MS Project Proposal

P-DeepMiner – Scaling up DeepMiner using MapReduce to handle large number of pages

Varun Goyal

Committee Chair: Dr. Xumin Liu

Reader: Dr. Rajendra K. Raj

Department of Computer Science
B. Thomas Golisano College of Computing and Information Sciences
Rochester Institute of Technology
Rochester, New York

July 25, 2013

Abstract

The vision of semantic web services promises a network of interoperable web services over different sources. A major challenge to the realization of this vision is the lack of automated means of acquiring domain ontologies necessary for annotating web pages. Wensheng Wu et al. propose DeepMiner, which learns domain ontologies from the source web sites to discover domain concepts and instances and thus develop the domain ontology. We propose P-DeepMiner to effectively learn domain ontology from interfaces and data pages of a set of domain sources. Given a set of sources in a domain of interest, P-DeepMiner first learns a base ontology from their query interfaces. It then grows the current ontology by probing the sources and discovering additional concepts and instances from the data pages retrieved from the sources. The reiterating step of growing the ontology is performed using map reduce thus making the system more efficient.

1 Introduction

Internet and its usage have grown exponentially in past four and a half decades. How did this come to be? First, it was the ease of communication via email and chat engines that brought more users on board. As the number of users grew, so did the data they brought along with them via blogs, social networking, ratings, reviews etc. Similarly, as companies increased their online presence they brought a huge dump of data with them, like product database, company information, location information, service information, travel schedules, news etc. However, the representation of data was never standardized. The main focus was to improve the data representation for human consumption and not machine consumption.

Search engines like Google, Yahoo and Bing are the most used tool to search and consume the content available over the web. These search engines rely on Keyword based search algorithms to look for the webpages most relevant to the search query. The user has to then manually analyze the most relevant and usable data from the resulting millions of pages.

For example, say the search query fired was: “semantic web is awesome”. Machine’s interpretation of the query will be: search for all the websites, which contain the keywords “semantic”, “web” and “awesome”. Thus resulting in websites containing these keywords “semantic” and “web” and everything that is “awesome”. The pages with the maximum number of hits are displayed as most relevant pages.

Semantic web was introduced to interrelate the data and create a symphony of content over the Internet [1]. One of the important applications of semantic web is to optimize the search engines and thus return most relevant search result based on the meaning of search query and the content on those page, thus, removing the need for the user to manually analyze the result. From the example above, the aim of semantic web is that the machine can process the search query as: search for all the websites, which praise semantic web.

2 Background

2.1 Semantic Web

Keyword-based search engines, such as Google, Bing, Yahoo and Ask.com, are the main tools for using today's web. These search engines are the reason that web is such a huge success today. However, as discussed in the introduction there are serious problems associated with the use of search engines. Listing out some of the main ones:

- *High recall, low precision*: Even if the main relevant pages are retrieved, they are of little use if another 110,000,000 mildly relevant or irrelevant documents were also retrieved in 0.20 seconds. That is a huge amount of data for the user to analyze.

- *Low or no recall*: Sometimes we don't get any result for our request, or the relevant and important pages are not retrieved in the search result. This often happens due to the results being highly sensitive to the search query.

- *Results highly sensitive to search query*: Often we have to use semantically similar keywords to get the results we wish; in these cases the relevant documents use different terminology from the original query. This behavior is unsatisfactory, since semantically similar queries should return similar results.

Even when the search is successful, user has to manually browse through the resulting web pages to extract the information he/she is actually looking for which can be very time-consuming. In other words, information retrieval is not supported that well, only data retrieval is. Therefore a more appropriate term for search engines would be data location finder instead of information retrieval tool. Also, the search engines are often isolated application, which can be plugged into tools and let the users find the location of data but not easily accessible to utilize the search result.

At present the main obstacle for providing a better support for information retrieval is that the meaning of the content available over the web is not machine processable, and in most of the cases not available. There are tools which can retrieve texts, check the spelling, split the text, translate them, find meanings and synonyms of the words, publish the data over various channels, but when it comes to extracting useful information and interpreting the data the current software tools have very limited capabilities. Say for example, these software tools cannot differentiate between the two sentences below:

"This is a project on semantics."
"This is one of the projects on semantics."

There is a lot of research going on in artificial intelligence, but such a research is expensive and time consuming, and has show limited results so far. An alternative approach is to, add a layer on top of the existing data and represent the content over the web in a form that is more

easily machine processable and then to use smart techniques to take advantage of these representations. This is what semantic web aims to achieve. As mentioned before semantic web would be a layer on top of the existing data available over the web instead of creating a whole new global information highway parallel to the existing World Wide web. Instead it will gradually evolve out of the existing web [1].

2.2 Ontology

Ontology defines a set of representational primitives with which to model a domain of knowledge or discourse. The representational primitives are typically

- classes (or sets),
- attributes (or properties),
- and relationships (or relations among class members).

The definitions of the representational primitives include information about their meaning and constraints on their logically consistent application. In the context of database systems, ontology can be viewed as a level of abstraction of data models, analogous to hierarchical and relational models, but intended for modeling knowledge about individuals, their attributes, and their relationships to other individuals. Ontologies are typically specified in languages that allow abstraction away from data structures and implementation strategies; in practice, the languages of ontologies are closer in expressive power to first-order logic than languages used to model databases. For this reason, ontologies are said to be at the "semantic" level, whereas database schema are models of data at the "logical" or "physical" level. Due to their independence from lower level data models, ontologies are used for integrating heterogeneous databases, enabling interoperability among disparate systems, and specifying interfaces to independent, knowledge-based services. [10]

OWL, RDF, RDFS, OIL, DAML, and DAML + OIL are some of the languages for marking ontological statements. XML is the foundation of these languages. We'll describe more about few prominent ontology languages and the influence of XML.

2.3 XML

Extensible Markup Language, is a meta-language that allows users to define their own customized markup languages, esp. in order to display documents on the World Wide web. XML provides a surface syntax for structured documents, but imposes no semantic constraints on the meaning of these documents. XML tags are specific to the project they are written for, thus similar tags in different projects can have different meaning.

2.4 RDF

RDF is a basic data model, like the entity-relationship model, for writing simple statements about web objects (resources). It represents the data in the form of subject-predicate-object expression. Such expressions are known as triples in RDF terminology. Example: for the

statement “Color of the plate is white.” Subject is “the plate” predicate is “color” and object is “white”. The RDF data model does not rely on XML, but RDF has an XML-based syntax.

2.5 RDF Schema

RDF Schema provides modeling primitives for organizing web objects into hierarchies. Key primitives are classes and properties, subclass and sub-property relationships, and domain and range restrictions. RDF Schema is based on RDF. RDF Schema can be viewed as a primitive language for writing ontologies. But there is a need for more powerful ontology languages that expand RDF Schema and allow the representations of more complex relationships between web objects. Example: horse is a sub-class of animal. This relation can be expressed in RDFS syntax as shown in the following syntax:

```
<rdf:Description rdf:ID="horse">
      <rdfs:subClassOf rdf:resource="#animal"/>
</rdf:Description>
```

2.6 OWL

OWL Web Ontology Language is designed for use by applications that need to process the content of information instead of just presenting information to humans. OWL has more facilities for expressing meaning and semantics than XML, RDF, and RDF-S, and thus OWL goes beyond these languages in its ability to represent machine interpretable content on the web. OWL is a revision of the DAML+OIL web ontology language incorporating lessons learned from the design and application of DAML+OIL [5].

2.6.1 Sublanguages of OWL

OWL provides three increasingly expressive sublanguages designed for use by specific communities of implementers and users.

- *OWL Lite*: supports those users primarily needing a classification hierarchy and simple constraints. For example, while it supports cardinality constraints, it only permits cardinality values of 0 or 1. It should be simpler to provide tool support for OWL Lite than its more expressive relatives, and OWL Lite provides a quick migration path for thesauri and other taxonomies. Owl Lite also has a lower formal complexity than OWL DL.
- *OWL DL:* supports those users who want the maximum expressiveness while retaining computational completeness (all conclusions are guaranteed to be computable) and decidability (all computations will finish in finite time). OWL DL includes all OWL language constructs, but they can be used only under certain restrictions (for example, while a class may be a subclass of many classes, a class cannot be an instance of another class). OWL DL is so named due to its correspondence with *description logics*, a field of research that has studied the logics that form the formal foundation of OWL.

- *OWL Full:* is meant for users who want maximum expressiveness and the syntactic freedom of RDF with no computational guarantees. For example, in OWL Full a class can be treated simultaneously as a collection of individuals and as an individual in its own right. OWL Full allows an ontology to augment the meaning of the pre-defined (RDF or OWL) vocabulary. It is unlikely that any reasoning software will be able to support complete reasoning for every feature of OWL Full.

2.7 Single Link Clustering Algorithm

Single-link clustering is one of several methods of agglomerative hierarchical clustering. In the beginning of the process, each element is in a cluster of its own. The clusters are then sequentially combined into larger clusters, until all elements end up being in the same cluster[19]. At each step, the two clusters separated by the shortest distance are combined. The definition of 'shortest distance' is what differentiates between the different agglomerative clustering methods. In single-linkage clustering, a single element pair makes the link between two clusters, namely those two elements (one in each cluster) that are closest to each other. The shortest of these links that remains at any step causes the fusion of the two clusters whose elements are involved. The method is also known as nearest neighbor clustering. The result of the clustering can be visualized as a dendrogram, which shows the sequence of cluster fusion and the distance at which each fusion took place [18].

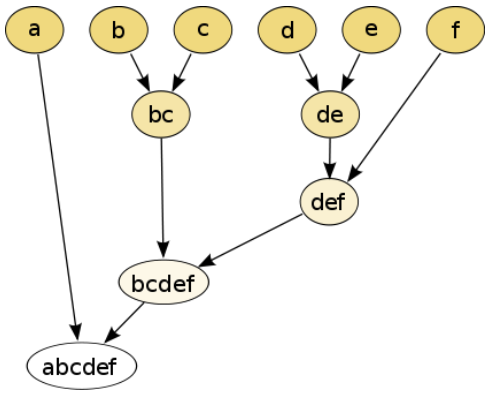

Figure 1: Dendrogram of a cluster.

2.8 Map Reduce

MapReduce is a programming model for processing large data sets with a parallel, distributed algorithm on a cluster [16]. A MapReduce program comprises a Map() procedure that performs filtering and sorting (such as sorting students by first name into queues, one queue for each name) and a Reduce() procedure that performs a summary operation (such as counting the number of students in each queue, yielding name frequencies). The "MapReduce System" (also called "infrastructure", "framework") orchestrates by marshaling the distributed servers, running the various tasks in parallel, managing all communications and data transfers between the various

parts of the system, providing for redundancy and failures, and overall management of the whole process.

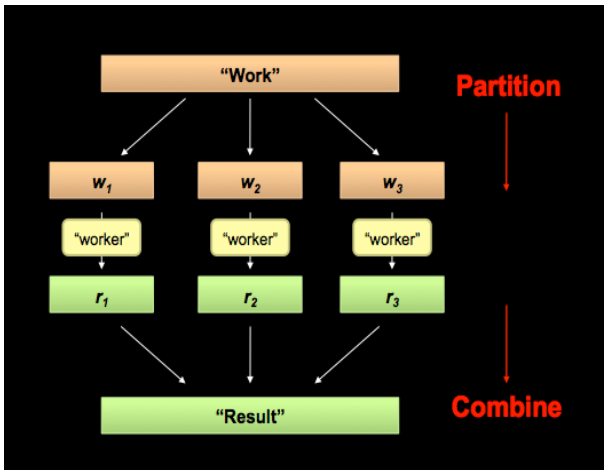

Figure 2: MapReduce Divide and Conquer

This model is inspired by the map and reduce functions commonly used in functional programming, although their purpose in the MapReduce framework is not the same as their original forms. Furthermore, the key contribution of the MapReduce framework are not the actual map and reduce functions, but the scalability and fault-tolerance achieved for a variety of applications by optimizing the execution engine once [17].

3 Related Work

Automating the discovery and interoperation of semantically marking up web services is being actively researched [11, 12, 13, 14, 15, 20, 21, 22, 23]. There have been some efforts in learning domain ontology for web services. This project is an enhancement of [2]. In the paper [2], Wensheng Wu. et al. explains about DeepMiner. This project is to implement DeepMiner evaluate its performance and then scale it up using map reduce and thus evaluate its performance.

P-DeepMiner is also most closely related to [19], but different in several aspects. First, [19] probes through the documentations, which might accompany the descriptions of web services to learn the ontology, while P-DeepMiner exploits the information from the source web sites. Second, [19] learns ontology from natural language texts while P-DeepMiner extracts concepts and instances from semi-structured data over source interfaces and data pages.

[6] Proposes METEOR S, a framework for annotating WSDL files with concepts from existing domain ontology. Exploiting a suite of matchers such as token matcher, synonym finder, and n-gram matcher, METEOR S identifies the mappings between elements in the WSDL files and the concepts in the ontology.

[7] Annotates semantic attributes in source interfaces by employing several machine learning algorithms. The annotation relies on manually constructed domain ontology. However, P-DeepMiner aims to automatically learn domain ontology from the information on the source web sites. The learned ontology can then be utilized to annotate the web services.

4 Hypothesis

The problem this project aims to solve is to bootstrap the source websites from a single domain, and build an ontology by probing through the source interface. Then further improve upon that ontology by firing queries and probing through the resulting data pages. So in simple statement, the problem is of learning domain ontology from a given set of sources in a domain of interest. The learned domain ontology should have the following components:
(1) Concepts: e.g. origin, destination, date of travel, class, and number of passengers are concepts of the airline booking domain.
(2) Instances of concept: e.g. NYC, Boston, London, Rochester etc. are instances of the concept origin (or destination).
(3) Synonyms: e.g. the concept destination may also be denoted by to, arrival, etc.
(4) Statistics: i.e., how frequent each concept and its instances appear in the domain.
(5) Data types: of the concept instances, e.g., instances of price are monetary values while instances of year are four-digit numbers.

There are certain challenges this project will face. We need to understand these challenges before discussing the solution of the proposed problem. Below is the list of challenges this project will face:

- **Limited information**: The knowledge acquired from source interfaces only is often incomplete since data pages of the sources may contain additional information. Further, different sources may contain a different set of concepts and instances.

- **Heterogeneities among sources**: Due to the autonomous nature of sources, the same concept may be represented quite differently over different sources. Another major challenge is thus to identify the semantic correspondences between concepts learned from different sources.

- **Concept retrieval**: Extracting concepts and instances from data pages is significantly more challenging than from query interfaces (since concepts and instances on an interface are typically enclosed in a form construct).

- **Processing Time**: P-DeepMiner will probe through source interfaces and their data pages and each source can have hundreds of data pages. Thus increasing the number of sources increases the processing time exponentially.

5 Solution Design and Implementation

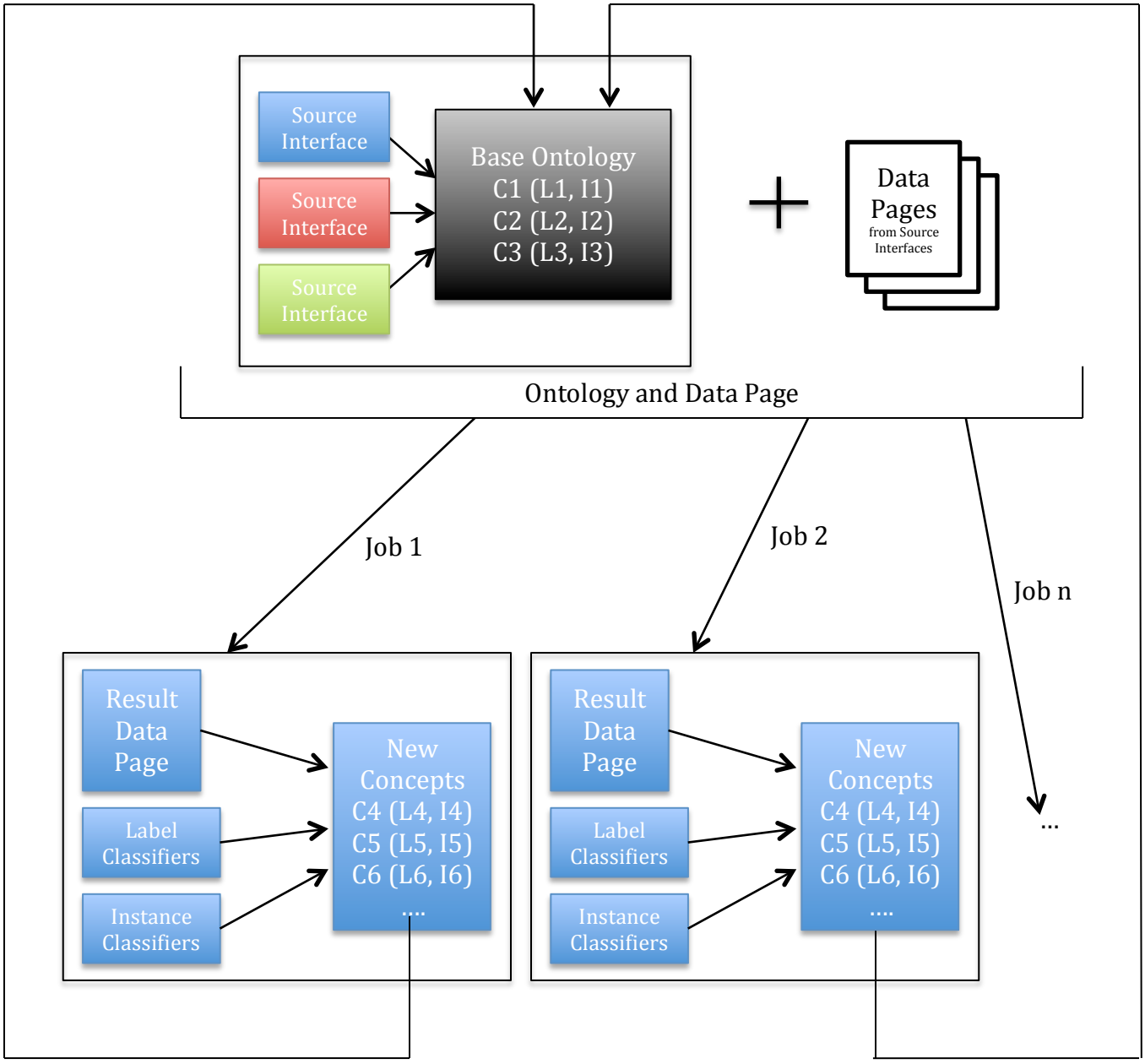

Figure 3. Solution Architecture

Solution for this project can be reached by following the three steps:

5.1

- **Parse source interface(s):** First we will parse the interfaces of the source

websites using Java's jSoup library. This will parse the interfaces converting them to a DOM structure for easy data extraction. Using the DOM structure of the interface all the available labels and their instances will be extracted and stored in complex data-structures like hash map for faster access.

- **Extract concepts, label classifiers, instance classifiers**: Using single-link clustering algorithm to effectively identify mappings of attributes over the interfaces, these labels will be sorted into clusters and classified as Label Classifiers. Specifically, the similarity of two attributes is evaluated based on the similarity of their labels (with the TF/IDF function commonly employed in Information Retrieval).

- **Create base ontology**: Finally for each produced cluster, P-DeepMiner will add a new concept into its base ontology. Each cluster will contain the information obtained from the attributes in the cluster, including labels and instances.

5.2 Map reduce

Now the system will fire queries to the source interface to generate data pages. These data pages will contain additional labels and instances. A job will be created for each data page generated along-with the query and the base ontology using Hadoop Map Reduce. This job will be sent to individual processes (Job Node) that will process the data page it receives.

5.3 Parse data page

Similar to the steps followed in Section 5.1, the data page will be converted into DOM structure to extract all the labels and instances from the page. From these labels with the base ontology, using single-link clustering algorithm, more concepts will be extracted and clustered. These new concepts will be then sent to the reduce function which will merge them with the base ontology. The Job Node will also try to improve the original concepts learned in the base ontology from the additional data in the data pages.

Steps 5.2 and 5.3 will be repeated till all the data pages and queries are exhausted. Thus the final Ontology will be printed out to the console.

6 Evaluation

Following steps will be performed to evaluate the precision of the generated ontology :-

- Check for all the concepts highlighted in the ontology, against those on the source interfaces.
- Check if the queries fired to generate data pages are correct
- Check for all the concepts highlighted in the ontology against those on the data pages.

To evaluate the time performance of the system, the system will be run on 1,2,3 and n processors and time required to generate the ontology in each scenario will be recorded. Thus comparing the P-DeepMiner with the DeepMiner proposed by Wensheng Wu. Et al.

7 Roadmap

July 25, 2013 Submit formal project proposal
July 30, 2013 download and setup all the tools required with testing the system integration
August 15, 2013 complete the step described in section 5.1
August 25, 2013 complete the step described in section 5.2 and section 5.3 without MapReduce
September 10, 2013 convert the system to Map Reduce with multiple processors
September 20, 2013 test the system and evaluate its performance
September 30, 2013 Project defense

8 References

[1] Grigoris Antoniou and Frank van Harmelen. The semantic web primer. In The MIT Press Cambridge, Massachusetts.

[2] W. Wu, A. Doan, C. Yu, and W. Meng. Bootstrapping domain ontology for Semantic Web services from source web sites. In Proceedings of the VLDB-05 Workshop on Technologies for E-Services, 2005.

[3] Lee, Tim Berners, James Hendler, and Ora Lassila. "The Semantic Web." (2001).

[4] Princeton University “About WordNet.” WordNet. Princeton University. 2010. <http://wordnet.princeton.edu>

[5] McGuinness, Deborah L., and Frank Van Harmelen. "OWL web ontology language overview." *W3C recommendation* 10.2004-03 (2004): 10.

[6] A. Patil, S. Oundhakar, A. Sheth, and K. Verma. METEOR-S: Web service annotation framework. In WWW, 2004.

[7] A. He. and N. Kushmerick. Machine learning for annotating semantic web services. In AAAI Spring Symposium on Semantic Web Services, 2004.

[8] V. Crescenzi, G. Mecca, and P. Merialdo. RoadRunner: Towards automatic data extraction from large Web sites. In Proc. of VLDB, 2001.

[9] W. Wu, C. Yu, A. Doan, and W. Meng. An interactive clustering-based approach to integrating source query interfaces on the Deep Web. In SIGMOD, 2004.

[10] Gruber, T. (2008). "Ontology". In Liu, Ling; Özsu, M. Tamer. *Encyclopedia of Database Systems* (Springer-Verlag). ISBN 978-0-387-49616-0.

[11] B. Benatallah, M. Hacid, A. Leger, C. Rey, and F. Toumani. On automating web services discovery. VLDB Journal, 14(1), 2005.

[12] F. Casati and M. Shan. Models and languages for describing and discovering eservices. In Tutorial, SIGMOD, 2001.

[13] The OWL-S Services Coalition. OWL-S: Semantic Markup for Web Services. http://www.w3.org/Submission/OWL-S/.

[14] M. Dumas, J. O'Sullivan, M. Hervizadeh, D. Edmond, and A. Hofstede. Towards a semantic framework for service description. In DS-9, 2001.

[15] B. Li, W. Tsai, and L. Zhang. Building e-commerce systems using semantic application framework. Int. J. Web Eng. Technol., 1(3), 2004.

[16] Jeffrey Dean and Sanjay Ghemawat. 2008. MapReduce: simplified data processing on large clusters. *Commun. ACM* 51, 1 (January 2008), 107-113. DOI=10.1145/1327452.1327492 (http://doi.acm.org/10.1145/1327452.1327492)

[17] Wikipedia. Map Reduce. <http://en.wikipedia.org/wiki/MapReduce>

[18] R. Sibson (1973). "SLINK: an optimally efficient algorithm for the single-link cluster method". *The Computer Journal*(British Computer Society) **16** (1): 30–34.

[19] M. Sabou, C. Wroe, C. Goble, and G. Mishne. Learning domain ontologies for web service descriptions: an experiment in bioinformatics. In WWW, 2005.

[20] M. Paolucci and K. Sycara. Semantic web services: Current status and future directions. In ICWS, 2004.

[21] K. Sivashanmugam, K. Verma, A. Sheth, and J. Miller. Adding semantics to web services standards. In ICWS, 2003.

[22] D. VanderMeer, A. Datta, et al. FUSION: A system allowing dynamic Web service composition and automatic execution. In CEC, 2003.

[23] L. Vasiliu, M. Zaremba, et al. Web-service semantic enabled implementation of machine vs. machine business negotiation. In ICWS, 2004.