Building a Better Stockbroker: Managing Big (Financial) Data by Constructing an Ontology-Based Framework

Logan Westrick
westricl@mail.gvsu.edu
Epic
Verona, Wisconsin 53593 USA

Jie Du
dujie@gvsu.edu

Greg Wolffe
wolffe@gvsu.edu

School of Computing & Information Systems
Grand Valley State University
Allendale, MI 49401 USA

Abstract

Financial investment decision making is a complex process, in which decision makers utilize specific techniques to analyze a large volume of noisy time-series data in order to arrive at a final decision. Collecting and managing the enormous amount of available financial data is an important task in this process, for both researchers and end-user investors. This paper proposes an ontology-based framework for effectively managing big financial data. It further describes the steps required to implement such a framework, and reports the results of a feasibility study into implementing the proposed framework. A Financial Statement Ontology (FSO) is created using the Web Ontology Language (OWL) in the Protégé knowledge framework together with a data acquisition driver written in Perl. The use of an ontology adds a layer of abstraction to Big Data, alleviating the need for end-users to concern themselves with added complexity. The framework thus allows researchers and investors to spend more time on problem-solving and less time managing Big Data. In addition to the described application to finance, the proposed framework has the potential to be applied to any other domain in which relevant data is distributed across multiple systems or is accessed using different formats or names, such as is common in medical research.

Keywords: big data, ontology, financial decision support systems, knowledge base

1. INTRODUCTION

The "3Vs" model defines Big Data as "high volume, high velocity, and/or high variety information assets that require new forms of processing to enable enhanced decision making, insight discovery and process optimization" (Beyer, 2014). Researchers have proposed various methods for addressing the difficulties caused by the unique properties of Big Data.

Examples include using visual analysis tools (Jafar, Babb, & Dana, 2014) and using ontologies (Buitelaara, Cimianob, Frankc, Hartungc, & Racioppa, 2008).

The explosive growth of Big Data has led to problems in managing data in such a way that it remains easily accessible to users (McAfee & Brynjolfsson, 2012). One of the disciplines in which this is most evident is the area of finance.

There is a wealth of financial information available on today's Internet. For example, Google, Yahoo, and MSN provide extensive financial data including financial statements, information on domestic and international financial markets, and business news relating to companies. Investors increasingly rely on these data to inform their financial investing decisions, such as predicting stock behavior (Deller, Stubenrath, & Weber, 1999). Given the rapid growth of accessible real-time financial data, the problem of effectively collecting and managing this data has become a challenging task. Given that context, this paper strives to answer the following research question:

How should one effectively manage big financial data so as to better make an informed financial investing decision?

Recently, Du and Zhou (2012) proposed the use of an ontology-based framework to address the problem of normalizing financial data obtained from multiple sources. They defined several data oriented concerns and then presented an ontology mapping mechanism designed to mitigate these problems. Building on their work, this paper describes the process and conducts a feasibility study into the creation of an ontology-based knowledge base for financial data. The goal is to allow investors to easily compare and use financial information obtained from heterogeneous online sources, and thus make intelligent and well-informed analytical decisions.

To enable this, a Financial Statement Ontology (FSO) is created using the latest version of the Web Ontology Language (OWL 2) in the Protégé knowledge framework. All key attributes found in balance sheets, income statements, and cash flows are captured in the FSO. Then, a Perl-based data acquisition driver is used to seamlessly access online sources. It combines the online data with the base ontology to produce an ontology containing individual entries.

As a case study, the practical implications of our findings show the promise of applying the proposed framework and knowledge base, especially to other domains.

The remainder of this paper is organized as follows: Section 2 provides relevant background information on ontologies, the semantic web, and financial decision making. Section 3 presents specifics of the proposed ontology-based framework, while Section 4 documents the implementation details. Section 5 describes the use of the proposed ontology-based framework and provides a sample workflow. Section 6 discusses our findings as to the maturity of the technology, lessons learned, and potential pitfalls. Section 7 concludes the paper.

2. BACKGROUND

An ontology is a formal specification of a set of concepts, agents, and relationships. Ontologies find their basis in the Semantic Web. The term *Semantic Web* was coined by Tim Berners-Lee and is defined as "a web of data that can be processed directly and indirectly by machines" (Berners-Lee, Hendler, & Lassila, 2001). The foremost purpose of a semantic web, or net, is to encapsulate knowledge and its representation so that machines can "understand" and respond to complex human requests. The Semantic Web movement is led by the World Wide Web Consortium (W3C) with the goal of embedding semantic data (data about what things mean; as opposed to syntax, which is simply their structure) into the current unstructured web to create a network of data that can be navigated by machines. This would enable intelligent agents to conduct the tedious work of finding and processing data, freeing humans to do more important (or at least less menial) tasks.

One of the central components of the Semantic Web is the ontology. Ontology was introduced by Gruber as meaning an explicit specification of conceptualization (Gruber, 1993). An ontology as the term is used in computer science is essentially an extension of the time-tested relational database to include semantic data in addition to syntactic data.

The standard language for ontologies in the Semantic Web is the Web Ontology Language (OWL); version 2 is the current standard (OWl, 2014). The OWL specification includes several variants that differ in their expressiveness. Some of these sublanguages, or profiles, allow for faster computer reasoning by restricting the set of allowed statements. OWL 2 DL (Direct Logic) is the most permissive subset that remains computable (use of the abbreviation OWL will refer to OWL 2 DL unless otherwise stated). The central idea of OWL is that any relational database can be simplified to three fields; subject, predicate, and object. A single entry is thus known as a relation. Other names for an entry include axioms (as they are the data assumed to be true by a computer reasoner) and triples. By specifying a set of default verbs and providing the ability to define more verbs within the ontology, OWL allows for embedding a knowledge base's semantics within itself. This gives a computer program the ability to read the ontology and reason with it without prior knowledge of its structure.

This constitutes a substantial improvement over current development processes, in which both the structure and semantics of data must be explicitly programmed into a program before meaningful work can be accomplished. As an additional benefit, an ontology that follows the proper OWL restrictions can be reasoned on by a computer reasoner, which can make deductive inferences based on the already-stated relations in the knowledge base. In theory, this can allow for faster and more precise development, since programmers no longer need to specify every single relation.

Similar to a database, an OWL ontology can be accessed by using a query language known as the SPARQL Protocol and RDF Query Language (SPARQL) (W3C, 2014b), a variant of Structured Query Language (SQL) that is customized for use with triple stores and ontologies.

Ontology plays a key role in the field of Information Systems, such as improving information consistency and reusability, systems interoperability and knowledge sharing (Kishore, Ramesh, & Sharman, 2007). The crucial research issues surrounding ontology focus on two aspects: ontology generation and ontology mapping (Ding & Foo, 2002a, 2002b). An ontology is generated to provide a shared framework of the common understanding of a specific domain. The creation process can be manual, semi-automated, or fully automated. Ontology mapping expands and combines existing ontologies to support communication and interaction between existing and new domains.

Ontologies have previously been applied to the field of finance (Chenga, Lub, & Sheu, 2009; Fensel & Brodie, 2003). For example, Wand & Wang (1996) focused on improving data quality, and Du and Zhou (2012) endeavored to mitigate data quality problems. When the data quality has been compromised, their framework is used to fix the data quality problem by retrieving the correct data from another data source. We extend their work and propose our own ontology-based framework which provides a knowledge base for end users, transparently allowing them to access multiple data sources as deemed necessary.

3. FRAMEWORK

This paper proposes an ontology-based framework for effectively managing Big Data. The basic structure of the framework is illustrated in Figure 1 (see Appendix). There are three key components to the framework:

- a base ontology,
- online data sources,
- a data acquisition driver.

The first component is the base ontology, called the FSO, in which the key financial concepts from balance sheets, income statements, and cash flows are defined. Their relationships are also captured in the FSO.

The second component is the abundant financial data available on the Internet. At the time this project was implemented, Google Finance, Yahoo!Finance, and MSN Money Central each provided free financial data for investors. Typically each of these data sources represents the data using their own unique knowledge representation structure.

The final component in the framework is the data acquisition driver. The role of the driver is to access online sources and to combine their data with the base ontology to create an ontology containing individual entries.

It is important to note the separation between information supplied by the base ontology and information added by the driver. In this step, different names for the same type of object are stored as labels to help with the ontology mapping of the framework.

For illustration purposes, and to graphically convey the complexity of the interconnections, the expanded diagram of an FSO populated with a single set of statements from a single company (Google) is given in Figure 2. The ontology used for framework testing and debugging has roughly 10x this amount of data; typical production ontologies could easily have 1000x as much data. This graphically illustrates the potential benefit of using an ontology to help manage Big Data. Abstraction and automation can relieve end-users of the burden of dealing with this level of detail and complexity.

4. IMPLEMENTATION

In order to implement an ontology-based framework, a number of important decisions need to be made:

1. Decide whether to use a development environment and choose an ontology syntax.
2. Decide which OWL reasoner to incorporate into the framework.
3. Determine how to handle inconsistent or incomplete data.
4. Decide how to store and access the ontology from within the driver.
5. Determine driver language and database access protocol.
6. Decide how best to access the driver-generated ontology from the user's perspective.

Development Environment / Syntax

Two choices for the development environment are the NeOn toolkit and the Protégé ontology editor. The NeOn Toolkit is the ontology engineering environment developed as part of the NeOn Project (NeOn, 2006). The toolkit is based on the well-known Eclipse platform and provides an extensive set of plug-ins. While the NeOn toolkit has a sleek and intuitive UI, it lacks support for some of the most recent features of OWL, including the ability to specify keys. The Protégé ontology editor is a free, open source ontology editor. It is referred to as "the leading ontological engineering tool" (Gašević, Djurić, & Devedžić, 2009). It completely supports all OWL features and allows for saving an ontology in all of the various OWL syntaxes, and includes several convenient visualization tools. For most applications, the ease of adding classes and the built-in visualization tools also make Protégé a far better solution than writing OWL by hand. Therefore, the decision was made to use Protégé for ontology development.

OWL supports several different syntaxes for creating ontologies. For this project, Functional syntax was chosen because of its conciseness and relative ease of specification (OWL's own formal specification uses the same syntax). Below is a sample giving the flavor of the Functional syntax:

```
ClassAssertion (:Company :GOOG)
DataPropertyAssertion (:hasName
    :GOOG "Google"^^xsd:string)
ObjectPropertyAssertion (:hasEntry
    :GOOG
    :GOOG_NonRecurringOpEx_2013-12-
    31_2014-04-02_03-46-51Z)
```

Other common syntaxes include the RDF/XML syntax (W3C, 2014a), which is very verbose but is the most widely supported OWL syntax, and the Manchester syntax (W3C, 2008), which is specifically designed to be easily readable by non-logicians. Less commonly used are the Turtle (W3C, 2012) and OWL/XML syntaxes (W3C, 2013).

Reasoner Selection

One of the benefits of using an ontology is to take advantage of computer reasoning. For example, a reasoner can exploit the knowledge embedded in a transitive relationship without direct user coding or intervention.

At this time, the choice of reasoner is straightforward: the HermiT reasoner is the only one that fully supports the newest specification of OWL (OWL 2), and it is also the fastest of the free reasoners (KRR, 2014). HermiT is written in Java; it can work with other languages (e.g. Perl), but naturally works best with Java. It can be imported directly in Java, it can be accessed via the OWL API, or it can be run from the command line. The Protégé development environment supports the use of different reasoners via a plugin system, so incorporating the HermiT reasoner was straightforward.

Inconsistency Handling

The most difficult part of this project was determining how to handle the occurrence of inconsistent data. An OWL reasoner can easily determine whether or not an ontology is consistent. It can make inferences, provided the ontology is consistent. But at this time, reasoning with an inconsistent ontology seems to be impossible without writing a custom reasoner (Rosati, Ruzzi, Graziosi, & Masotti, 2012) or requiring direct human intervention.

A further question arises when dealing with financial applications: whether or not it is even desirable to repair inconsistent data? After all, inconsistent data might be a sign of fraud.

Given the nature of this study, it made sense to simply leave inconsistent data as is. As long as the ontology contains data about a subject from at least one source, a properly formed query will return that result transparently. If data is inconsistent, the same query will return all of the multiple results. It is left to the user to determine the significance of the inconsistency and how best to handle it.

Another common problem found in the online financial data sources is terminological ambiguity (Du & Zhou, 2012). Specifically, different financial sources use different names for the same data. One possible solution to this general problem is to create and use a thesaurus to handle the terminological ambiguity. For example, Mannette-Wright (2009) developed a shared knowledge environment for automating the document transformation problem in the medical field, in which the HL7 industry standard thesaurus is used to resolve terminological ambiguity. In the finance field, a recent study combines Semantic Web technologies and linked data principles to increase interoperability and comparability of business reports represented with XBRL markup (O'Riain, Curry, & Harth, 2012). XBRL US GAAP Taxonomies v1.0 defines concepts and their relationships in financial statements and can be used as a thesaurus to resolve discrepancies among reports that are prepared by accountants who are using different accounting principles. In our study, the terminology ambiguity problem is solved by adding label annotations to the various entry classes in the ontology based on the XBRL US GAAP Taxonomies. For example, by creating an appropriate label in the FSO, the reasoner can infer that the concept of "Cost of Revenue, Total" found in Google Finance is equivalent to the concept of "Cost of Revenue" as given in Yahoo!Finance.

A diagram of the final implementation of the ontology can be seen in Figure 3. The "base" ontology, specified by the user, is given in blue. The remainder of the ontology, filled in by the accompanying driver and inferred by the included reasoner, is in red. The ontology contains several types of entities:

- Class: a set or category, represented by a taller rectangle (e.g. "`GrossProfit`")
- Individual: a member of a class, represented by a shorter rectangle (e.g. "`:GOOG`")
- Relation: a connection between concepts or objects, represented by a rounded box (e.g. "`hasVal`")
- Literal: a fixed value, represented by an ellipse (e.g. "`"2013-12-31"^^xsd:date`")

Ontology Storage/Access
Since it captures concepts and relationships, an ontology is meant to be persistent, implying the need to store the ontology for future access. In a full production environment, as might be found in a brokerage, the expectation would be substantial redundancy and security. It would employ triple-store servers and authentication tools. As part of a research project, this prototype simply leaves the base ontology as a file in OWL functional syntax. The file can either be explicitly supplied to the driver or the driver can be configured to access the default base ontology online.

Driver Design
Most OWL-related software is written in Java, including the OWL application programming interface (API). However, for this project, Perl was chosen as the language for driver implementation. This was due partly to local expertise with the language but primarily because of Perl's advanced capabilities for text-processing. APIs send all data as text, requiring it to be parsed; Perl was well suited to that task.

User Interface
Many commercial systems include an API for a SPARQL engine. The API contains hooks which allow a developer to easily build a GUI on top of it, abstracting the actual SPARQL code away from the user. In other words, the user constructs a SPARQL query by selecting options from drop-down lists; essentially interacting with the system by filling out forms.
Due to time and budget constraints, the prototype described here uses the free SPARQL engine built into Protégé. It is basically a command-line interface. SPARQL is the SQL-based language used to construct queries to the Resource Description Framework (RDF), the standard model for Web-based data interchange. Figure 4 illustrates the command-line interface included with Protégé. A sample SPARQL query shows the SQL-like nature of the language. The figure also shows the outcome of executing the query – multiple results are matched and returned.

5. USAGE

A diagram of a typical workflow can be seen in Figure 5. Rectangles are user-directed actions, ellipses are automated framework activities. The diagram is also color-coded to match the ontology structure diagram in Figure 3, showing interactions with the base ontology and the fully-populated ontology.

The first step is initiated by the user; simply run the driver. The driver expects:

- a list of names and ticker symbols of companies whose data should be retrieved,
- the location of the base ontology ("`def`" tells the driver to access and use the default base ontology stored online),
- the name of the file to write/store the completed ontology to,
- whether to access quarterly or annual reports, and
- at least one group number.

The syntax is therefore "driver.pl tickerList {localFile | def} outfile {annual|quarterly} $group_1$ ($group_2$... $group_n$)". This command will automatically generate an ontology populated with all of the available online data for the companies specified in the ticker list. The data will be normalized as per the specifications in the ontology, and the final ontology will be written to backing store.

In the next major step, the user opens the populated ontology in the Protégé knowledge editor.

Protégé includes a console-based SPARQL engine. The user can use this command-line interface to query the ontology, similar to querying a database. The reasoner in the framework helps make inferences regarding the relationships in the ontology; e.g. it can reconcile different names for the same entity.

Budget and time permitting, future enhancements might involve developing GUI-based interaction with the SPARQL engine.

6. DISCUSSION / LESSONS LEARNED

The proposed ontology-based framework goes well beyond traditional databases. It:

- Encodes *semantics* in addition to *syntax.*
- Facilitates knowledge pooling and inter-departmental communication.
- "Knows" that different terms mean the same thing, allowing different users to query the ontology using the terminology most familiar to them.
- Infers unstated relations that logically follow from stated ones, shortening development time and reducing error.
- Allows both researchers and business people to spend more time on the core problems in their fields and less time managing Big Data.

Experience with this project demonstrates that the ontology model is definitely ready for practical use. OWL supports either adding labels to classes in an ontology or adding multiple names to an individual as data properties. This means that a user can query a properly constructed ontology using the terminology that they are familiar with and receive data from all disciplines, even if other disciplines added their data to the ontology under a different name. The most practical immediate application for such a feature would be in constructing research databases.

For example, a genetics research database might have numerous papers and publications about a particular gene. The gene symbol used by the Mouse Genome database is *Pax6*, while the homolog in the Human Genome database would be called *PAX6*. The entry in the ontology could be given both names, and a scientist studying the gene in humans can then add a paper to the ontology, tagging it with the keyword "*PAX6*". If a mouse researcher later queried the ontology for all papers written about "*Pax6*", the ontology would also return papers concerned with that gene and written by human gene researchers.

This study did expose several limitations with the current implementations of the ontology model. Perhaps because of its origins in philosophy, logic, and linguistics, the area where the ontology framework still has the most maturing to do is its handling of numerical data. For this reason, a financial knowledge base is not really the best subject for a case study on ontologies. While this project was successful in creating a working ontology framework, there were several places where it became readily apparent that OWL was being used in ways it was not quite designed for. Most notably, there is no way to specify that the value of one entry should be the "sum of", or "difference of", or is otherwise numerically related to other entries.

For instance, gross profit is defined as the difference between total revenue and cost of revenue in GAAP. But the current ontology cannot specify this kind of relationship between numerical data. To get around this limitation, the framework was configured with a property such that a user could still see these relationships, but a machine reasoner would have to be explicitly told what the properties meant before it could use them. This is perhaps the most important task at which a standard relational database is still better than an ontology.

Another potential pitfall to those considering a similar project in the area of finance is the lack of free financial APIs. In the recent past there were several free APIs including Yahoo! Finance, MSN Money, and Google Finance. At the time of this writing, only Yahoo! Finance remains free. MSN Money and Google Finance still exist as websites accessible to human users. However, they no longer have APIs for developer use, and their current terms of service prohibit using an automated program to access their data. Therefore, to assess the ontology's performance at normalizing data from multiple sources, it was necessary to manually enter and use mock data for eventual population into the complete ontology. Although this method worked and demonstrated a proof-of-concept, it is obviously not the same as using multiple online sources.

7. CONCLUSION

This research project proposes an ontology-based framework to facilitate the management of Big (financial) Data. While still nascent, the ontology model shows great promise for managing enormous amounts of data in an efficient, transparent, and user-friendly way.

The main contribution of this research is to implement the proposed ontology framework and to assess its performance in a financial application. The implementation is described in detail, ranging from a discussion of available tools to consideration of important issues. Ironically, OWL's focus on object classification rather than numerical data means that at present a financial knowledge base is not really the best choice of subject for an ontology; our success in spite of that bodes well for the future of OWL and the Semantic Web in general.

8. REFERENCES

Berners-Lee, T., Hendler, J., & Lassila, O. (2001). The Semantic Web: Scientific American.

Beyer, M. (2014). Gartner Says Solving 'Big Data' Challenge Involves More Than Just Managing Volumes of Data Retrieved June 6, 2014

Buitelaara, P., Cimianob, P., Frankc, A., Hartungc, M., & Racioppa, S. (2008). Ontology-based information extraction and integration from heterogeneous data sources. International Journal of Human-Computer Studies, 66(11), 759–788.

Chenga, H., Lub, Y.-C., & Sheu, C. (2009). An ontology-based business intelligence application in a financial knowledge management system. Expert Systems with Applications, 36(2), 3614–3622.

Deller, D., Stubenrath, M., & Weber, C. (1999). A survey on the use of the Internet for investor relations in the USA, the UK and Germany. *European Accounting Review,* 8(2), 351-364.

Ding, Y., & Foo, S. (2002a). Ontology research and development. Part 1 - a review of ontology generation. Journal of Information Science, 28(2), 123-136.

Ding, Y., & Foo, S. (2002b). Ontology research and development. Part 2 - a review of ontology mapping and evolving. Journal of Information Science, 28(5), 375-388.

Du, J., & Zhou, L. (2012). Improving the quality of financial data using ontologies, Decision Support Systems. Decision Support Systems, 54(1), 76-86.

Fensel, D., & Brodie, M. (2003). Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce (2 ed.). New York, NY: Springer.

Gašević, D., Djurić, D., & Devedžić, V. (2009). Model Driven Engineering and Ontology Development (2nd Edition): Springer.

Gruber, T. (1993). A transitional approach to portable ontology specifications. Knowledge Acquisition, 5(2), 39-41.

Jafar, M., Babb, J. S., & Dana, K. (2014). Decision-Making via Visual Analysis using the Natural Language Toolkit and R. Journal of Information Systems Applied Research, 7(1), 33-46.

Kishore, R., Ramesh, R., & Sharman, R. (Eds.). (2007). Ontologies: A Handbook of Principles, Concepts and Applications in Information Systems: Springer–Verlag.

KRR. (2014). HermiT OWL Reasoner Retrieved June 6, 2014, from http://www.hermit-reasoner.com/

Mannette-Wright, A. (2009). A Feasibility Study of Ontology-based Automatic Document Transformation: Pace University.

McAfee, A., & Brynjolfsson, E. (2012). Big Data: The Management Revolution. Harvard Business Review.

NeOn. (2006). NeOn Toolkit User's Guide Retrieved June 6, 2014, from http://www1.cs.unicam.it/insegnamenti/reti_2008/Readings/neon_users_guide.pdf

O'Riain, S., Curry, E., & Harth, A. (2012). XBRL and open data for global financial ecosystems: A linked data approach," International Journal of Accounting Information Systems. International Journal of Accounting Information Systems, 13(2), 141-162.

OWl. (2014). OWL 2 Web Ontology Language Document Overview (Second Edition) Retrieved April 4, 2014, from http://www.w3.org/TR/owl2-overview

Rosati, R., Ruzzi, M., Graziosi, M., & Masotti, G. (2012). Evaluation of Techniques for Inconsistency Handling in OWL 2 QL Ontologies. In P. Cudré-Mauroux, J. Heflin, E. Sirin, T. Tudorache, J. Euzenat, M. Hauswirth, J. Parreira, J. Hendler, G. Schreiber, A. Bernstein & E. Blomqvist (Eds.), The Semantic Web – ISWC 2012 (Vol. 7650, pp. 337-349): Springer Berlin Heidelberg.

W3C. (2008). OWL 2 Web Ontology Language: Manchester Syntax Retrieved September 7, 2014, from http://www.w3.org/TR/2008/WD-owl2-manchester-syntax-20081202/

W3C. (2012). Turtle - Terse RDF Triple Language Retrieved September 7, 2014, from http://www.w3.org/TR/2012/WD-turtle-20120710/

W3C. (2013). OWL Web Ontology Language: XML Presentation Syntax Retrieved September 7, 2014, from http://www.w3.org/TR/owl-xmlsyntax/

W3C. (2014a). RDF/XML Syntax Specification Retrieved September 7, 2014, from http://www.w3.org/TR/rdf-syntax-grammar/

W3C. (2014b). SPARQL Query Language for RDF Retrieved June 6, 2014, from http://www.w3.org/TR/rdf-sparql-query/

Appendix (Figures)

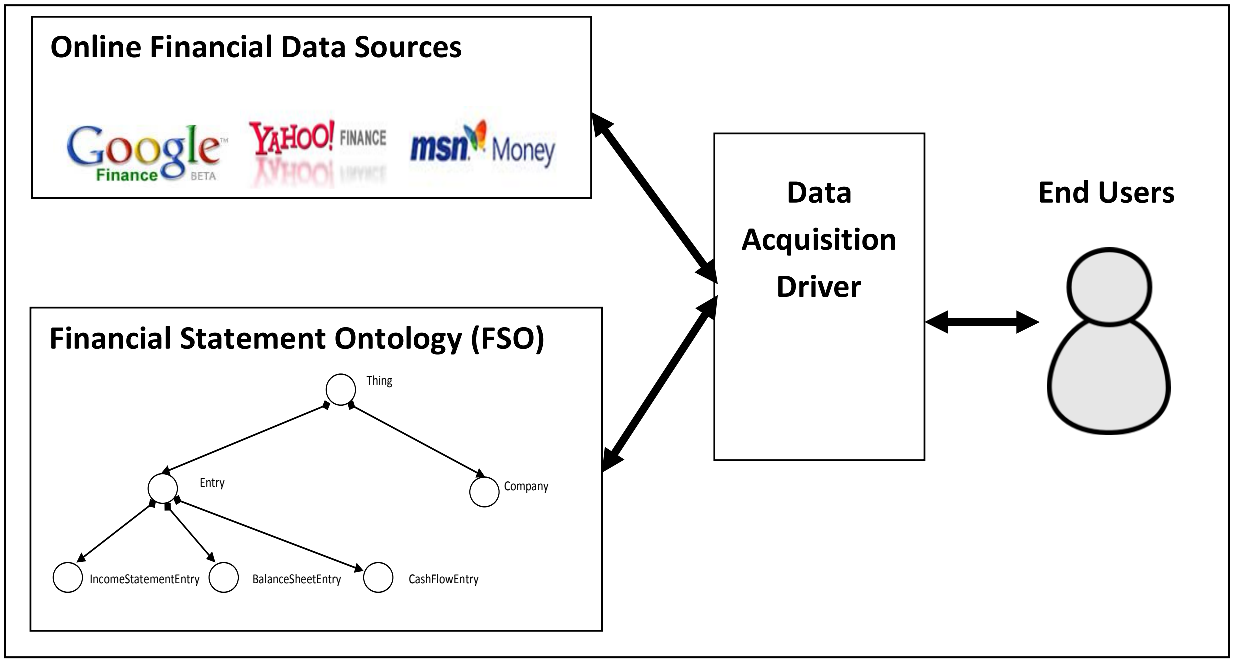


Figure 1: The proposed ontology-based framework for managing Big (financial) Data

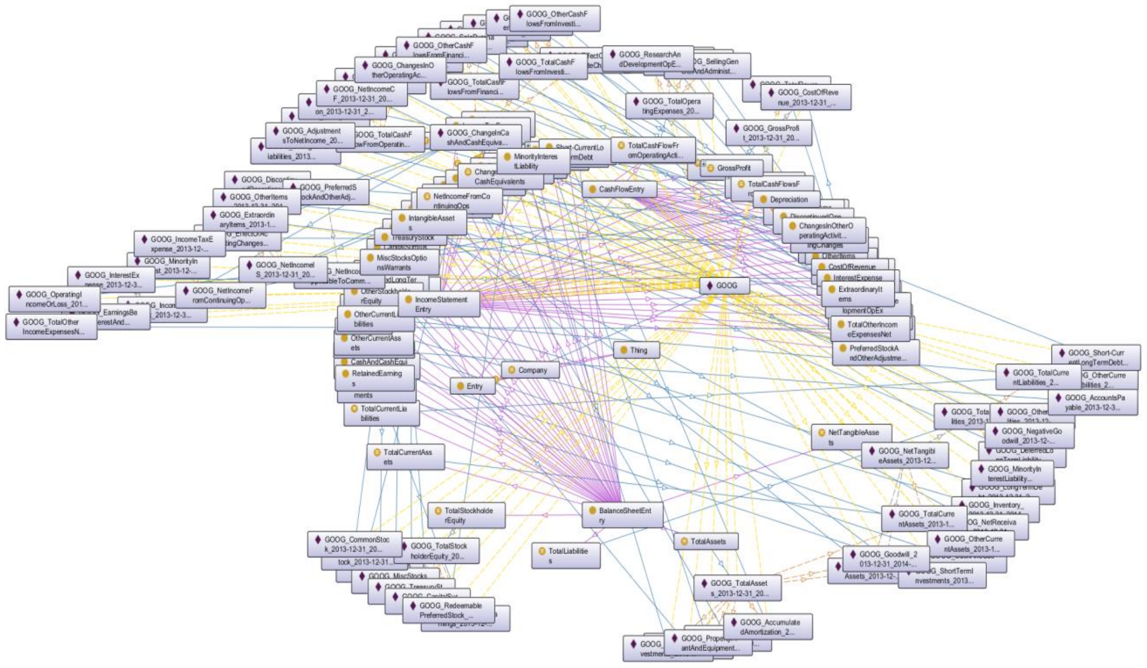


Figure 2: Fully-expanded diagram of the financial ontology populated with a single set of statements from a single company. Typical systems would be many orders of magnitude more complex.

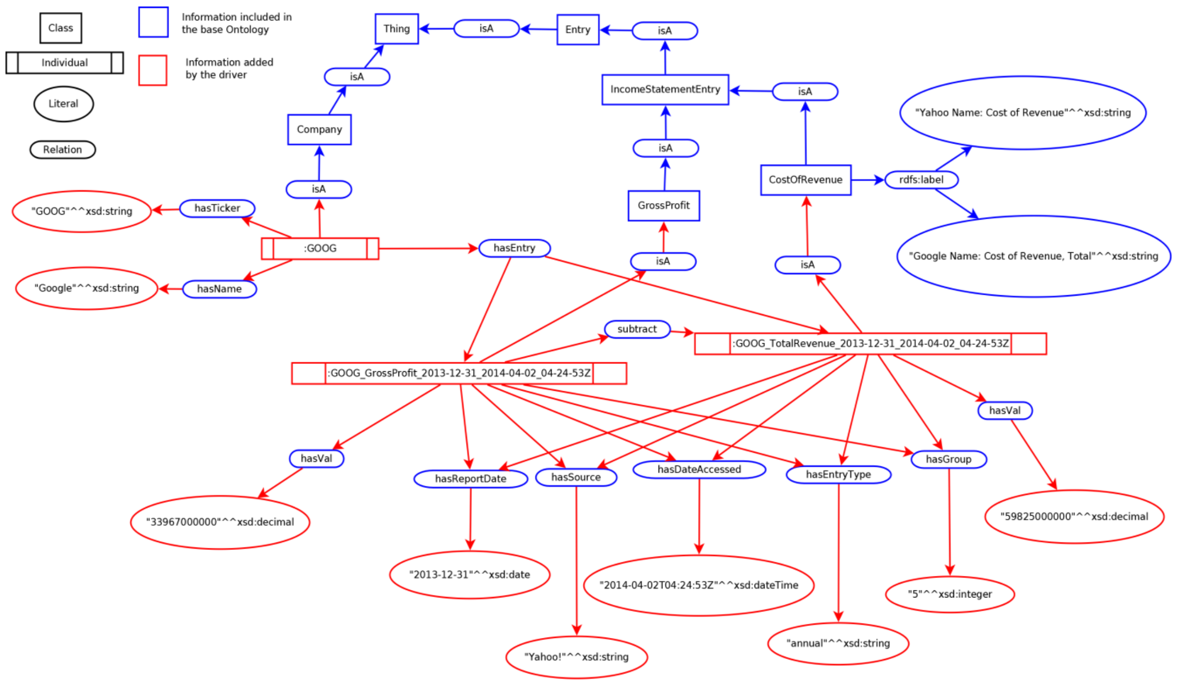
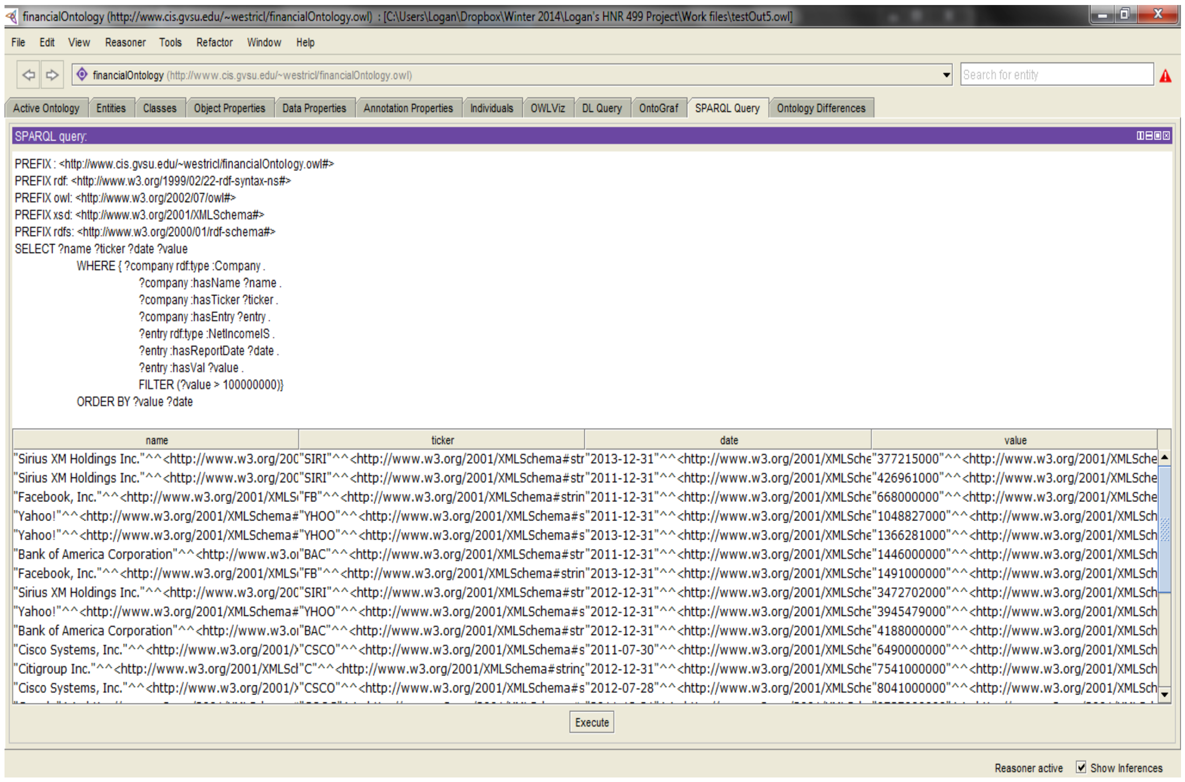

Class
Individual
Literal
Relation
Information included in the base Ontology
Information added by the driver
Thing
isA
Entry
isA
isA
Company
IncomeStatementEntry
isA
"Yahoo Name: Cost of Revenue"^^xsd:string
isA
isA
CostOfRevenue
rdfs:label
GrossProfit
"GOOG"^^xsd:string
hasTicker
:GOOG
hasEntry
isA
isA
"Google Name: Cost of Revenue, Total"^^xsd:string
"Google"^^xsd:string
hasName
subtract
:GOOG_TotalRevenue_2013-12-31_2014-04-02_04-24-53Z
:GOOG_GrossProfit_2013-12-31_2014-04-02_04-24-53Z
hasVal
hasVal
hasReportDate
hasSource
hasDateAccessed
hasEntryType
hasGroup
"33967000000"^^xsd:decimal
"59825000000"^^xsd:decimal
"2013-12-31"^^xsd:date
"2014-04-02T04:24:53Z"^^xsd:dateTime
"5"^^xsd:integer
"Yahoo!"^^xsd:string
"annual"^^xsd:string

Figure 3: Diagram of the final design of the ontology. Blue represents the "base" ontology. Red entities are obtained via the driver and/or inferred by the reasoner.


financialOntology (http://www.cis.gvsu.edu/~westricl/financialOntology.owl) : [C:\Users\Logan\Dropbox\Winter 2014\Logan's HNR 499 Project\Work files\testOut5.owl]
File Edit View Reasoner Tools Refactor Window Help
financialOntology (http://www.cis.gvsu.edu/~westricl/financialOntology.owl)
Search for entity
Active Ontology Entities Classes Object Properties Data Properties Annotation Properties Individuals OWLViz DL Query OntoGraf SPARQL Query Ontology Differences
SPARQL query:
Execute
Reasoner active
Show Inferences

```
PREFIX : <http://www.cis.gvsu.edu/~westricl/financialOntology.owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?name ?ticker ?date ?value
        WHERE { ?company rdf:type :Company .
                ?company :hasName ?name .
                ?company :hasTicker ?ticker .
                ?company :hasEntry ?entry .
                ?entry rdf:type :NetIncomeIS .
                ?entry :hasReportDate ?date .
                ?entry :hasVal ?value .
                FILTER (?value > 100000000)}
        ORDER BY ?value ?date
```

name	ticker	date	value
"Sirius XM Holdings Inc."^^<http://www.w3.org/200	"SIRI"^^<http://www.w3.org/2001/XMLSchema#str	"2013-12-31"^^<http://www.w3.org/2001/XMLSche	"377215000"^^<http://www.w3.org/2001/XMLSche
"Sirius XM Holdings Inc."^^<http://www.w3.org/200	"SIRI"^^<http://www.w3.org/2001/XMLSchema#str	"2011-12-31"^^<http://www.w3.org/2001/XMLSche	"426961000"^^<http://www.w3.org/2001/XMLSche
"Facebook, Inc."^^<http://www.w3.org/2001/XMLS	"FB"^^<http://www.w3.org/2001/XMLSchema#strin	"2011-12-31"^^<http://www.w3.org/2001/XMLSche	"668000000"^^<http://www.w3.org/2001/XMLSche
"Yahoo!"^^<http://www.w3.org/2001/XMLSchema#	"YHOO"^^<http://www.w3.org/2001/XMLSchema#s	"2011-12-31"^^<http://www.w3.org/2001/XMLSche	"1048827000"^^<http://www.w3.org/2001/XMLSch
"Yahoo!"^^<http://www.w3.org/2001/XMLSchema#	"YHOO"^^<http://www.w3.org/2001/XMLSchema#s	"2013-12-31"^^<http://www.w3.org/2001/XMLSche	"1366281000"^^<http://www.w3.org/2001/XMLSch
"Bank of America Corporation"^^<http://www.w3.o	"BAC"^^<http://www.w3.org/2001/XMLSchema#str	"2011-12-31"^^<http://www.w3.org/2001/XMLSche	"1446000000"^^<http://www.w3.org/2001/XMLSch
"Facebook, Inc."^^<http://www.w3.org/2001/XMLS	"FB"^^<http://www.w3.org/2001/XMLSchema#strin	"2013-12-31"^^<http://www.w3.org/2001/XMLSche	"1491000000"^^<http://www.w3.org/2001/XMLSch
"Sirius XM Holdings Inc."^^<http://www.w3.org/200	"SIRI"^^<http://www.w3.org/2001/XMLSchema#str	"2012-12-31"^^<http://www.w3.org/2001/XMLSche	"3472702000"^^<http://www.w3.org/2001/XMLSch
"Yahoo!"^^<http://www.w3.org/2001/XMLSchema#	"YHOO"^^<http://www.w3.org/2001/XMLSchema#s	"2012-12-31"^^<http://www.w3.org/2001/XMLSche	"3945479000"^^<http://www.w3.org/2001/XMLSch
"Bank of America Corporation"^^<http://www.w3.o	"BAC"^^<http://www.w3.org/2001/XMLSchema#str	"2012-12-31"^^<http://www.w3.org/2001/XMLSche	"4188000000"^^<http://www.w3.org/2001/XMLSch
"Cisco Systems, Inc."^^<http://www.w3.org/2001/	"CSCO"^^<http://www.w3.org/2001/XMLSchema#s	"2011-07-30"^^<http://www.w3.org/2001/XMLSche	"6490000000"^^<http://www.w3.org/2001/XMLSch
"Citigroup Inc."^^<http://www.w3.org/2001/XMLSc	"C"^^<http://www.w3.org/2001/XMLSchema#string	"2012-12-31"^^<http://www.w3.org/2001/XMLSche	"7541000000"^^<http://www.w3.org/2001/XMLSch
"Cisco Systems, Inc."^^<http://www.w3.org/2001/	"CSCO"^^<http://www.w3.org/2001/XMLSchema#s	"2012-07-28"^^<http://www.w3.org/2001/XMLSche	"8041000000"^^<http://www.w3.org/2001/XMLSch

Figure 4: Sample SPARQL query. Pictured is the SPARQL console included in the *Protege* ontology editor; it uses an SQL-based language to query RDF graphs.

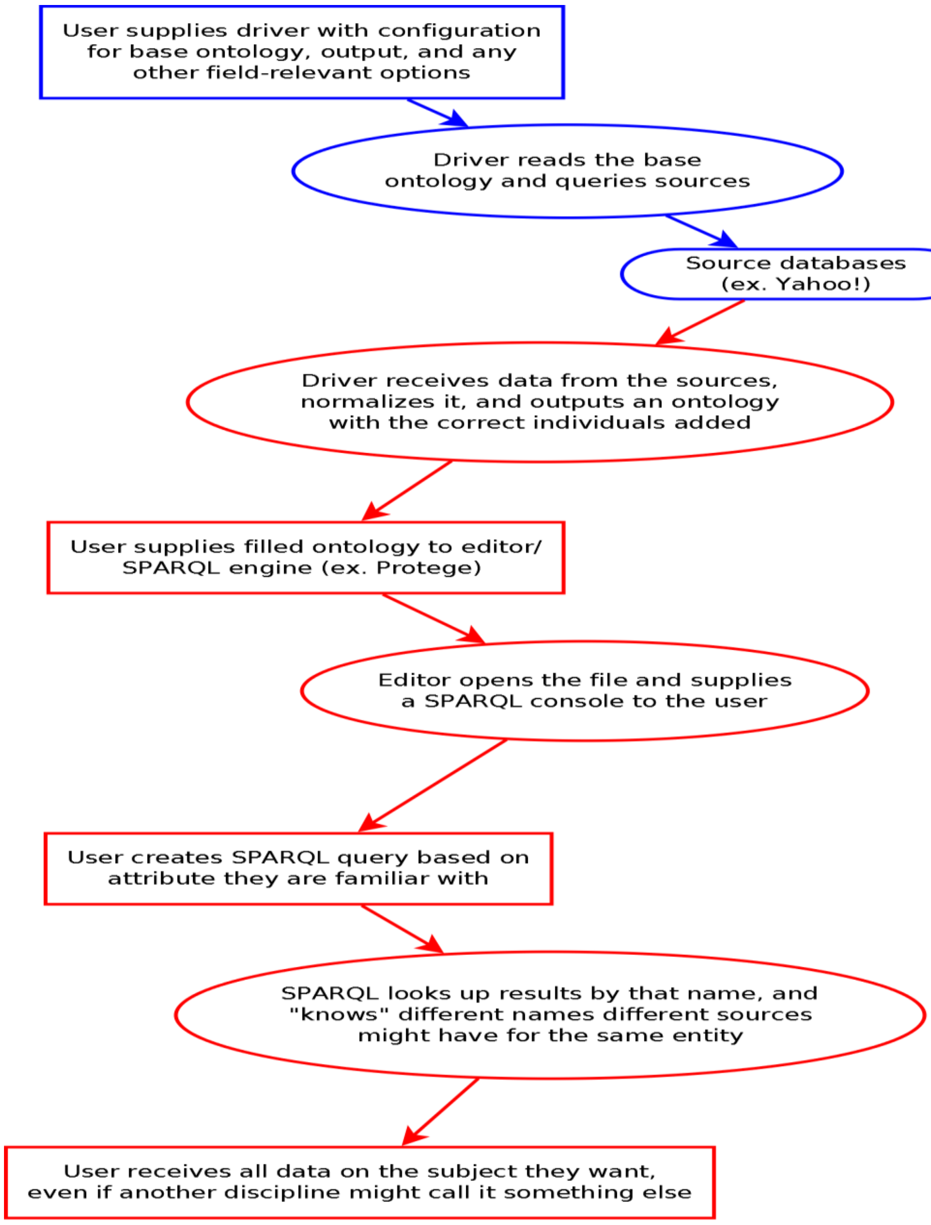

User supplies driver with configuration for base ontology, output, and any other field-relevant options
Driver reads the base ontology and queries sources
Source databases (ex. Yahoo!)
Driver receives data from the sources, normalizes it, and outputs an ontology with the correct individuals added
User supplies filled ontology to editor/ SPARQL engine (ex. Protege)
Editor opens the file and supplies a SPARQL console to the user
User creates SPARQL query based on attribute they are familiar with
SPARQL looks up results by that name, and "knows" different names different sources might have for the same entity
User receives all data on the subject they want, even if another discipline might call it something else

Figure 5: Workflow diagram. Rectangles represent user tasks, ellipses represent automated tasks performed by the ontology framework. Blue represents processes interacting with the base ontology; red represents interactions with the completely populated ontology.