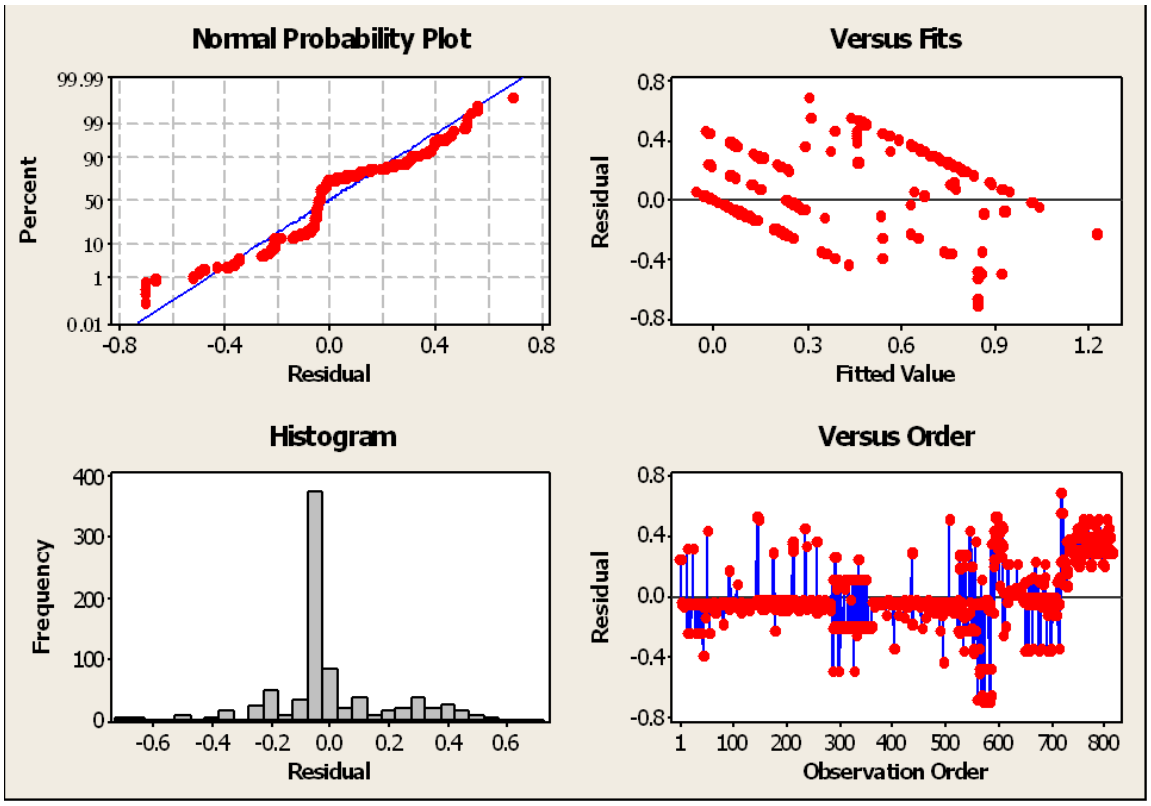

Figure 4.17 Standard residual Plots for IQ

So given this analysis, we initially conclude that the model is a good predictor of IQ, and therefore that it predicts the quality of the reduced ontology for answering the users request.

4.3.2.4 PG Model

For the PG model, a number of variable options were analyzed using a stepwise regression approach. The stepwise regression was done with the Minitab statistical software package [54] using a threshold of 0.15 to enter and remove. Based on the results of this stepwise regression, the variables selected were Proximity, NOEC, leafClasses, Completeness, and RI. The model selected was

$$PG = 1458 - 331\ RI - 108C - 15.3\ NOEC + 0.831LC. \quad (4.11)$$

The results of the t-test on the predictors are shown in Table 4.18. A threshold of 0.10 was chosen. The p values are all < 0.005 and they all fall within our chosen threshold. All of the Variance Inflation Factors (VIF) are < 2, so Multicollinearity doesn't appear to be a problem.

Table 4.18 T-test Results Whether There is a Relationship between the Metrics and PQ

Predictor	**t**	**p**	**VIF**
Constant	28.37	<0.00005	N/A
RI	-25.00	<0.00005	1.12
Completeness	-10.07	<0.00005	1.76
NOEC	-7.39	<0.0005	1.40
leafClasses	4.28	<0.0005	1.86

The R2 values are given in Table 4.19. These were the highest values of all the variable combinations examined. These values give us a high confidence in future predictions. ANOVA analysis shown below, demonstrates that there is definitely a relationship between the metrics and IQ.

Table 4.19 R2 statistics for PQ model

R Type Statistic	**Value**
R2	59.2%
R2adj	59.0%
R2pred	55.9%

Source	**DF**	**SS**	**MS**	**F**	**P**
Regression	4	3126737	781684	293.41	< 0.0005
Residual Error	809	2155296	2664		
Total	813	5282034			

In order to be more confident in our model, we conducted further analysis on the residuals. The standard residual plots are given in Figure 4.18. The first assumption, normality of the residual data, is checked through the normality plot in the upper left of Figure 4.18. There appears to be non-normal behavior for large residuals. However for

lower values of residuals a normal distribution is reasonable. This can also be seen in the histogram plot of the residuals, where the bell shape is skewed to the right. The second assumption, randomness, is checked through the versus fit plot given in the upper right of Figure 4.18.

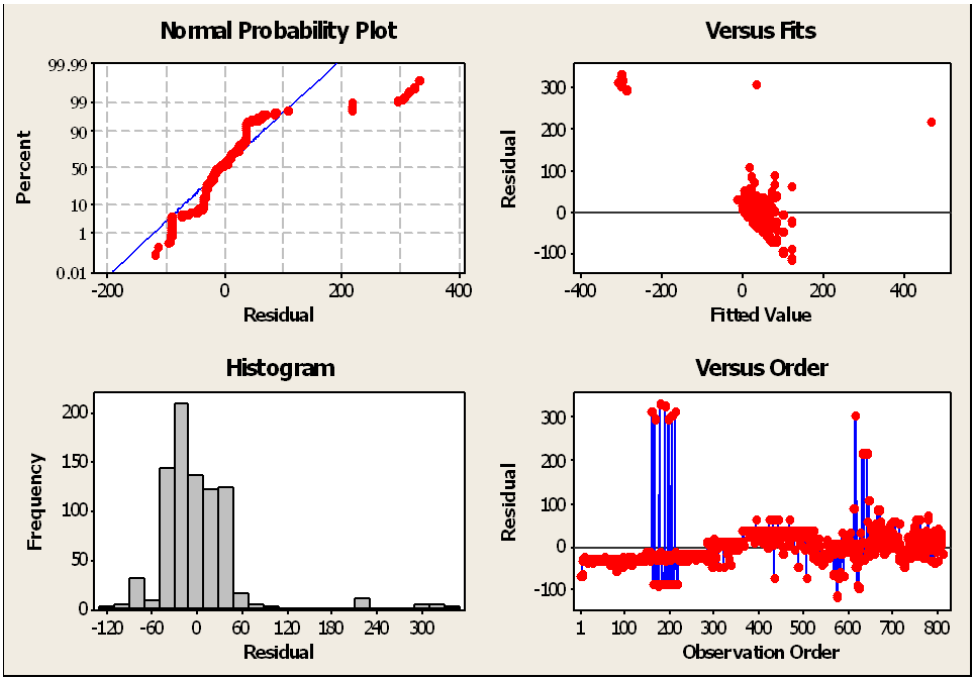

Figure 4.18 Standard residual plots for PG

Again it seems that large residuals make the plot look nonrandom. The third assumption, that residuals are independent of order, is shown in the versus order plot in the lower right of Figure 4.18 Again, very large residuals appear to be outliers, but otherwise the data seems independent of order.

Given this analysis, it is possible that this model is either non-linear, or there is some variable inherent in the model that is not being measured. As this is our first attempt at a PG model, we are confident that the model can be adjusted, with either a higher order term or a new metric currently not being measured.

4.3.2.5 Discussion

This study expanded the two preliminary studies to include multiple domains and ontologies. The preliminary studies focused on a single domain, travel agencies, and a small set of ontologies to prove the possibility of a predicative model. In this study we used four domains and twenty ontologies, creating 814 candidate reduced ontologies from which to build predicative models. Based on the data, we developed what we believed was a general regression model, employing several metrics, and thus improving our IQ and PG predictive capability.

4.4 Experiment Conclusions

Creating a true Semantic Web requires handling all types of clients that may use available services. As the size of the web grows, the size of web service ontologies will grow as well, out pacing mobile devices' ability to use them. In this research, we created a workable model to reduce the size of the ontologies that a mobile device must process while maintaining the core of the information the client needs to perform his task.

Given the results from this experiment we have the ability to quickly compute IQ and PG using structural ontology metrics. The model for performance (PG) is not as convincing and may require additional research. That being said, we were confident a dynamic algorithm could be developed to generate high quality reduced ontologies dynamically.

This study concluded the first phase of the research, resulting in a general predictive model of IQ and PG. The next step in this research was to build an algorithm that would dynamically create reduced ontologies at the time of the user requests. This

would allow a web service ontology to be tailored to a specific task a software agent is engaged in. We discuss the second phase in the following chapter.

CHAPTER FIVE

A METRICS BASED DYNAMICALLY REDUCED ONTOLOGY GENERATION ALGORITHM FOR MOBILE DEVICES

This chapter describes the second phase of our research. Using the predictive models developed in Chapter Four as the measures for information quality (IQ) and performance gain (PG), in this chapter we developed a dynamic algorithm to generate reduced ontologies "on-the-fly" based on user requests.

5.1 Research Description

The purpose of this study was to develop and demonstrate a dynamic reduced ontology generating (DROG) algorithm based on the predictive model developed in Chapter Four. We started with a brute force, syntactic locality based traversal algorithm previously described in Section 4.1.1. This proved to be inefficient and a change in the algorithms methodology was required.

In order to validate the DROG algorithm, we conducted four experiments. Each experiment was conducted on a subset of twenty ontologies from four domains. The first experiment analyzed the changes in IQ and PG over time to find a method for determining a stopping point when the best balance of IQ and PG had been achieved. The second experiment was conducted to compare ranking algorithms to limit the number of candidate nodes to add at each step of the algorithm while still achieving similar results in checking every node. The third experiment was conducted to develop a hybrid approach from the options in experiment two to enable us to build a reduced ontology more quickly. The final experiment implemented the DROG algorithm with the selected hybrid heuristic on the full set of ontologies and scored each reduced ontology with its recall and performance increase versus the original. In the rest of this section, we discuss our DROG algorithm and describe our various validation experiments. Then in Section 5.2, we provide results from our experiments.

5.1.1 Basic DROG Algorithm

The Dynamic Reduced Ontology Generator algorithm (DROG) is designed to dynamically build a reduced ontology based on a client's request. This request will include a core concept of interest (COI) that summarizes the client's interest. DROG balances PG with IQ to minimize the bandwidth and processing time required while including all relevant information to the client's needs.

The general structure of the algorithm treats the ontology as a complex graph and is similar to the traversal algorithm of Noy & Musum, 2007 [33]. It starts with a core COI and adds classes, properties, and individuals at each step based on the IQ and PG scores. When it reaches a point where the increase in IQ doesn't justify the decrease in

PG, it stops and returns the reduced ontology to the client. The notional view of the algorithm is given in Table 5.1.

Table 5.1 Notional View of DROG Algorithm

```
1   DROG(Ontology o, COI c)
2   Returns Reduced Ontology ro
3     Add node that matches c to ro
4       While improvement to IQ better than decrease in PG
5             Find best node in o to add within 1 path-length of any nodes in ro
6             Add new node to ro, including any new properties and individuals
7             Calculate IQ and PG
8       End while
9   Return ro
```

To illustrate how DROG works at a high level, we offer an example. See Figure 5.1 for a conceptual view of what the algorithm is doing. Assume we took this notional algorithm and applied it to a travel agency ontology where the COI is a "flight." First we added the COI node, or *Flight*, to the new reduced ontology (line 3). The next step was to select the best node within one path-length of the nodes in the reduced ontology, currently only *Flight*. That returns *Transportation*, a parent class to *Flight*. Continuing on, we then added *Location* which allowed us to add two properties with the Domain of *Transportation* and range of *Location*, *source* and *destination*. Finally in step 4, we added *Aircraft* which also contains a property *uses* with a domain of *Flight* and a range of *Aircraft.*

With each step in Figure 5.1, the IQ and PG values were recomputed for the current reduced ontology. As you can see from the example, in the early steps IQ increases rapidly and PG decreases steadily. As the algorithm continued, though, the change in IQ gets smaller and smaller and PG continues to shrink. The method we used to determine a stopping point for the DROG algorithm was created in experiment #1.

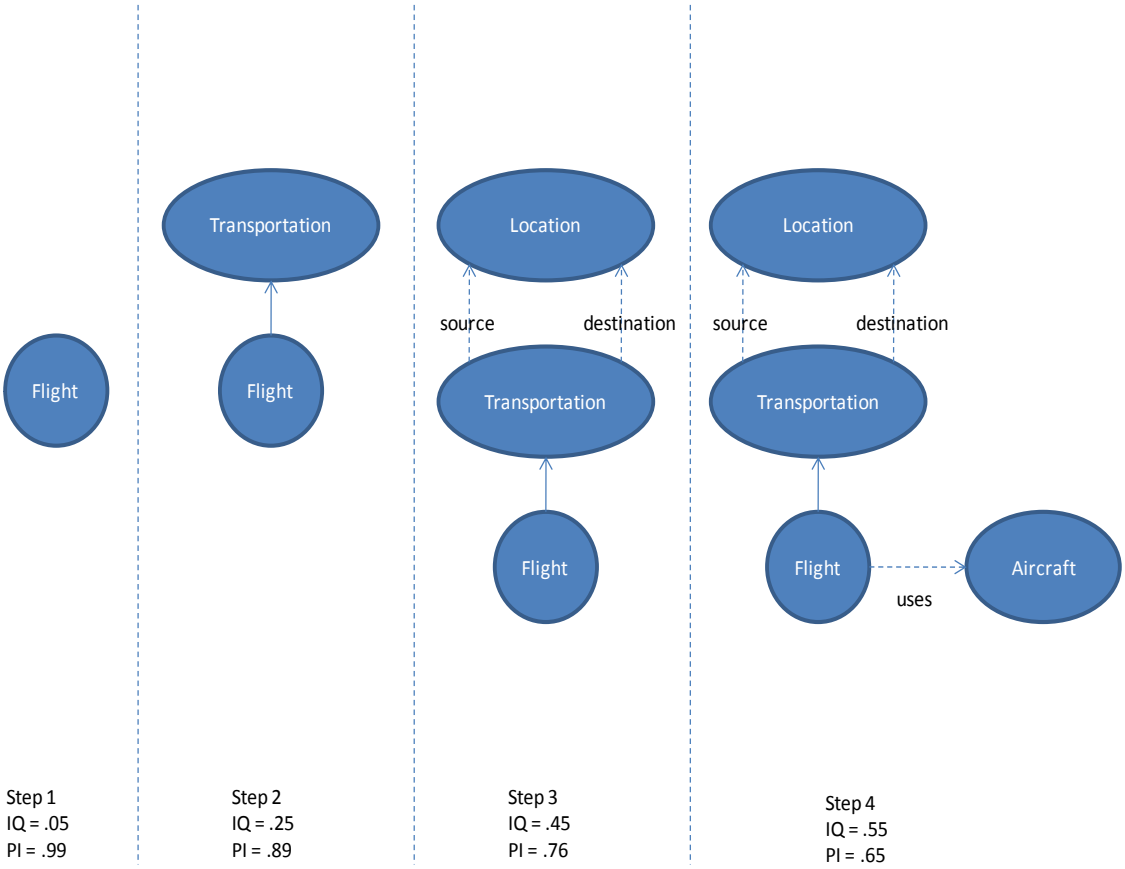

Figure 5.1 A notional view of how the DROG algorithm adds new nodes & properties at each step in the algorithm and computes the new IQ and PG scores.

5.1.1.1 Finding the Best Balance for IQ and PG

Finding the right reduced ontology was a balance between IQ and PG. If we focused solely on IQ, we ended up with the original ontology, since its IQ score is by definition 1.0. If we focused solely on PG, we ended up with a single node, the COI, since that is by definition the best PG score. An objective and general stopping point of the algorithm, referenced in line 4 of Table 5.1, is needed to effectively generate a reduced ontology.

One way of approaching this stopping problem is to look at the rate of change of both the IQ and PG values over time. As the reduced ontology grows larger, the impact of new nodes becomes smaller. We conducted experiment #1 to find a mathematical formula to find when the IQ rate of change was insignificant relative to the PG rate of change.

5.1.1.2 Candidate Node Ranking

The second issue when implementing the DROG algorithm was filtering the nodes for which we would compute the IQ and PG scores. Our initial attempt at checking every possible node proved impractical for even medium sized ontologies. What was needed was a simple heuristic way to rank the nodes so as to limit the number used to compute IQ and PG values.

In experiment #2, we compared three possible ranking methods with the "brute force" method of checking every possible node. The first method was locality, which chose the five closest nodes to the COI. We believed this would focus on concepts more relevant to the COI sooner. The second method was degree centrality, which chose the five nodes with the most relationships to other nodes. We believed this would focus on hub concepts more important to the overall ontology. The final method was a "wholesale" method. In it we did to not rank nodes at all, but rather added all nodes *k* distance from the COI simultaneously, where k was the current round. We believed this would get to the final ontology quicker, even if it was less granular and would add some nodes that were unnecessary.

5.1.1.3 Hybrid DROG Algorithm

Once experiment #2 was completed, we understood the strengths and weakness of each method for building a reduced ontology. It occurred to us that a combination of hybrid and granular filtering methods might give us the best of both worlds, the speed up of the "wholesale" method with the minimal size of the ranking methods.

In experiment #3 we combined the "wholesale" method with the other two ranking methods to create a hybrid method. This gave us a performance boost in the initial rounds of adding nodes and then fine tuning when DROG got near the answer. This was the final form of the DROG algorithm used in the final experiment.

5.1.1.4 Validation of DROG

In the final experiment we took the hybrid DROG algorithm developed over the first three experiments and applied it to all twenty ontologies from four domains. We then performed the queries developed in Chapter Four on each reduced ontology and computed the real IQ scores.

5.2 Results

This study was broken into four experiments. The first three were designed to find the best method for selecting nodes at each round of the algorithm. The final method is to check the correctness of the algorithm in creating reduced ontologies.

5.2.1 Experiment #1 Finding the Best Balance of IQ and PG

As we discussed in Section 5.1, at some point the increases in IQ are not worth the decrease in PG. A test was run on sixteen ontologies in four domains to determine the relationship between the increase in IQ and decrease in PG. A sample chart of IQ and

PG versus the steps in the algorithm is shown in Figure 5.2(a-d). As one can see, if the algorithm was run to completion without stopping the original ontology would be the result.

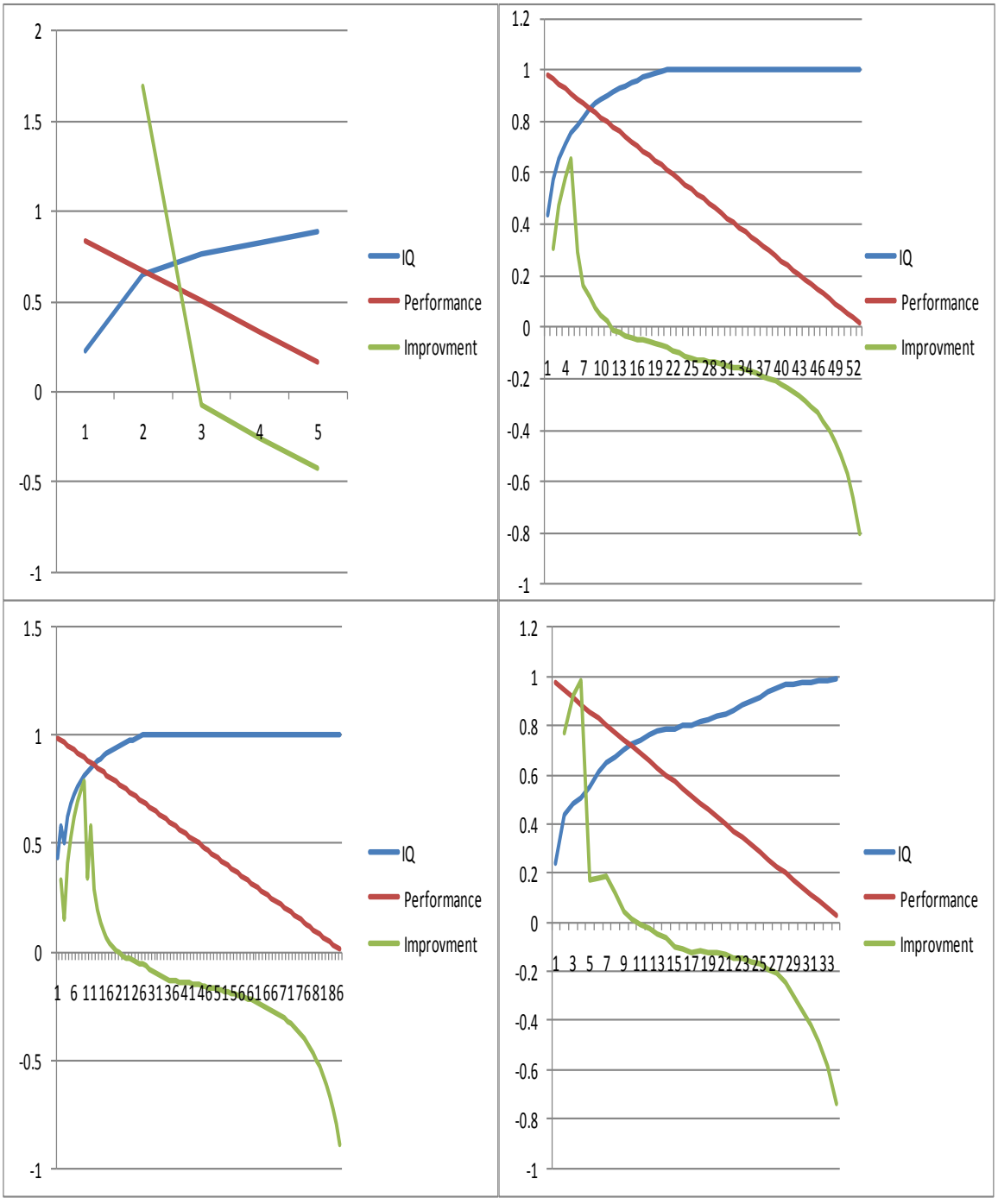

Figure 5.2(a) Comparison of how IQ and PG change as the algorithm runs across the animal domain ontologies. Notice how IQ increases a large amount in the early steps but tapers off while the PG decreases linearly.

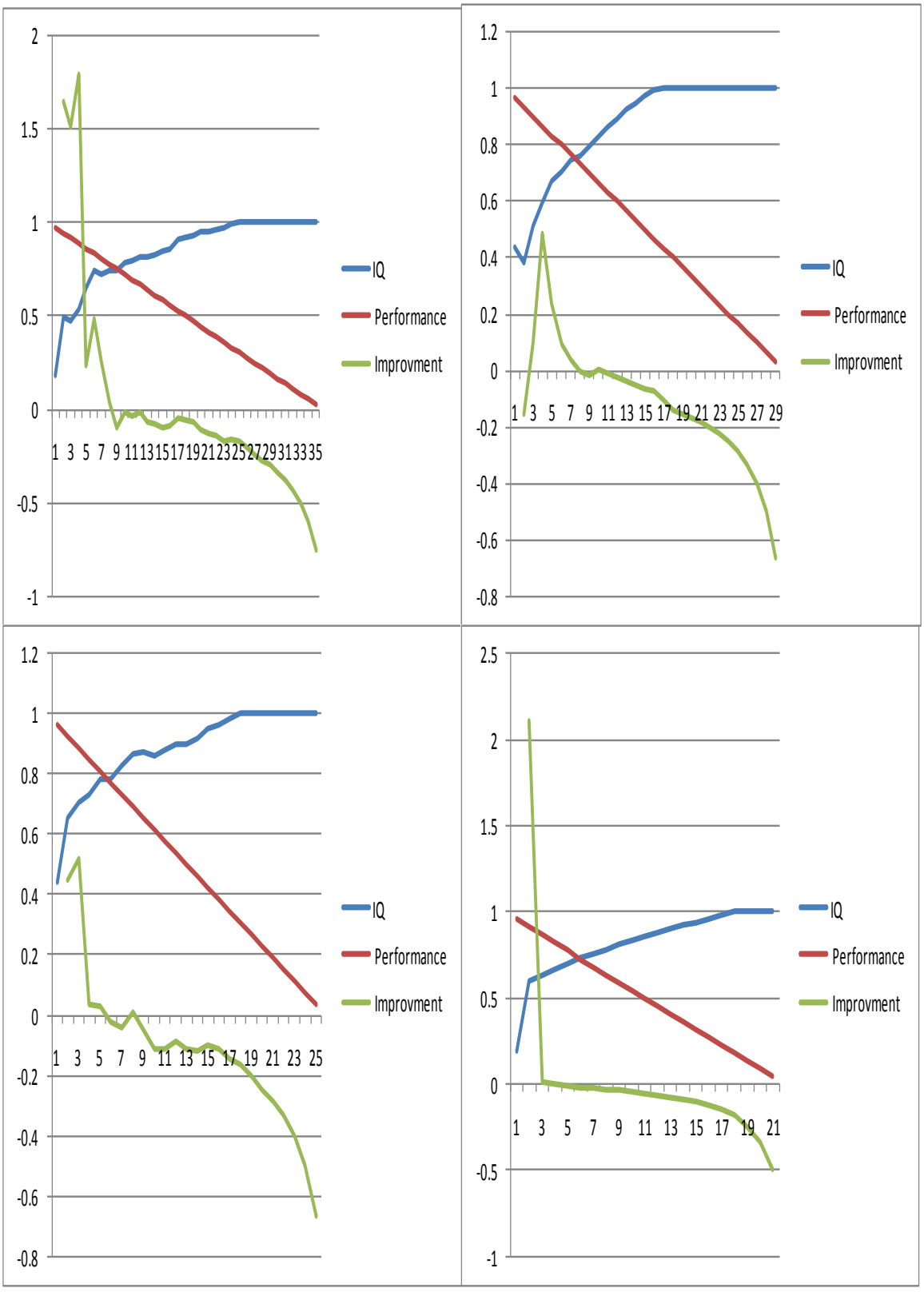

Figure 5.2 con't (b) Comparison of how IQ and PG change as the algorithm runs across the organizational domain ontologies. Notice how IQ increases a large amount in the early steps but tapers off while the PG decreases linearly.

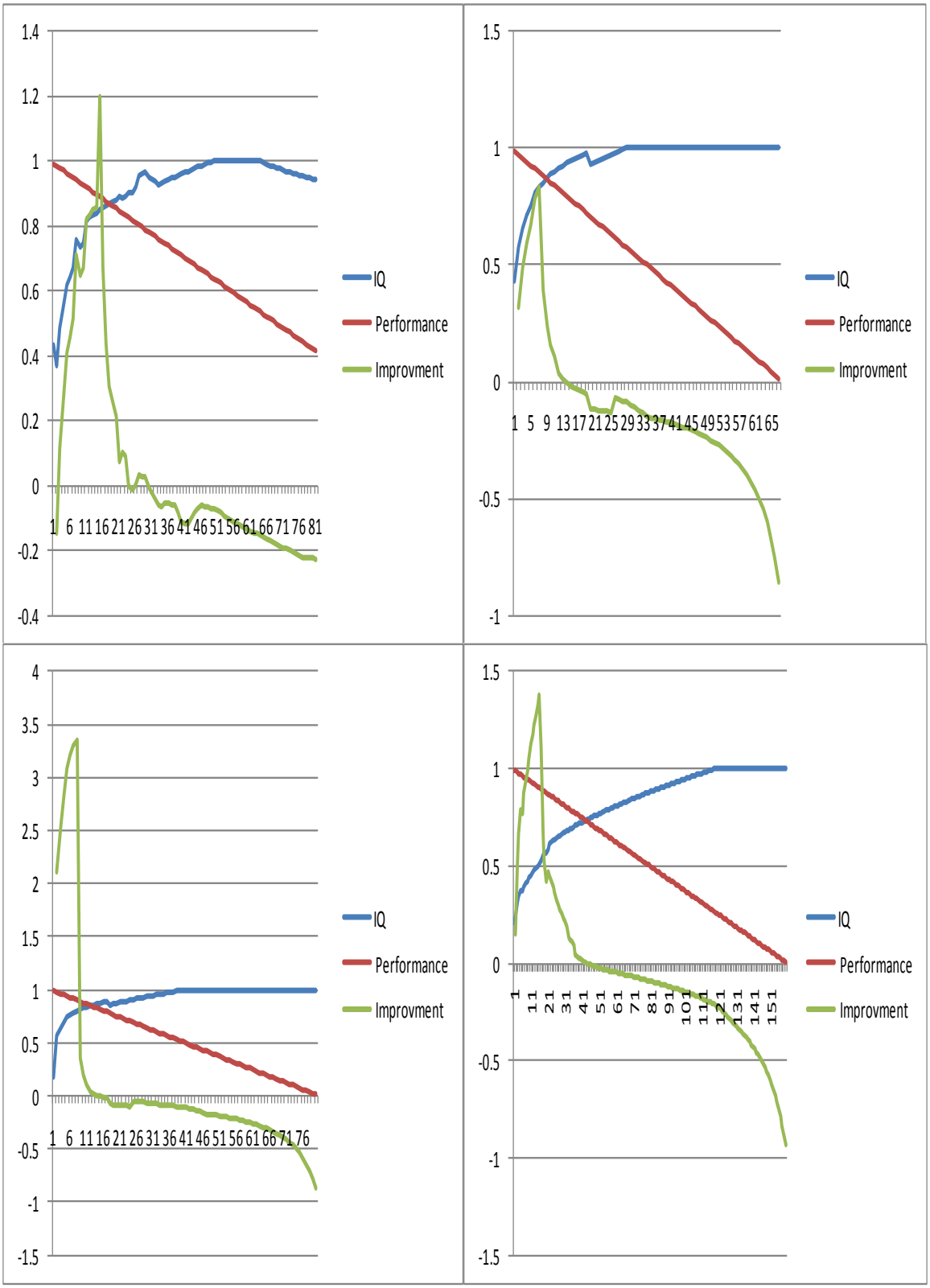

Figure 5.2 con't (c) Comparison of how IQ and PG change as the algorithm runs across the food domain ontologies. Notice how IQ increases a large amount in the early steps but tapers off while the PG decreases linearly.

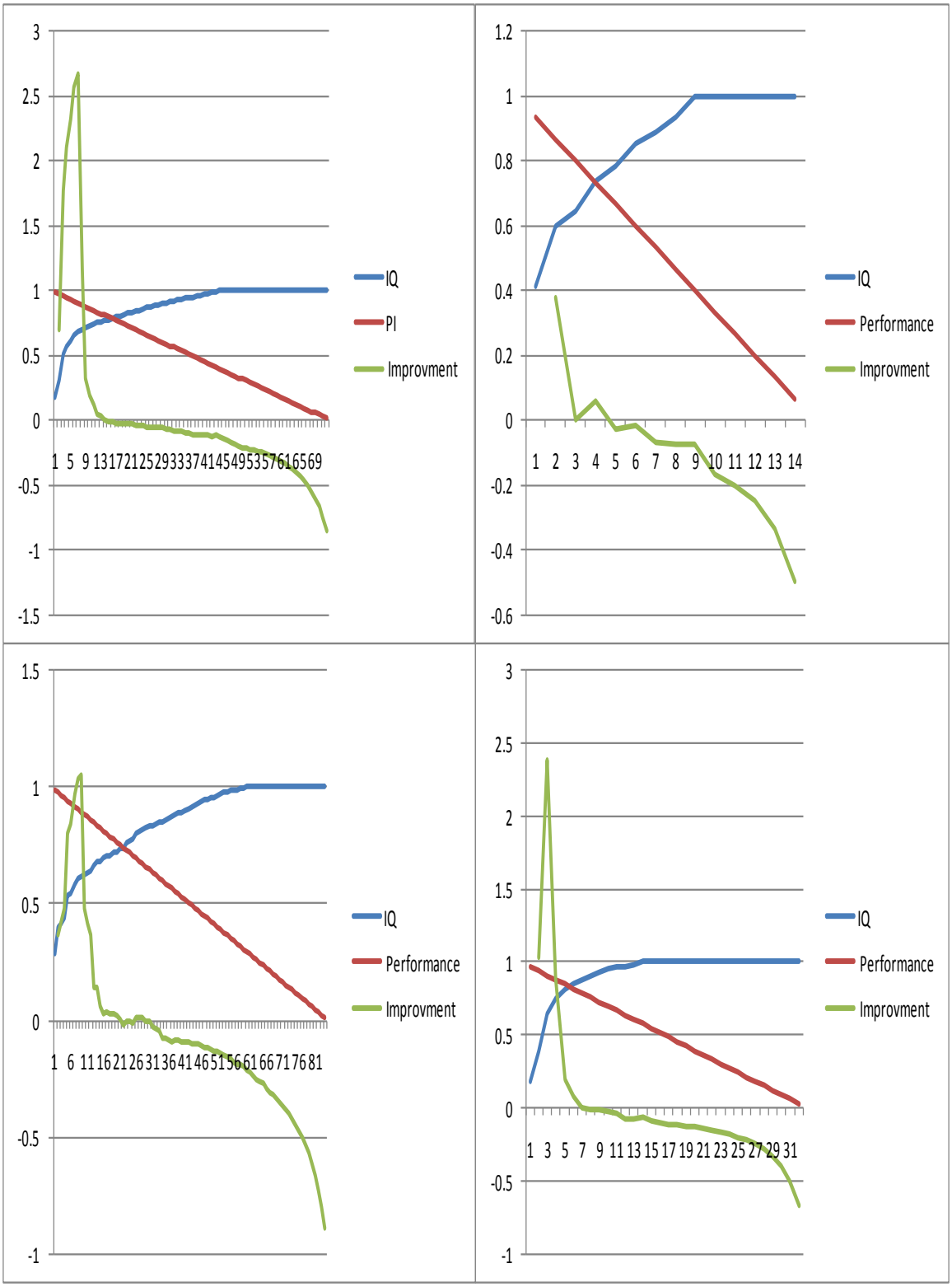

Figure 5.2 con't (d) Comparison of how IQ and PG change as the algorithm runs across the travel domain ontologies. Notice how IQ increases a large amount in the early steps but tapers off while the PG decreases linearly.

In order to achieve the gains possible with the DROG algorithm, a stopping point needs to be selective. As one can see from Figure 5.2, the IQ gains per step early on are far greater than the PG loses. As the algorithm continues, however, the decrease in PG equals and then surpasses the gains in IQ. Figure 5.3 shows an expanded view of the travel agency ontology. Notice how the rate of change for the predicted IQ has three periods, rapid initial improvement, medium improvement, and little to no improvement.

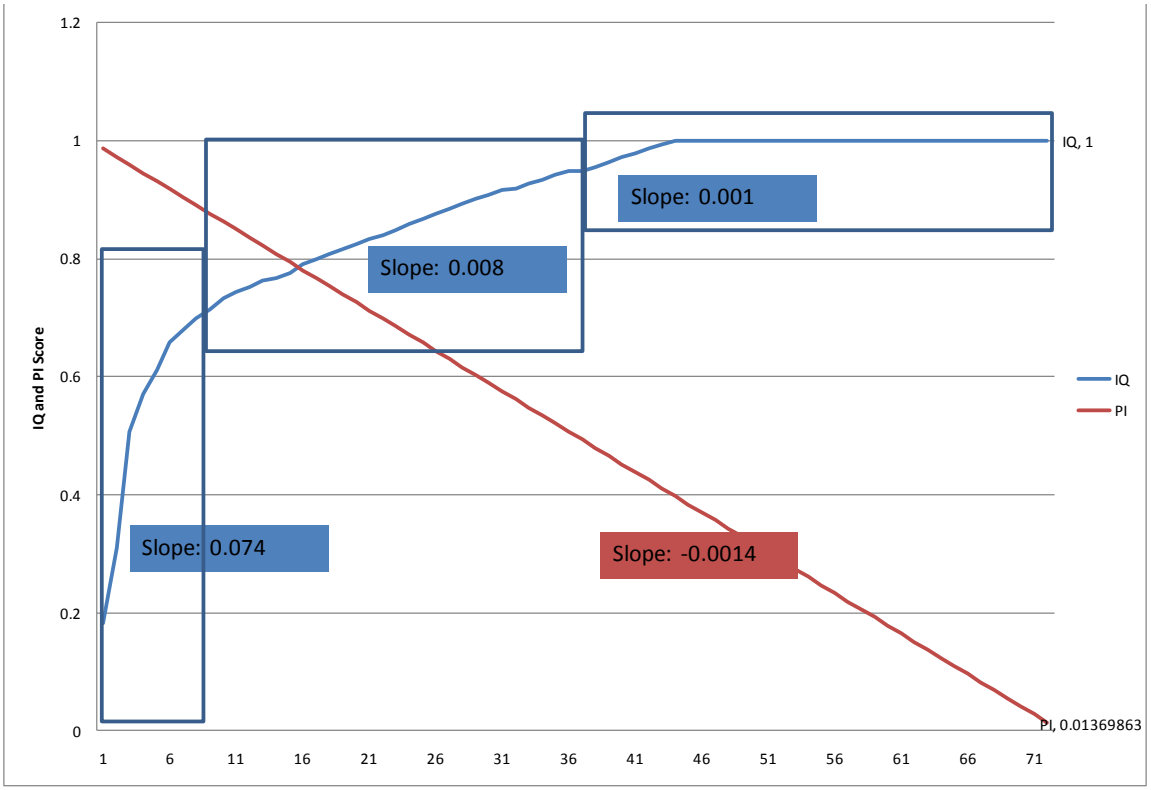

Figure 5.3 A detailed analysis of the travel agency ontology. Notices how the slope of IQ falls into three categories, huge initial increase, moderate increase, and finally minimal increase. The essence of the stopping threshold is finding these areas when adding IQ gives minimal increase to the predicted IQ value.

Given the nature of the models, we chose to view this problem as a game between IQ and PG. A min-max approach from game theory was used to attempt to minimize the PG loss while maximizing the IQ gain.

We combined the slope of the IQ and the PG to get a combined *improvement* score. To prevent a single outlying value from affecting the algorithm, we instead used a moving average of the IQ and PG slope. We then compare each candidate "move" (i.e., a specific nodes IQ and PG score) to doing nothing at all. This comparison is shown Table 5.2.

Table 5.2 A Comparison of Three Choices. Case 1 gives a positive improvement, thus the algorithm would continue. Case 2 gives a negative improvement, so the algorithm would stop and returned the completed reduced ontology. Doing nothing yields a 0 improvement

Candidate	**IQ Change Moving Average**	**PG Change(Moving Average)**	**Improvement**
Case 1	0.09	-0.08	.01
Case 2	0.07	-0.08	-.01
Do Nothing	0	0	0

We implemented his idea of a min-max decision at the end of each round. We did some sample empirical runs on various ontologies and determined that the following combination of IQ and PG moving average gave the best results

Improvement = 2 * IQ_moving_ave + PG_moving_ave. (5.12)

Using equation 5.12, a sample run was conducted (shown in Figure 5.4). The algorithm would terminate at step 13 with an IQ of 0.75 and a PG of 0.84. This reduced ontology is only 16% as big as the original ontology while containing 75% of the information relevant to the COI.

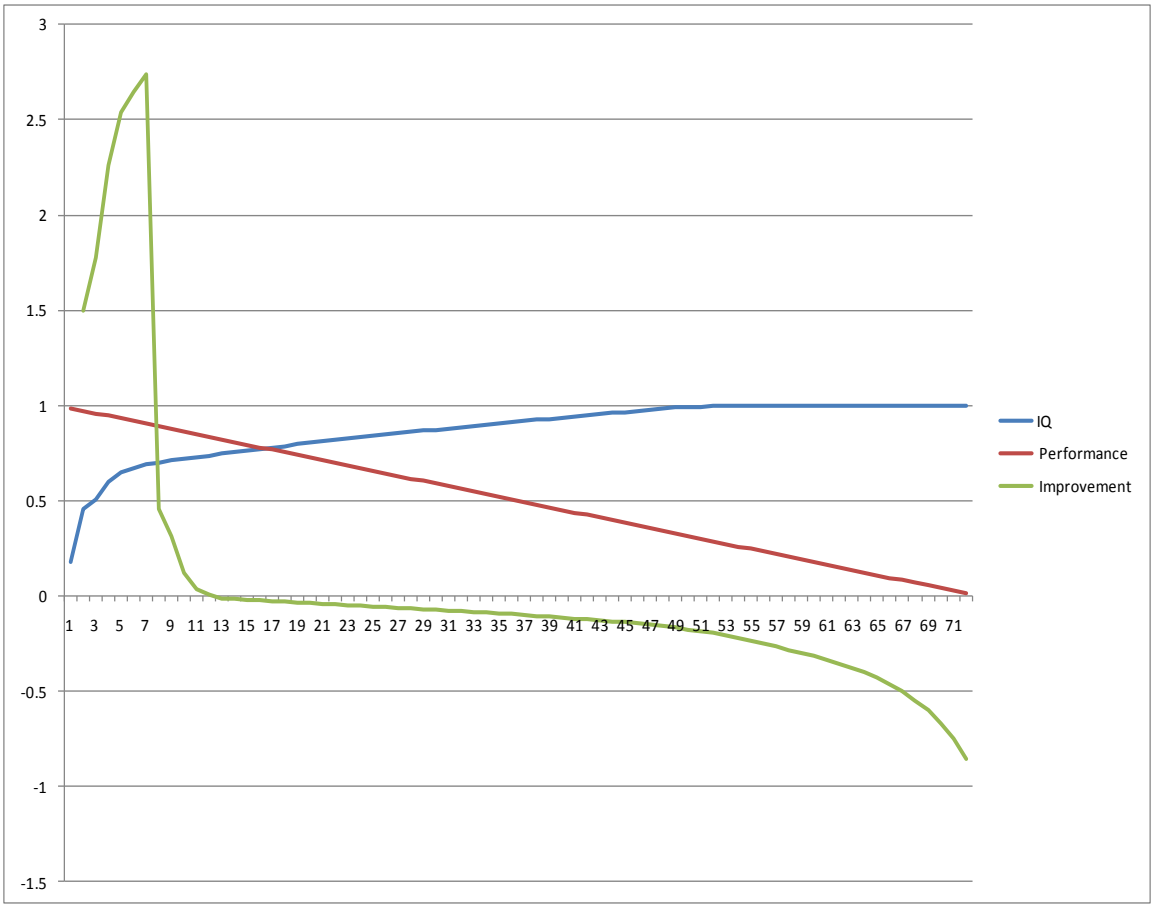

Figure 5.4 A graph showing the improvement gain at each step based on 5.12. When improvement is < 0, the algorithm would stop and return the reduced ontology.

5.2.2 Experiment #2 Comparing Ranking Methods

5.2.2.1 Experiment Description

At each step in the DROG algorithm, there were a large number of nodes that could be added to the reduced ontology. The optimal choice would be a brute force method to compute the IQ and PG changes for each node and select the best improvement. While it always makes the best choice at each step, it takes a long time to run and doesn't make sense in an online, dynamic algorithm.

Since this isn't a reasonable choice, we choose two other possible selection methods, degree centrality and proximity. In this experiment, these two methods and the

brute force method for the DROG algorithm were implemented and applied to a sample ontology. Based on the results we determined which selection method was closest to the optimal and chose that as our ranking/selection method.

5.2.2.2 Brute Force Selection

In order to create a baseline for selection methods, the brute force method was implemented and run as part of the DROG algorithm. A sample of the improvement seen with the brute force method is shown in Figure 5.5. It should be noted that this test took approximately 34 minutes to run on a 72 node ontology. The star marks where the online DROG algorithm would stop and return the reduced ontology.

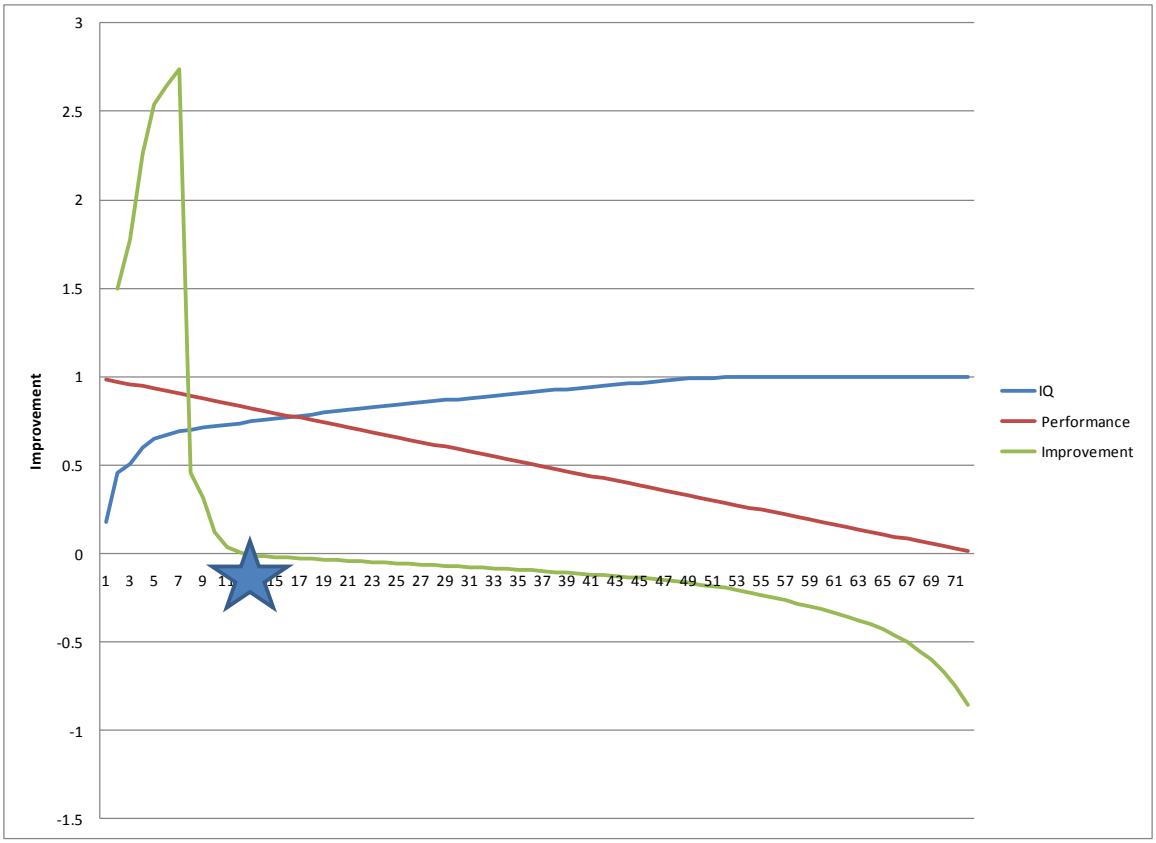

Figure 5.5 Improvement seen at each step using the brute force method for selecting candidate nodes. This is the optimal choice for adding new nodes, but it took approximately 34 minutes to run on a 72 node ontology.

The brute force method is the baseline method so it was expected to outperform the other methods. The improvement curve reached the stopping point at step 14. The IQ was 0.75 and the PG was 0.83, yielding a reduced ontology that was 17% of the original's size.

5.2.2.3 Proximity Ranking

The proximity method is based on the assumption that nodes close to the COI are more important to it than nodes that are far away. It ranks possible nodes by the shortest path length between the COI and the node. The top three are then selected and the IQ and PG are computed and the highest one is added to the reduced ontology. A sample of the result for proximity is shown in Figure 5.6. The proximity method was expected to achieve the best results. It reached the stopping point at step 23 and had an IQ of 0.75 and a PG of 0.69, yielding a reduced ontology only 31% of the original's size.

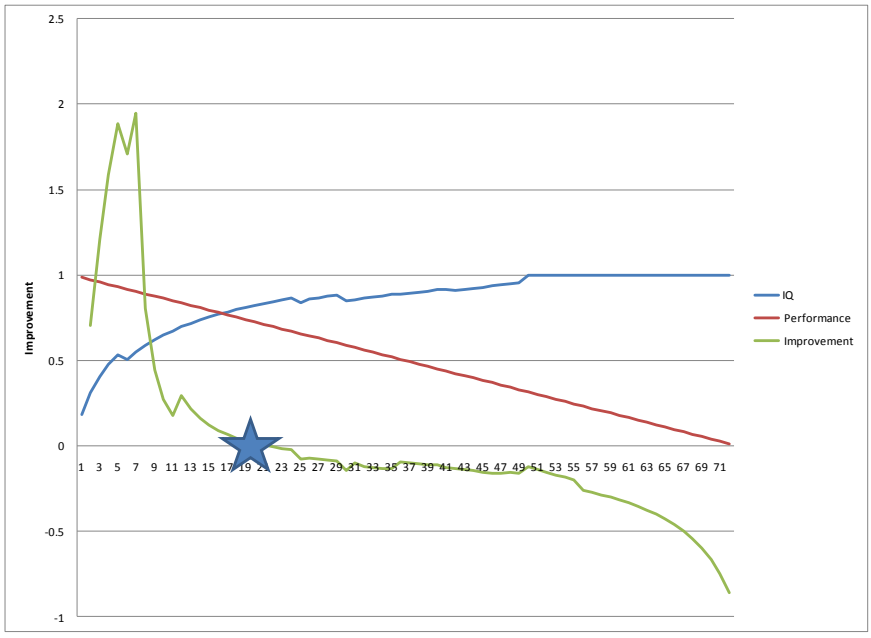

Figure 5.6 Improvement seen at each step using the proximity method for selecting candidate nodes. Notice that this method takes more steps to reach the stopping point. It was, however, an order of magnitude faster than the brute force method.

5.2.2.4 Degree Centrality Ranking

The degree centrality method ranks nodes by the number of edges it contains. In the context of an ontology, it is the number of parents, children, and property relationships it has. In this method, we rank each possible node by it degree centrality and chose the top three to examine. IQ and PG are computed for these three and the highest one is added to the reduced ontology. A sample of the result for degree centrality is shown in Figure 5.7. The degree centrality method reached the stopping point at step 15 and had an IQ of 0.74 and a PG of 0.81, yielding a reduced ontology only 19% of the original's size.

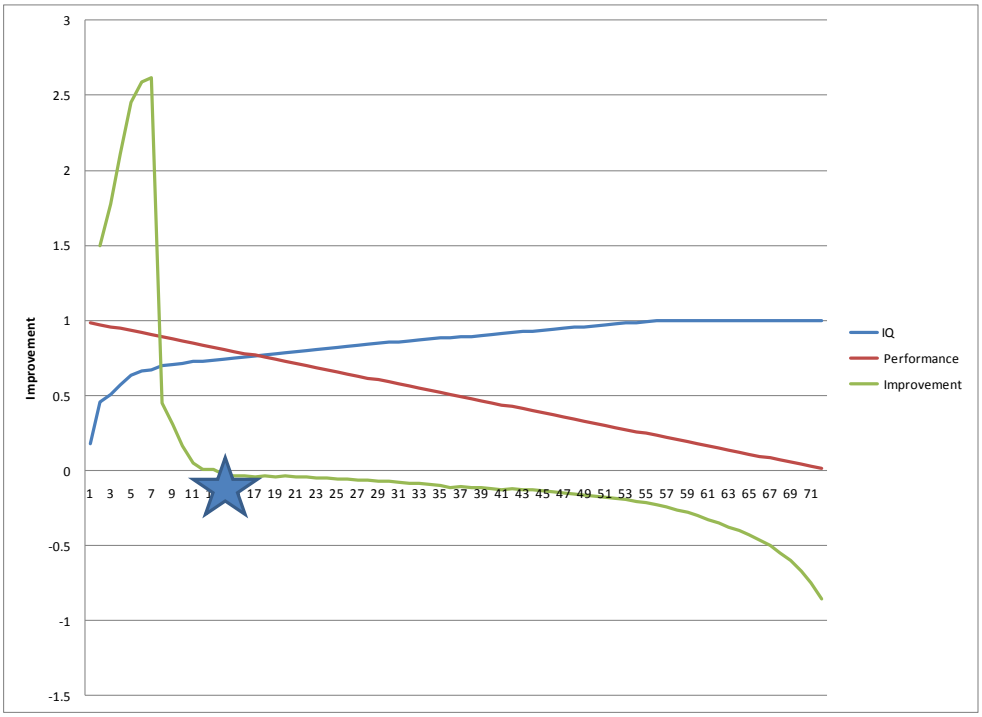

Figure 5.7 Improvement seen at each step using the centrality method for selecting candidate nodes. Notice that this method is very similar to the brute force method. It was also an order of magnitude faster than the brute force method.

5.2.2.5 Wholesale Selection

Another option to select nodes that was considered was to add all nodes within one path length of the current reduced ontology. Each step in the algorithm would then add a larger number of nodes reducing the number of times we had to compute the IQ and PG. It is, however, not intelligent about which nodes it adds. The results, as seen in Figure 5.8, are interesting. Its final result had an IQ of 1 and a PG of .42, or 58% of the original ontology size.

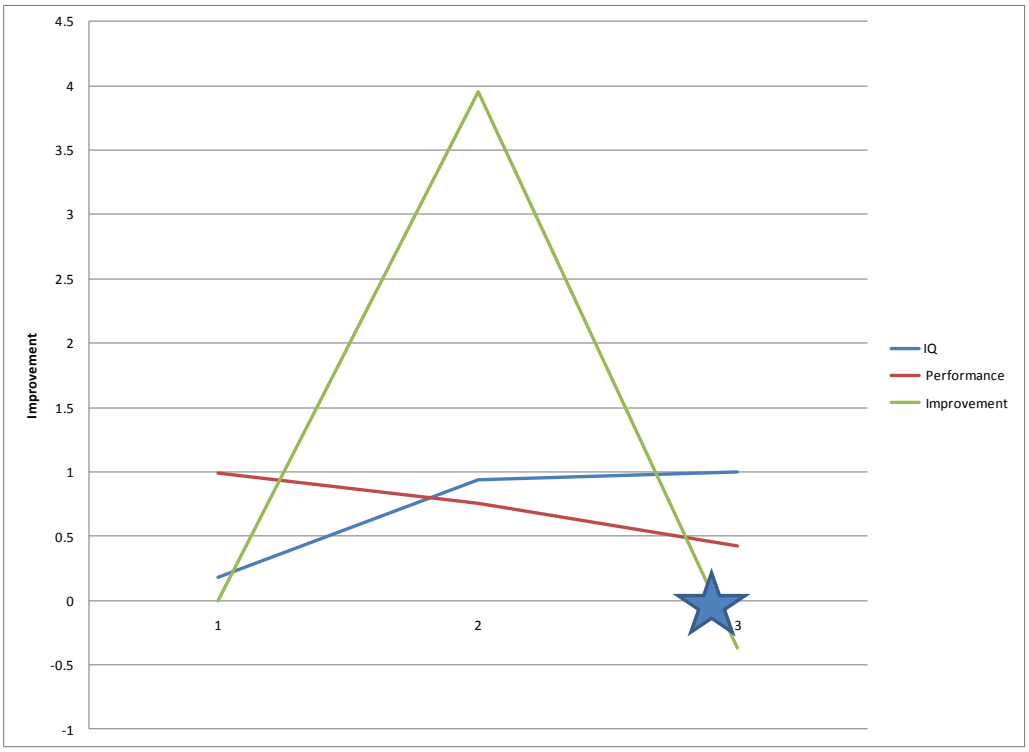

Figure 5.8 Improvement seen at each step using the wholesale method for selecting candidate nodes. Notice that this method finished in three steps. It ran past the optimum point, though. It ran in the 100 ms range, four orders of magnitude faster than the brute force method.

5.2.2.6 Discussion

We expected before this experiment was conducted that proximity would give the closest results to the brute force method. This was not the result seen, however. As is

evident from Table 5.3, degree centrality gave the closest result to brute force. The reason for this result, while not intuitive, seems to be that adding high degree nodes early on allows the algorithm to get to more important nodes sooner in the process.

Table 5.3 Comparison of the Results of the Selection Methods. Proximity and Degree centrality are both close to the brute force method on IQ but degree centrality is much better on PG.

Selection Method	IQ	IQ vs. BF	PG	PG vs. BF
Brute Force	0.75	100%	0.83	100%
Proximity	0.75	100%	0.69	83%
Degree Centrality	0.74	99%	0.81	98%
Wholesale	1.0	125%	0.42	58%

The wholesale method was also very interesting. It found a reduced ontology very fast, four orders of magnitude faster than the brute force algorithm. It goes past the optimum point, though, giving a much larger reduced ontology than necessary. We believe that this is because of its lack of granularity. The wholesale method, however, turns out to be very useful to the DROG algorithm as seen in the next section.

5.2.3 Experiment #3 Hybrid Method to Select Candidate Nodes

5.2.3.1 Experiment Description

Combining the speed of the wholesale method with the granularity of the other methods pointed to a possible hybrid algorithm that could perform better than either alone. The idea was to allow the wholesale algorithm to run until its improvement became less than zero, meaning it has past the optimum point. It then backs up one path length and runs one of the granular algorithms from the previous point until it finds where the improvement crosses the zero line. The general algorithm is given in Table 5.4.

Table 5.4 Notional View of Hybrid DROG Algorithm

```
1   AddNewNodes(Ontology o, COI c)
2   Returns Reduced Ontology ro
3           Add node that matches c to ro
4           //First run Wholesale
5           While improvement to IQ better than decrease in PG
6                   Add all nodes within 1 path-length of current RO
7                   Calculate IQ and PG
8           Remove nodes added for last round
9           //Then run granular algorithm
10          While improvement to IQ better than decrease in PG
 11             Find best node in o to add within 1 path-length of any nodes in ro
12              Add new node to ro, including any new properties and individuals
13                  Calculate IQ and PG
14          End while
15  Return ro
```

We ran the combination of the wholesale and each of the granular algorithms. Again we found degree centrality to be the best choice again. It took ten stops and ran in less than a second. As shown in Figure 5.9, it had a final IQ of 0.98 and a PG of .63, yielding a reduced ontology 37% of the original size. This is the selection method chosen for the full DROG experiment given in the next section.

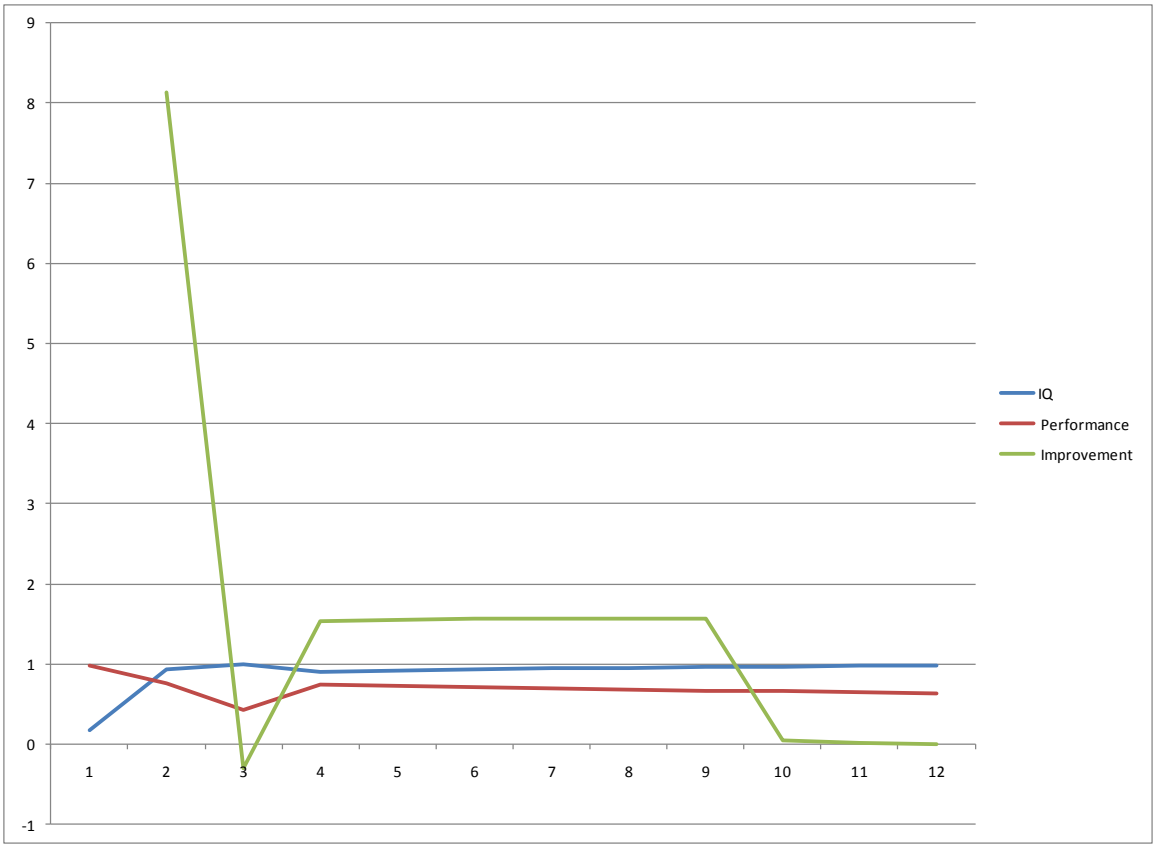

Figure 5.9 Hybrid Centrality algorithm took 10 steps and less than one second to run.

5.2.4 Experiment #4 DROG Validation

5.2.4.1 Experiment Description

Once the details of the algorithm have been selected, a larger study was conducted on twenty potential web service ontologies in four domains, travel, organizations, food, and zoology. A set of queries was applied to the original ontologies to create a baseline. The algorithm was used to create a reduced ontology for each ontology. Then, the same queries used for each original ontology were applied to the reduced ontology and recall computed for each query. These recall scores were averaged together to get a mean average recall score for the reduced ontology. Recall ranges from 0.0 to 1.0, with 0.0 meaning no recall and 1.0 meaning perfect recall.

The queries were created in a previous study, described in Chapter Four. These queries using sample queries for each domain generated by various people unaffiliated with this study. There were seven queries for each ontology, or 98 queries in all.

5.2.4.2 Results

The results of the experiment are shown in Table 5.5. Each ontology shows its wall clock time, predicted IQ and PG scores, and its minimal and maximum recall scores. The averages over all ontologies is shown at the bottom.

Table 5.5 Results of Algorithm Runs on Fourteen Ontologies in Four Domains.

Ontology	Algorithm Time (ms)	IQ	PG	Recall Min	Recall Max
Animal	156	1.00	0.33	1.00	1.00
animals	687	1.00	0.78	1.00	1.00
biosphere	422	1.00	0.85	1.00	1.00
cno	11656	1.00	0.70	1.00	1.00
food	7016	1.00	0.83	1.00	1.00
hu	625	1.00	0.30	1.00	1.00
MscProgram	52877	0.99	0.04	0.93	1.00
pet	3422	1.00	0.74	1.00	1.00
pizza	1219	1.00	0.88	1.00	1.00
pizzaToppings	13624	1.00	0.52	1.00	1.00
restaurant	23778	1.00	0.76	1.00	1.00
travelAgency	42314	0.89	0.63	0.25	1.00
TravelMessageOntology	1031	1.00	0.40	1.00	1.00
travelontology	458591	0.84	0.64	0.00	1.00
Average	**44101.29**	**0.98**	**0.60**	**0.87**	**1.00**

5.3 Discussion

Given the results of this experiment, the DROG algorithm generated useable reduced ontologies, on average 60% of the size of their original ontologies. These reduced ontologies provided good results to the user in most cases, with an average IQ (recall score) of 0.98. There are a couple of oddities, where recall is low (travelAgency

and travelontology). It may be that there is an issue with the travel domain that the predictive model does not address. A possible area of future research could be looking at more domain specific predictive models.

CHAPTER SIX

END-TO-END TESTING OF DYNAMIC REDUCED ONTOLOGY GENERATION WITH A SIMULATED SEMANTIC WEB SERVER AND MOBILE DEVICES

This chapter describes the third phase of our research. Previously, in Chapter Five, the Dynamic Reduced Ontology Generation (DROG) algorithm was developed and tested on a series of small to medium sized ontologies to prove it can build reduced ontologies that contain the same information content as the original ontology. To finish up this research, an end-to-end simulation of a semantic web server and mobile device making a request was developed. This test will demonstrate DROG's ability to significantly decrease the download and processing time required for an ontology while retaining the information the user requires.

6.1 Research Description

This final experiment consisted of three parts. First we developed a simulated web server using a blackboard architecture. This framework accepted requests from a user, executed the DROG algorithm as an agent, and read the results from the blackboard when the algorithm finished. Second, we executed Monte Carlo runs of the simulation using RIM's Blackberry simulation for PC across currently available networks for the small and medium sized ontologies used in previous experiments. The networks examined included 2G, EDGE, 3G, and Wi-Fi.

Finally we chose a large, publicly available ontology, Word Net, and executed it in the same environment. This last experiment is representative of the application of DROG to a real world situation.

6.1.1 Semantic Web Server Architecture

The simulated semantic web server proof-of-concept is built on the blackboard architecture discussed in Chapter Three. It consists of a controller, which receives requests from the user and manages the blackboard and the DROG algorithm, a blackboard, which is a temporary storage area for the DROG algorithm to save its intermediate results, and the DROG algorithm.

6.1.2 Blackberry Simulation

RIM's software development kit provides a library of mobile device simulations that run on a PC. A Blackberry Torch was chosen for this experiment due to its current hardware and support for all four network types. Figure 6.1 shows a screenshot of the setup.

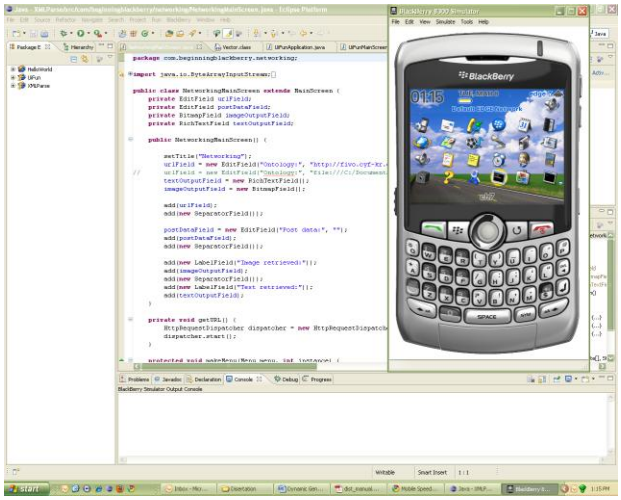

Figure 6.1 The RIM SDK development kit integrated with the Eclipse IDE

Four major mobile speeds were simulated in this experiment. Table 6.1 shows a list of typical download speeds for the obtained from real world tests at mobile speed test [64].

Table 6.1 A List of Typical Download Speeds for each Type of Network used in this Test

Network	Average Data Rate (kbps)
2G	40 kbps
EDGE	236.8 kbps
3G	400 kbps
WiFi	1500 kbps

6.1.3 Small to Medium Ontologies

We first used the twenty ontologies from four domains to validate that the end-to-end framework successfully allowed a simulated mobile device to request a reduced ontology and process the queries. We did not expect DROG to make a significant difference in these tests to the user since these were relatively small ontologies and not representative of the cases where DROG would be most effective.

Each of the ontologies was downloaded and queried using the queries described in Chapter Four. Then the DROG algorithm was run on each ontology and the reduced ontology was returned to the user. These reduced ontologies were queried and the total wall clock time was recorded.

6.1.4 Large Ontology

For the large ontology we chose Word Net, version 1.7.1 [65]. The OWL file is 63 megabits, including over 770,000 concepts, so it is an ontology that mobile devices would have great difficulty in downloading and processing, even on a high bandwidth network.

The format for the experiment was the same as the validation test. The original ontology was downloaded and processed using queries generated by volunteers who did not possess knowledge of the ontology's specific structure. Then a reduced ontology was generated by DROG, downloaded and processed. The recall value and wall clock time were recorded. A final test with the large ontology was conducted using the RIM Blackberry simlulator on all four networks.

6.2 Results

6.2.1 Validation Test

To validate the framework, we first applied the ontologies used in previous studies. Each ontology was downloaded and queried using the RIM blackberry simulator on 2G, EDGE, 3G, and WiFi networks. The total time was computed using download time and query time. Then DROG was used to create the reduced ontology, and again it was downloaded and queried. In this case, the total time was the overhead time required by DROG, the download time, and the query time. Figure 6.2 shows the comparison of total time on each network for each of the ontologies.

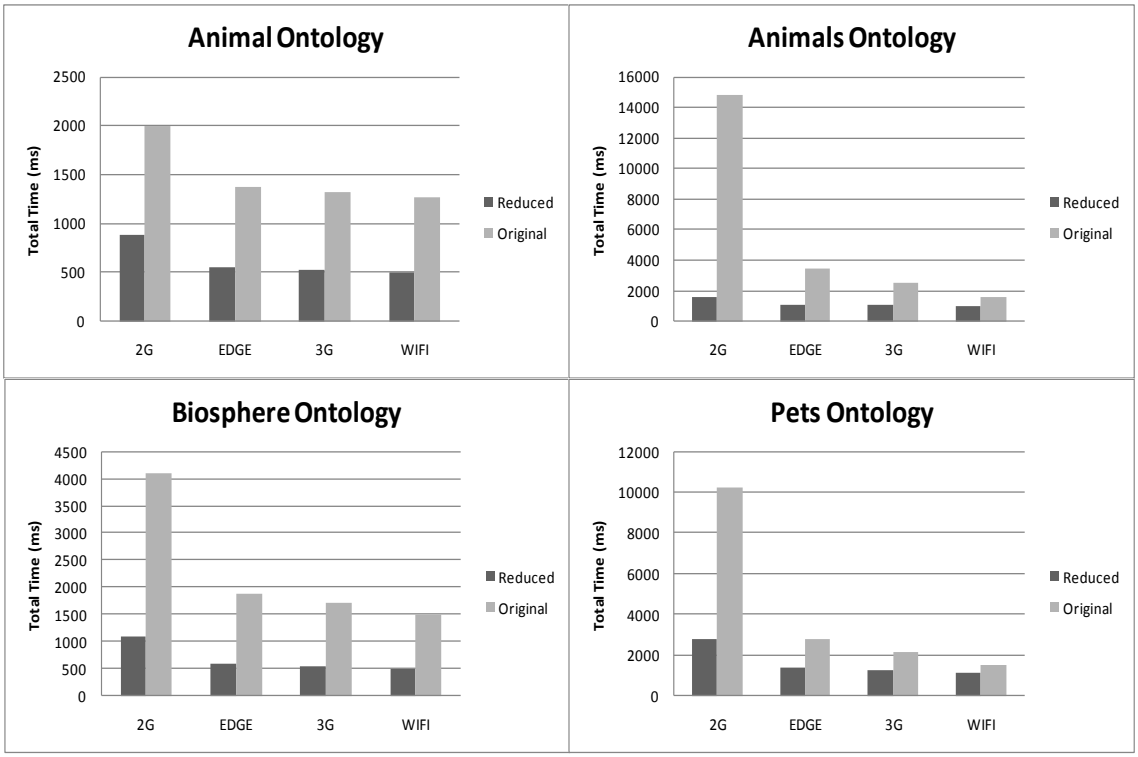

Figure 6.2(a) - Comparison total time for end-to-end DROG runs of four medium sized ontologies in the animal domain.

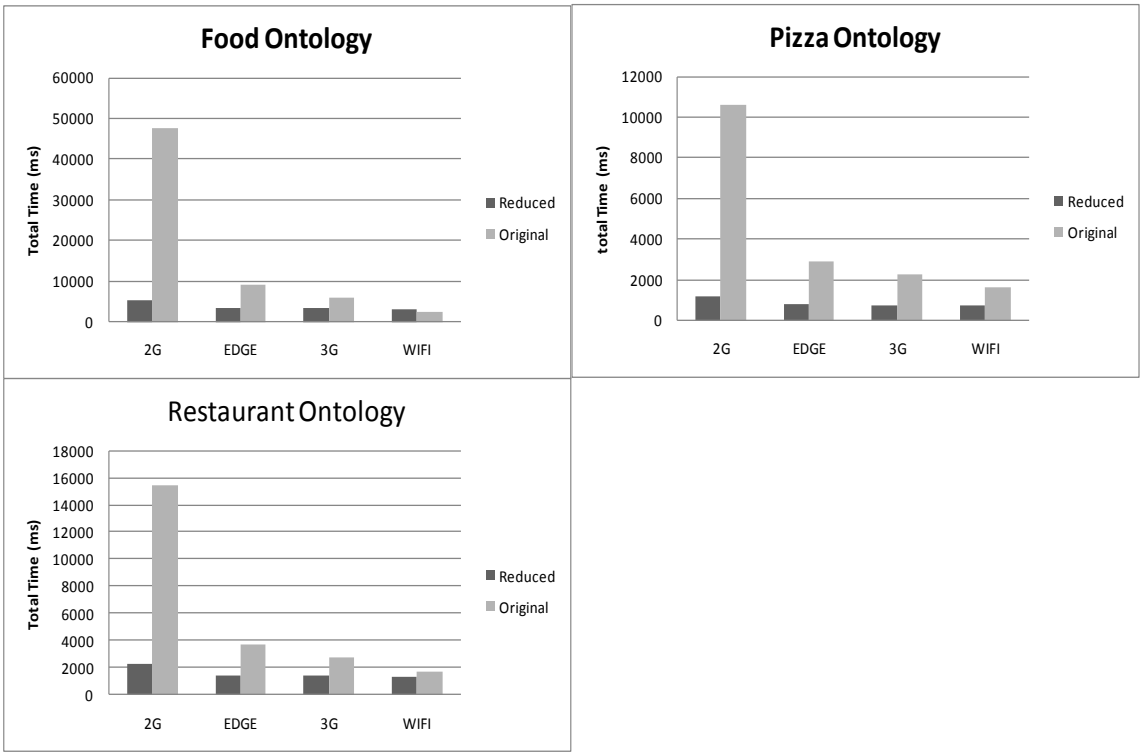

Figure 6.2 con't (b) - Comparison total time for end-to-end DROG runs of three medium sized ontologies in the food domain.

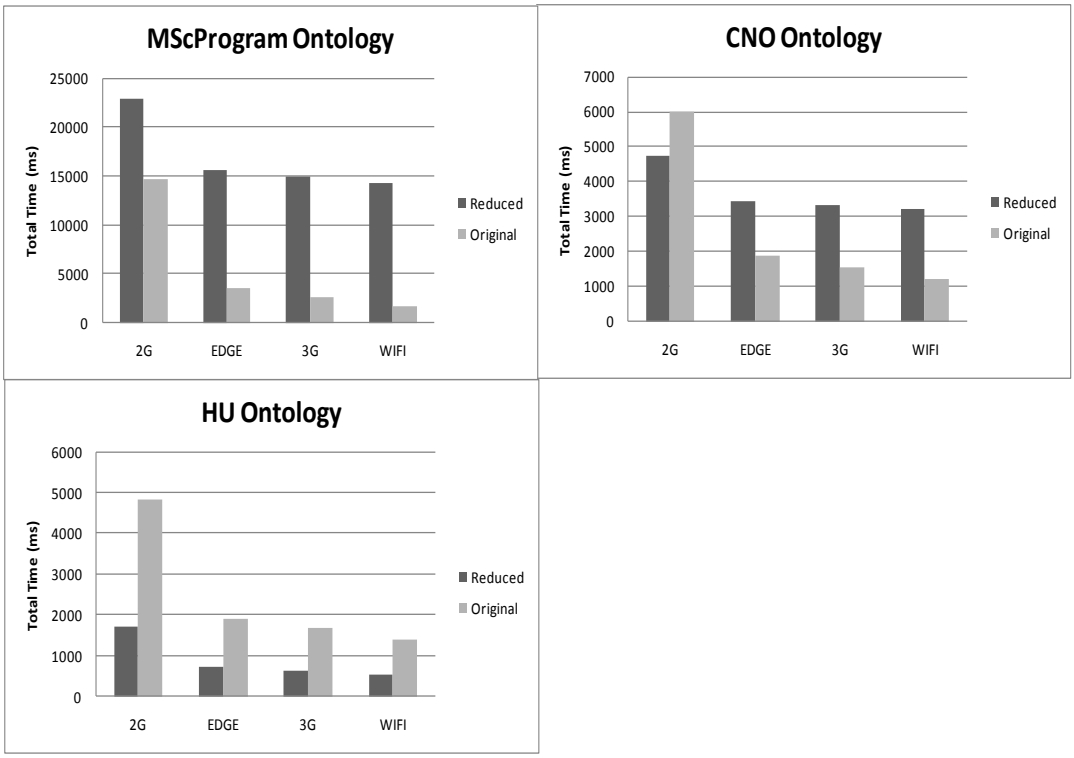

Figure 6.2 con't(c) - Comparison total time for end-to-end DROG runs of three medium sized ontologies in the organizational structure domain.

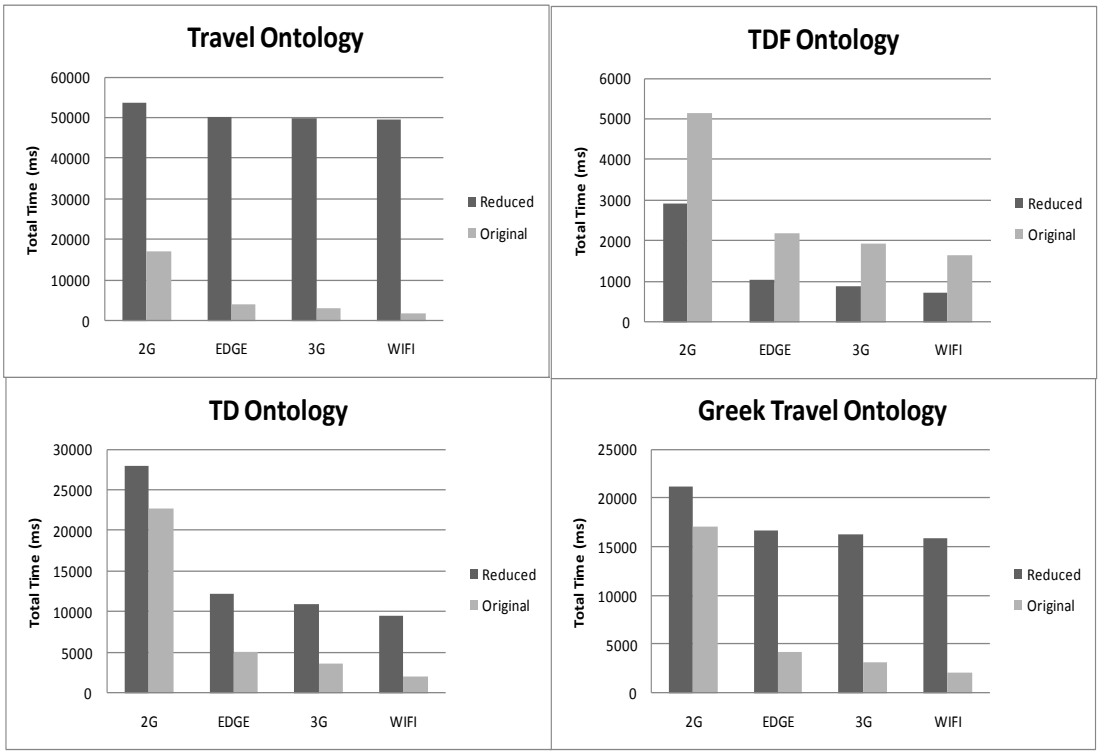

Figure 6.2 con't (d) - Comparison total time for end-to-end DROG runs of three medium sized ontologies in the travel domain.

The total time for nine of the sixteen reduced ontologies was less than the required total time for the original ontologies for all networks. In ten cases, the reduced ontology total time was less in at least one of the networks.

The reason the reduced ontologies were not always better than the original is due to the size of these validation ontologies. If an ontology was too small, the overhead time of running the DROG algorithm outweighed the download and query time benefits. Once we were sure of the functionality of the blackboard simulation, we applied DROG to a large, representative ontology, WordNet.

6.2.2 Representative Validation Test

To test the algorithm in the framework, we applied the Wordnet 1.7.1 ontology [65]. A COI of "adverb" was selected and six queries related to "adverb" were created. This ontology was downloaded and queried using the RIM blackberry simulator on 2G, EDGE, 3G, and Wi-Fi networks. The total time was computed from summing download time and query time. Then DROG was used to create the reduced ontology, and again it was downloaded and queried. In this case, the total time was the overhead time required by DROG added to the download time, and the query time. Figure 6.3 shows the comparison of total time on each network for the ontology.

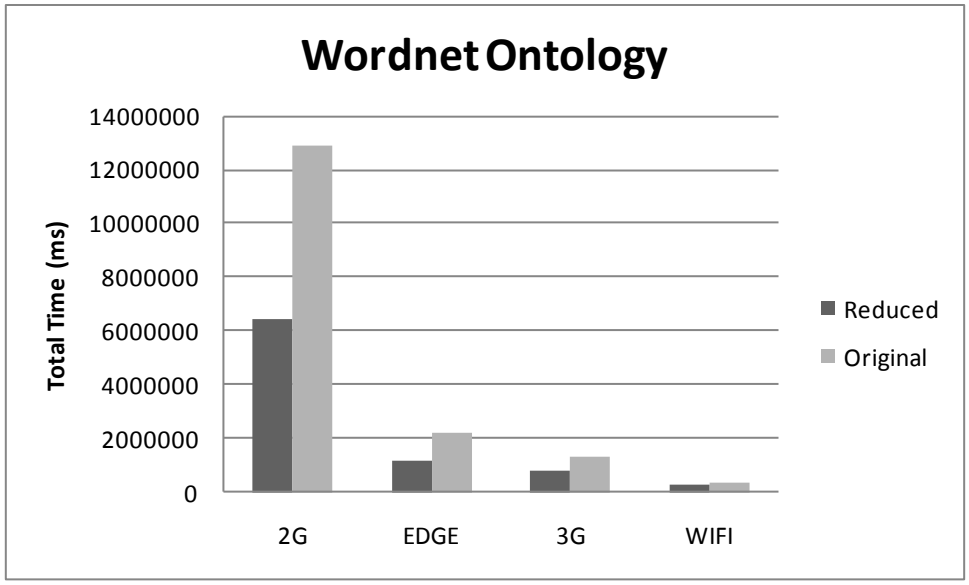

Figure 6.3 Comparison total time for end-to-end DROG runs of Wordnet ontology

In every network, the total time for the reduced ontology was less than the time to download the original. It is true that as the network became faster, the advantage of the reduced ontology was lessoned. However, we believed that if DROG and the OWL parsing library were optimized, the overhead time could be diminished and a greater increase in performance could be achieved.

6.2.3 Real Word Test

Given the large size of the Wordnet ontology, we decided to attempt to download it manually using a real mobile device. We chose the RIM Blackberry Torch 9800 using a Wi-Fi connection to the internet [65]. This was not a test of the DROG algorithm, but rather a sanity check of whether the ontology could even be downloaded and stored by a high-end mobile device.

To conduct the test we manually typed in the address of the Wordnet OWL file in the Web-Kit browser standard on the Blackberry Torch. After waiting five minutes,

approximately 1/3 of the 63 MB file was downloaded and the device froze with the error shown in Figure 6.4.

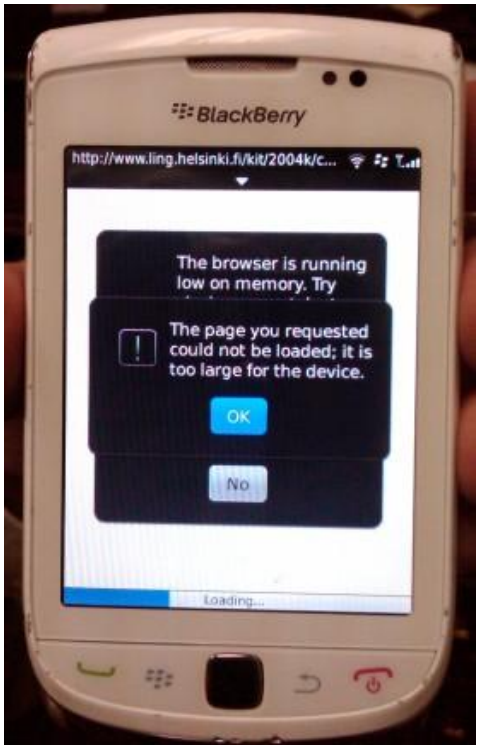

Figure 6.4 Results of downloading the Wordnet ontology on a RIM Blackberry Torch 9800 using Wi-Fi.

While this was a single example it was obvious that original ontology was too large to be downloaded to a typical mobile device, much less queried and processed. We attempted the same procedure on the reduced ontology hosted on a local web server and found the Blackberry Torch was able to handle downloading the reduced ontology. The results can be seen in Figure 6.5.

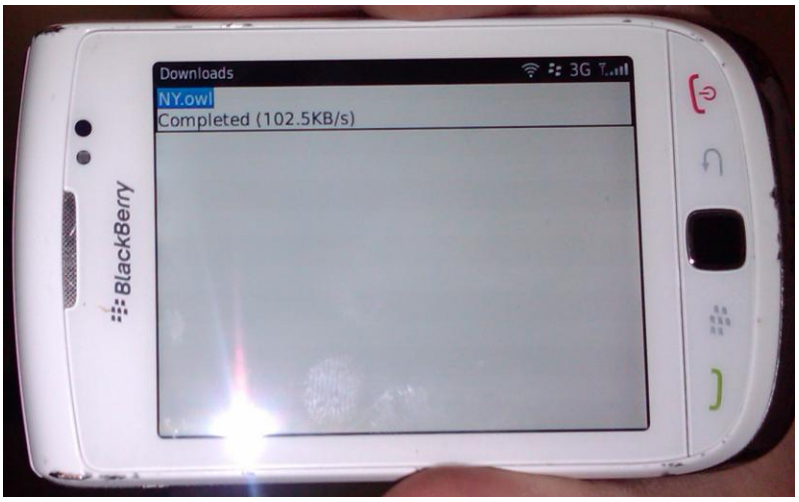

Figure 6.5 Results of downloading the reduced Wordnet ontology on a RIM Blackberry Torch 9800 using Wi-Fi.

6.3 Discussion

With this final experiment we have demonstrated the performance advantage of downloading and using a reduced ontology generated by DROG comparing to downloading and using the original ontology. For very large ontologies, it has been illustrated that the reduced ontology is likely to be the only one that can be handled by a mobile device; that is, some very large ontologies are too large for one of today's mobile devices (in this case, a Blackberry Torch) to handle.

It is apparent that the reduced ontology was not always faster, due the overhead time, but even in those cases the reduced ontology may have an advantage. Many wireless carriers charge a fee per byte of information on their broadband networks. Even in cases where the original ontology is downloaded and processed faster, the reduced ontology would offer a cost savings by requiring the user to download fewer bytes. Table 6.2 shows a comparison of cost for each ontology tested in this experiment. In every case the reduced ontology is cheaper than the original.

Table 6.2 A Comparison of Cost of Downloading Reduced and Original Ontologies at a Typical Rate of $0.01 KB.

Ontology	$/Reduced	$/Original
Animal	$ 0.02	$ 0.04
animals	$ 0.03	$ 0.68
Biosphere	$ 0.03	$ 0.13
CNO	$ 0.08	$ 0.25
Food	$ 0.11	$ 2.33
Greek Travel	$ 0.27	$ 0.77
HU	$ 0.06	$ 0.18
MscProgram	$ 0.44	$ 0.67
Pets	$ 0.08	$ 0.45
Pizza	$ 0.02	$ 0.46
Restaurant	$ 0.05	$ 0.71
TD	$ 0.95	$ 1.06
TDF	$ 0.11	$ 0.18
travel	$ 0.02	$ 0.78
Wordnet	$ 314.14	$ 645.12

CHAPTER SEVEN

CONCLUSIONS AND FUTURE RESEARCH DIRECTIONS

The overall goal of this research is to allow mobile devices to fully exploit the abilities of semantic web services despite their resource limitations compared to a traditional computer. To do this, we developed an algorithm, based on easily computed metrics predicting information quality and performance gain, to generate a good reduced ontology useable by a client with a resource constrained mobile device. We tested this algorithm in a simulated web service environment, comparing total overhead, download, and processing time of the reduced ontologies compared to their original counterparts. This research has shown that metrics can be used to reduce a large ontology into a more manageable sub-set based on the users needs.

As with all areas of research, it is likely that the solution chosen for this research may be improved upon. It may also be the case that different kinds of ontologies lend themselves to different kinds of reduction algorithms or that certain algorithms run better in a very short time while others run better given a longer time. For these reasons, as part of this research, and to allow expansion to future research, we created a test-bed architecture that allows new reduced ontologies to be created and to handle time constraints given by the client.

The test-bed itself is not limited to reduced ontologies. It could be used to perform a number of operations on an ontology before it goes out to a client. It is also not limited to semantic web services and could be used to preprocess knowledge bases dynamically.

The concepts used in DROG have not been fully explored. It is possible that the overhead time could be reduced to gain increased performance. Post-list creation from information retrieval might offer a way of cashing reduced ontologies based on common requests. Another possible performance enhancement to DROG is allowing users to negotiate a time limit with the controller, putting a hard stop on how long the DROG algorithm runs.

Using more semantically based metrics may offer better reduced ontologies. Using Wordnet's semantic distance between concepts on the COI may give a better ranking method than degree centrality. Also, allowing multiple concepts in the COI may allow users to do more complex processing than the current implementation of DROG.

There are a number of ways this research could be expanded, but we believe we have made a good first pass. Mobile devices and the Semantic web are only going to grow in use over time, and it is important that we as computer science researchers focus on how these two powerful technologies can be integrated.

REFERENCES

[1] Alani, H. & Brewster, C. 2006. Metrics for Ranking Ontologies. In *4th Int. EON Workshop, 15th Int. World Wide Web Conference (WWW2006)* Edinburgh, UK.

[2] Amato,C., Esposito, F., Fanizzi, N., Fazzinga, B., Gottlob, G., & Lukasiewicz, T. 2010. Inductive Reasoning and Semantic Web Search, In *Proceedings of the 2010 ACM Symposium on Applied Computing*, Sierre, Switzerland.

[3] Botti, V., Barber, F., Crespo, A., Onaindia, E., Garcia-Fornes, A., Ripoll, I., Gallardo, D., & Hernndez, L. 1995. A Temporal Black-board for Multi-agent Environment. *Data and Knowledge Engineering* **15**, 189-211.

[4] Cellular-News. 2007. *Converged Mobile Devices Adoption to Reach 82 Million Units by 2011]* http://www.cellularnews.com/story/24073.php

[5] Cohen, J. 1998. *Statistical Power Analysis for the Behavioral Sciences*, 2nd edn. Lawrence Erlbaum Publishing Company.

[6] Crubezy, M. & Musen, M. 2004. Ontologies in Support of Problem Solving. In *Handbook on Ontologies*.

[7] The DARPA Agent Markup Language Homepage. 2005. *Semantic Web Services Ontology (SWSO) version 1]0* http://www.daml.org/services/swsf/1]0/swso

[8] Deng, X., Haarslev, V., Shiri, N., Franconi, E., Kifer, M., & May, W. (ed.) 2007. Measuring Inconsistencies in Ontologies. *Lecture Notes in Computer Science* **4519**, 326-340.

[9] Earman, D., Hayes-Roth, F., Lesser, V., & Reddy, D. 1980. The HEARSAY-II Speech Understanding System: Integrating Knowledge to Resolve Uncertainty. *ACM Computing Survey* **12**, 213-253.

[10] Etzkorn, L., Gholston, S., Fortune, J., Stein, C., Utley, D., Farrington, P., & Cox, G. 2004. A Comparison of Cohesion Metrics for Object-oriented Systems *Information and Software Technology* **46**, 667-687.

[11] Gangemi, A., Catenacci, C., Massimiliano, C., & Lehmann, J. 2005. A Theoretical Framework for Ontology Evaluation and Validation. In *Proceedings of Semantic Web Applications and Perspectives (SWAP2005)* Trento, Italy.

[12] Gangemi, A., Catenacci, C., Ciaramita, M., and Lehmann, J. 2005. Ontology Evaluation and Validation: An Integrated Formal Model for the Quality Diagnostic Task. *Technical report, Laboratory of Applied Ontologies - CNR, Rome, Italy* http://www.loacnr.it/Publications.html

[13] Gangemi, A., Catenacci, C., Ciaramita, M., & Lehmann, J. 2005. Modelling Ontology Evaluation and Validation. In *Proceedings of the 3rd European Semantic Web Conference (ESWC2006)* Budva, Montenegro.

[14] Gangemi, A., Catenaccia, C., Ciaramita, M., & Lehmann, J. 2006. Qood grid: A Metaontology Based Framework for Ontology Evaluation and Selection. In *4th Int. EON Workshop, 15th Int. World Wide Web Conference (WWW2006)* Edinburgh, UK.

[15] Grau, B., Parsia, B., Sirin, E., & Kalyanpur, A. 2006. Modularity and Web Ontologies. In *Proceedings of the 10th International Conference on Principles of Knowledge Representation and Reasoning (KR-06)* Lake District of the United Kingdom.

[16] Grau, B., Kazakov, Y., & Sattler, U. 2007. Just the Right Amount: Extracting Modules from Ontologies. In *Proceedings of the 16th International Conference on World Wide Web (WWW2007)* Banaff, Canada.

[17] Gruber, T. 1993. A Translation Approach to Portable Ontology Specification. *Knowledge Acquisition* **5**(2), 199-220.

[18] Guarino, N. & Welty, C. 2004. Evaluating Ontological Decisions with ONTOClean *Communications of the ACM* **45**(2), 61-65.

[19] Hull, R. 2005. Web Services Composition: A Story of Models, Automata, and Logics. In *Proceedings of the IEEE International Conference on Services Computing (SCC2005)* Orlando, FL.

[20] Hunter, A. 2004. Logical Comparison of Inconsistent Perspectives using Scoring Functions. *Knowledge and Information Systems* **6**(5), 528-543.

[21] Jimnez-Ruiz, E., Nebot, V., Berlanga, R., Sanz, I., & Rios, A. 2007. A Protege Plug-in-Base System to Anage and Query Large Domain Ontologies. In *Proceedings of 10th International Protg Conference* Budapest, Hungary.

[22] Jimnez-Ruiz, E., Berlanga, R., Nebot, V. & Sanz, I. 2007. OntoPath: A Language for Retrieving Ontology Fragments. *Lecture Notes in Computer Science* **1**, 897-914.

[23] Kusnierczyk, W. 2008. Taxonomy-based Partitioning of the Gene Ontology. *Journal of Biomedical Informatics* **41**, 282-292.

[24] Lacy, Lee W. 2005. Chapter 9 - RDFS. *OWL: Representing Information Using the Web Ontology Language* Trafford Publishing.

[25] Lalanda, P., Charpillet, F., & Haton, J. 1992. A Real-time Blackboard based Architecture. In *Proceedings of the Tenth European Conference on Artificial Intelligence (ECAI1992)* Vienna, Austria.

[26] Lozano-Tello, A. & Gomez-Perez, A. 2004. OntoMetric: A Method to Choose the Appropriate Ontology. *Journal of Database Management* **15**, 1-18.

[27] Manning, C., Raghavan, P., & Schtze, H. 2006. *An Introduction to Information Retrieval* Cambridge University Press.

[28] Martin, D., Paolucci, M., McIlraith, S., Burstein, M., McDermott, D., McGuinness, D., Parsia, B., Payne, T., Sabou, M., Solanki, M., Srinivasan, N., & Sycara, K. 2004. Bringing Semantics to Web Services: The OWL-S Approach. In *Proceedings of the First International Workshop on Semantic Web Services and Web Process Composition (SWSWPC 2004)* San Diego, CA.

[29] Mobile Marketer. 2009. *Daily Mobile Web Consumption of News, Information Doubles:comScore* http://www.mobilemarketer.com/cms/news/research/2842.htm

[30] Montgomery, D., Peck, E., & Vining G. 2006. *Introduction to Linear Regression Analysis*. Wiley-Interscience.

[31] Nii, H. 1986. The BlackboardModel of Problem Solving and the Evolution of Blackboard Architectures. *AIMagazine* **7**(2), 38-53.

[32] Noy, N. & Musen, M. 2003. The PROMPT Suite: Interactive Tools for Ontology Mapping and Merging. *International Journal of Human-Computer Studies* **6**(59)

[33] Noy, N. & Musen, M. 2004. Specifying Ontology Views by Traversal. In *Proceedings of the Third International Semantic Web Conference (ISWC2004)* **3298**, 713-725.

[34] Ontology Portal. 2009. *Suggested Upper Merged Ontology (SUMO)*. http://www.ontologyportal.org/

[35] Orme, A., Yao, H., & Etzkorn, L. 2006. Coupling Metrics for Ontology-Based Systems *IEEE Software* **23**(2), 102-108.

[36] Orme, A., Yao, H., & Etzkorn, L. 2009. Complexity Metrics for Ontology Based Information *International Journal of Technology Management* 47:1-2 p. 161-173.

[37] Orme, A., Yao, H., & Etzkorn, L. 2007. Indicating Ontology Data Quality, Stability, and Completeness Throughout Ontology Evolution. *Journal of Software Maintenance and Evolution* **19**(1), 49-75.

[38] Qi, G. & Hunter, A. 2007. Measuring Incoherence in Description Logic-based Ontologies. In *Proceedings of the 6th International Semantic Web Conference (ISWC2007)* Busan, Korea.

[39] Schrimpsher, D. & Etzkorn, L. 2009. Sub-Graphing Web Service Ontologies to Support Resource Constraints of Mobile Devices. In *Proceedings of the 47nd Annual Association for Computing Machinery Southeast Conference (ACMSE2009)* Clemson, SC.

[40] Schrimpsher, D. & Etzkorn, L. 2009. A Web Service Ontology Sub-graph Quality Model to Support Mobile Devices. In *Proceedings of the 3rd Annual International Symposium on Empirical Software Engineering and Measurement (ESEM2009)* Orlando, FL.

[41] Schrimpsher, D. & Etzkorn, L. 2010. A Model to Predict Quality of a Reduced Ontology for Web Service Discovery on Mobile Devices *Knowledge Engineering Review* (accepted March 17, 2010)

[42] Seidenberg, J. & Rector, A. 2006. Web Ontology Segmentation: Analysis, Classification and Use. In *Proceedings of the 15th international conference on World Wide Web (WWW2006)* Edinburg, UK.

[43] Serafini, L., Borgida, A., & Tamilin, A. 2005. Aspects of Distributed and Modular Ontology Reasoning. In *Proceedings of the International Joint Conference on Artificial Intelligence* Edinburgh, UK.

[44] Sowa, John F. 1999. *Knowledge Representation: Logical, Philosophical, and Computational Foundations* Brooks Cole Publishing Co.

[45] Statistical and Process Management Software For Six Sigma and Quality Improvement - Minitab 2010. *Statistical and Process Management Software For Six Sigma and Quality Improvement - Minitab* [Internet]. State College PA. Minitab Inc. from:¡http://www.minitab.com/en-US/default.aspx¿ [May 7, 2010]

[46] Stuckenschmidt, H. & Klein, M. 2003. Integrity and Change in Modular Ontologies. In *Proceedings of the International Joint Conference On Artificial Intelligence (IJCAI2003)* **18**, 900-908.

[47] Taghva, K., Borsack, J., Coombs, J., Condit, A., Lumos, S., and Nartker, T. 2003. Ontology-based Classification of Email. In *Proceedings of the International Conference on Information Technology: Computers and Communications* Las Vegas, NV.

[48] Takeda, H., Iino, K. & Nishida, T. 1995. Ontology-supported agent communication. Presented at *Working Notes of the AAAI Spring Symposium on Information Gathering from Heterogeneous, Distributed Environments* Standford, CA.

[49] Tartir, S., Arpinar, I., Moore, M. & Sheth, A. 2005. OntoQA: Metric-based Ontology Quality Analysis. In *Proceedings of IEEE Workshop on Knowledge Acquisition from Distributed, Autonomous, Semantically Heterogeneous Data and Knowledge Sources* Houston, TX.

[50] Volz, R., Oberle, D., & Studer, R. 2003. Implementing Views for Light-weight Web Ontologies. In *Proceedings of the Seventh International Database Engineering and Applications Symposium (IDEAS2003)* 160-169.

[51] Vrandecic, D. & Sure, Y. 2007. How to Design Better Ontology Metrics In *Proceedings of European Semantic Web Conference (ESWC2007)* Innsbruck, Austria.

[52] W3C. 2004] *OWL-S: Semantic Markup for Web Services* http://www.w3.]org/Submission/OWL-S/ W3C. 2008. *SPARL Query Language for RDF* http://www.w3.org/TR/rdf-sparql-query/

[53] Yao, H., Orme, A., & Etzkorn, L. 2005. Cohesion Metrics for Ontology Design and Applications *Journal of Computer Science* **1**, 107-113.

[54] Research in Motion. 2011. Blackberry – Smartphone Simulators *http://us.blackberry.com/developers/resources/simulators.jsp.*

[55] OWL-S: Semantic Markup for Web Services, http://www.w3.org/Submission/OWL-S/, 2004.

[56] Semantic Web Services Ontology (SWSO) version 1.0, http://www.daml.org/services/swsf/1]0/swso/, 2005.

[57] Menzies, T. 1999. Knowledge Maintenance: State of the Art *The Knowledge Engineering Review* 14:1, p. 1-46.

[58] Hsi, I., Potts, C., & Moore, M. 2003. Ontological Excavation: Unearthing the Core Concepts of the Application In *Proceedings of 10th Working Conference on Reverse Engineering (WCRE 2003)* p. 345.

[59] Sabou, M., Wroe, C., Goble, C., & Stuckenschmidt, H. 2005. Learning Domain Ontologies for Semantic Web Service Descriptions *Journal of Web Semantics* 3.

[60] Guo, Y., Pan, Z., Heflin, J. 2005. LUBM: A Benchmark for OWL Knowledge Based Systems *Journal of Web Semantics* 3.

[61] Ceruti, M., Wright, T., Wilcox, D., & McGirr, S. 2005. Modeling and Simulation for the Knowledge Management for Distributed Tracking (KMDT) Program *In Proceedings of the International Workshop on Modeling & Applied Simulation* Genova, Italy, p. 67-75.

[62] Kitchenham, B., Pfleeger, S., & Fenton, N. 1995 Towards a Framework for Software Measurement Validation *IEEE Transactions on Software Engineering* 21:12 p. 929-944.

[63] Jena - A Semantic Web Framework for Java, 2008. http://jena.sourceforge.net/index.html

[64] Mobile Speed Test. 2011. http://www.mobilespeedtest.com/

[65] Wordnet 1.7.1 2011. http://www.ontologyportal.org/WordNet.owl