Financial News Analysis Using a Semantic Web Approach

Alex Micu[1], Laurens Mast[2], Viorel Milea[3], Flavius Frasincar[4], and Uzay Kaymak[5]

All authors are affiliated with:

Erasmus University Rotterdam

Burgemeester Oudlaan 50

3062PA, Rotterdam, Netherlands.

Tel.: +31-10-4088926

Fax: +31-10-408 9162

[1] alex_micu@hotmail.com, [2] 289820hm@student.eur.nl, [3] milea@few.eur.nl, [4] frasincar@few.eur.nl, [5] kaymak@few.eur.nl

Financial News Analysis Using a Semantic Web Approach

ABSTRACT

In this paper we present StockWatcher, an OWL-based web application that enables the extraction of relevant news items from RSS feeds concerning the NASDAQ-100 listed companies. The application's goal is to present a customized, aggregated view of the news categorized by different topics. We distinguish between four relevant news categories: i) news regarding the company itself, ii) news regarding direct competitors of the company, iii) news regarding important people of the company, and iv) news regarding the industry in which the company is active. At the same time, the system presented in this chapter is able to rate these news items based on their relevance. We identify three possible effects that a news message can have on the company, and thus on the stock price of that company: i) positive, ii) negative, and iii) neutral. Currently, StockWatcher provides support for the NASDAQ-100 companies. The selection of the relevant news items is based on a customizable user portfolio that may consist of one or more of these companies.

Keywords*: Intelligent Support Systems, Web-Based Applications, Business intelligence, Financial IS, Knowledge Acquisition.*

INTRODUCTION

Unlike printed media or television programs, on the Web, news items can be made public as soon as they emerge. Simultaneously, Web coverage is continuously increasing. News websites provide RSS-feeds facilitating the public to remain up-to-date on nearly any topic of interest.

To better understand what the impact of the Internet is on our daily lives, we should first take a look at the main ideas behind its creation. The suggestion of social communications through networks dates from 1962, when J.C.R. Licklider, a professor at the Massachusetts Institute of Technology (MIT), suggested the “Galactic Network” theory. In this theory, he imagined a “globally interconnected set of computers through which everyone could quickly access data and programs from any site” (Licklider & Clark, 1962). The next step towards the Internet as we know it today was taken by the Defense Advanced Research Projects Agency (DARPA), which created The Advanced Research Projects Agency Network (ARPANET). This project was the first operational computer network in the world, and is seen as the ancestor of the Internet. As time progressed, different networks were created outside the ARPANET, and became eventually interconnected into one super network in 1990, creating the roots of today's modern

Internet. With the presence of this technological infrastructure, the next step towards public accessibility was the foundation of the World Wide Web (WWW). This project, led by Tim Berners-Lee, included the now so popular Hypertext Markup Language (HTML) being used for the creation of web pages, and the Hypertext Transfer Protocol (HTTP) being used to access Web content.

In our technology-driven society many people have a hard time to even imagine a world without the services and benefits of the Internet. To a certain degree it is safe to assume that the society we live in is turning into an information technology society, characterized by the Internet use (Slabber, 2007). Presently more than 1.1 billion people (Miniwatts Marketing Group, 2007) make use of services provided by the Internet. Such services include e-mail messaging, file sharing, streaming media and voice communications. By making use of popular search engines such as Google, people all around the world have access to vast amounts of online information, provided by different websites. Simultaneously, the same people are even able to create Web content, and place information on the WWW without much effort. Eventually the success of the WWW has made it progressively more challenging to find, access, present, and maintain the information available on the web.

In 1998 a new idea was born, under the name of Semantic Web (SW) (Berners-Lee, Hendler, & Lassila, 2001). This was supposed to be an extension to the current WWW as we know it. The SW would revolutionize the way in which data is described and presented, so that it can be read, interpreted and used by various software applications. There are three main goals that the SW seeks to achieve: i) provide common formats for integration and combination of data drawn from diverse sources, ii) record how the data relates to real world objects, and iii) semantically link documents.

Data on the Web is controlled by certain applications, and only useable by these applications. The SW tries to make this data neutral, available to all applications. One of the building stones to achieve this is the Resource Description Framework (RDF) (Brickley & Guha, 2004). RDF is a general-purpose language for representing information in the Web. Together with RDF Schema (RDFS), RDF can be used to code the data so that relations and information about the described entities get coded along (Brickley & Guha, 2004). This can be achieved by basing the representation on triples (Carroll & Stickler, 2004). The next step in the transition to the SW would thus consist of transforming all data into RDF triples.

The Semantic Web proved to be more of a challenge than expected. While it is the next logical step in the evolution of the WWW, it is still relatively unknown to the majority of Internet users. Programmers and website builders prefer to use traditional tools for data presentation, for different reasons. Another difficulty the SW is facing at the moment is converting the current available data on the web. Because of extremely large amounts of data, automatic conversion tools will have to be developed. This is where Natural Language Processing (NLP) should play an important part. With the use of NLP, developers are

trying to teach machines what we understand so easy - human language (Chowdhury, 2003). With the help of NLP, the SW can extract knowledge from different web sources, and convert it into machine understandable formats. The role of the SW is however not different from the role of the WWW: providing access to information.

One of the domains where access to information, and implicitly news, plays a crucial role is represented by financial markets. With the introduction of new products such as click funds, the level of involvement of the general public in investment activities is on the rise. This increased involvement comes to underline the need for access to mediums which can provide relevant and reliable economical news within short time intervals. The Web comes to meet this need, while at the same time confronting users with an overwhelming amount of information. Questions such as '*Where does news appear faster?'* or '*Which news websites are trustworthy?'* have already risen.

Different companies have attempted to fill this 'technology for finance' gap, by providing different internet or desktop applications. Some popular websites already begun providing extensive financial data, allowing users to compose their own portfolio and customize the information flow. Google Finance, Microsoft Money Central and Yahoo Finance are good examples of how these major corporations attempt to implement various data sources with customized user portals for different categories of users. However, the focus of these portals is on providing raw data to users. There is no attempt made in trying to interpret the data, and give viable end solutions to users (for example: "we would advise you to buy this stock based on …"). More detailed applications for expert users are also available, in the form of databases integrated with various news sources like The Wall Street Journal. Thomson Financial DataStream is an expensive database providing reliable financial data for more then 175 countries, going back 50 years. Nevertheless, the user is presented with raw data, without any interpretation at all. In both cases it is safe to assume that companies are still reluctant in creating applications to interpret financial data.

With the emergence of the Semantic Web, languages such as RDF(S) (Brickley & Guha, 2004; Klyne & Carroll, 2004) and OWL (Patel-Schneider, Hayes, & Horrocks, 2004) help provide the basis towards speeding up this process. The goal we pursue here is related to this, and consists of creating an application that helps casual internet users with a relevant involvement in financial markets to find relevant news regarding their portfolio. This effort has resulted in StockWatcher, an application that enables the presentation of a customized, aggregated view of news items categorized by different topics, and at the same time rates these news items based on their relevance.

In the remainder of this chapter, we first provide a background of tools relevant for the current research. Next, we move on to presenting the design choices and architecture of StockWatcher. After this section, we provide a preliminary overview of the system's output. Finally, we end this chapter with some conclusions and suggestions for further research.

BACKGROUND

In this section we give an overview of projects and tools that are similar to the application presented in this chapter. We highlight the main characteristics hereof with a focus on the features offered, while placing everything in the context of the goal pursued here: the development of an application that enables the presentation of a customized, aggregated view of news items categorized by different topics, and at the same time the rating of these news items based on their relevance. Additionally, we provide an overview of current technologies that support (some of) the features envisioned for StockWatcher.

TOWL

The Time-determined Ontology Web Language (TOWL) (Milea, Frasincar, Kaymak, & di Noia, 2007) is a European 6th framework project, with the focus on content studies towards institutional equities services, investors and businesses. The main goal of TOWL is extending the current state-of-the-art ontology language OWL with a temporal dimension. Having time available in an ontology will give the opportunity to migrate from the current static representations of the world into a more dynamic environment. Equipped with this new technology, automated approaches in knowledge extraction from text (with a focus on news) should provide the edge in making improved business decisions.

A semantic stock broker system is being developed in order to show the potential of TOWL as well as to benchmark the features of the project. This system is able to process news items and adjust the ontology employed for this purpose in order to provide better representations of the real world. Based on the extracted information, the system gives a projection of the evolution of the prices of stocks affected by this knowledge, with the final goal of generating excess returns. This last point comes to underline the similarity of this system with the goals being pursued in StockWatcher: better investment decisions based on information extracted automatically from economic news.

The stock broker application developed for TOWL has three important features. First, news items are the only input source for the prediction of the stock price. The second important feature relates to the domain ontology, which offers a great deal of background information regarding the stock markets (for example companies, economical events, persons of interest, and relationships between all these groups). Finally, the impact of the individual news items on the development of stock prices is assessed, enabling the composition of these impacts into an aggregated effect on stock prices.

A prototype of the stock broker application is currently available. This prototype is a pipeline based upon various plugins, from which a considerable percentage are offered by GATE (D. H. Cunningham, Maynard, Bontcheva, & Tablan, 2002):

- ANNIE Tokeniser and Orthographic Analyzer;

- TOWL Sentence Splitter;
- ANNIE POS Tagger;
- WordNet Lemmatiser;
- Cafetiere.

The first two plugins are responsible for the breakdown of the text in words, or sentences. The third plugin, ANNIE POS Tagger is responsible for identifying nouns, verbs, adjectives and adverbs in the text. The WordNet Lemmatiser provides an interface for extracting the lemma of the words in the text (for example if the input is “running”, the output would be “run”). Subsequently, Cafetiere (Black, McNaught, Vasilakopoulos, Zervanou, & Rinaldi, 2005) is also employed for analyzing the text given the domain ontology and finding matches between the ontology and the text. Finally, the OWL Ontology Instantiator updates the ontology with the newly discovered instances.

By making use of different NLP techniques in combination with Semantic Web technologies, the TOWL stock broker shows an overlap with StockWatcher’s goals. The main difference between the two applications relates to the main idea from which they stem from. While the stock broker application is intended for professional investors, StockWatcher is created for the benefit of the ordinary internet user and casual investor.

Artequakt

The Artequakt (Kim et al., 2002) project is one of the popular Semantic Web projects currently in development. One of the main factors that contribute to its popularity is the way in which it tries to bring together some of the most important advantages the Semantic Web. Artequakt's goal is to enable automated internet searches on artists and paintings, from different sources, bring that information together, and present it to different users. Furthermore, the presentation is tailored to suit the user's interest.

The project consists of three significant components. To begin with, a domain ontology describes the world of artists and paintings. Different tools are used for information extraction from web sources. The use of NLP tools is very important at this stage. In the next phase, the focus lies on managing the information. This is done with the help of a knowledge base used to stored information. Finally, an interface allows users to query the knowledge base for information. The end-user can customize the way in which the information is presented.

The first phase in the Artequakt project is of big relevance to StockWatcher. In this phase the knowledge extraction takes place with the help of NLP tools. Right from the start there a number of differences become obvious between the TOWL stock broker and Artequakt. While the stock broker breaks down text to sentences and even words, Artequakt tries to keep the text intact, and treats paragraphs as a whole. Artequakt's

approach considers that a lot of important and relevant information is overseen by breaking down the original text. The paragraphs are then processed through a syntactical analysis where verbs, nouns and other grammatical objects are identified. This is followed by a semantic analysis that consists of four parts: the text is simplified, resulting in simple sentences composed from a subject, verb and an object (also known as a triple). Then, the system attempts to identify named entities (e.g. person's name) by making use of the Apple Pie Parser (Sekine & Grishman, 1995) and GATE (D. H. Cunningham et al., 2002). Finally, the system attaches subjects inherited from the main clauses to the sentences missing one. Artequakt makes use of WordNet (Miller, 1995) to find synonyms, hypernyms, and hyponyms in order to expand the knowledge base.

Although no access to Artequakt is provided for testing purposes, a number of comments can still be made. First of all, there is a clear difference between Artequakt's aims and StockWatcher's goals. Speed is not of essence in the NLP for the knowledge acquisition phase because this process takes place before users access the website. StockWatcher on the other hand needs to carry out the NLP tasks on an on-demand basis, therefore making speed a relevant issue. Furthermore, the Artequakt project involves different Semantic Web tools and approaches, resulting in a wide application of NLP over several phases instead of one recognizable phase. This makes it hard to identify a similar solution for StockWatcher.

SemNews

SemNews (Java, Finin, & Nirenburg, 2006a) is a project focused on applying a complex text understanding process on the news items found on different RSS feeds. It extracts significant information and stores it in a Semantic Web environment, available for browsing and querying. The main power of SemNews is the presentation of the data, which enables complex queries on the news items, going further than simple keyword searches. SemNews also allows users to browse the stored data, so that the logic and mechanics behind the application become clear.

The SemNews workflow consists of three individual phases. Initially the news items are collected and parsed from different RSS feeds. The news items are then transferred to the NLP system where, after being processed, important information and metadata are extracted and stored in the database. The last phase relates to the fact repository interface, where users get the possibility to query and/or browse the data.

The NLP system used by SemNews is called OntoSem (Java, Finin, & Nirenburg, 2006b). With this system, the text gets processed through three stages: a syntactic, semantics and finally a pragmatic analysis. All these result in a text meaning representation (TMR) (Java, Nirneburg et al., 2006). This is where the main difference with the two previous projects comes to light. While TOWL's stock broker and Artequakt make use of WordNet as a lexicon, SemNews utilizes an ontology as resource for fundamental text operations. The TMR does not only act as a lexicon, it goes even further, providing

discourse relations, speaker attitudes, stylistic and other pragmatic factors. In this way SemNews goes beyond the normal text understanding processes, and tries to find relations between different articles. The TMR applied by OntoSem is made possible by a general-purpose ontology called Mikrokosmos, which provides over 30.000 concepts and 400.000 instances.

SemNews is available for free testing on the Web. From the first look at the website, it is worth mentioning that the user interface is friendly and easy to understand. There are different topics available for browsing, and there are a lot of graphical presentations, making the information look interesting. Although SemNews is a great application, the difference between the goals of this project and StockWatcher's are large. The presentation of information in SemNews is not about the meaning of individual news items, but is centered around decomposing the news item in semantically rich information.

OntoSem

OntoSem is a linguistic text processing setting, developed by the University of Maryland (UMBC). OntoSem has as input text and as output a text meaning representation (TMR). The main difference between this platform and other NLP software is that OntoSem makes use of a knowledge base for the operations needed on words.

OntoSem has the ability to perform the following tasks on any available text: preprocessing, morphological analysis, syntactic analysis and semantic analysis. The center of attention of this NLP developing platform is on finding the meaning behind words, sentences, paragraphs, and eventually create relationships between different pieces of text (McShane, Zabludowski, Nirenburg, & Beale, 2004).

GATE

GATE refers to itself as a Software Architecture for Language Engineering (H. Cunningham & Scott, 2004), a platform for developing and testing NLP software. GATE is a popular platform in the scientific world because the same results can be obtained when running the same experiment in different environments. Other benefits of GATE are the benchmarking abilities when researching various processes, and the architecture of the system. Many tools that GATE offers are used as plugins, making it possible to use them outside this platform.

GATE comes with a graphical developing interface, making it easier for users to create their own NLP applications. Another advantage for our current purpose is the Java implementation of GATE. There is also a fully documented application programming interface (API) available in Javadoc accompanied by examples available in the User Guide accessible on the website. However, GATE offers the possibility to use a Java Annotation Patterns Engine (JAPE) (H. Cunningham, 2000). This language is used to express the

required items needed to be annotated in a basic text. Users involved in larger projects can create their own components for working with GATE.

At the first sight, GATE offers all the plugins necessary for StockWatcher, as discussed in the previous section of this section. To start with, it makes use of its own tokeniser, which is tuned for efficiency, so that it will not use many resources. It can also make distinctions between words, numbers, symbols, punctuations and spaces. Additionally, GATE possesses a part of speech tagger plugin, required for the identification of the place of the words in a sentence. The ANNIE POS-tagger produces a part of speech tag for every word that gets processed. Last but not least there is the morphological analyzer, which returns the lemma of each word using also its part of speech tag.

Java NLP Tools

We decided that looking at NLP developing platforms offering all the tools combined in one software package was not enough as our tool's requirements were not met. There are various tools available to fit the requirements of StockWatcher, and we discuss them here.

First, we have the Stanford part of speech tagger, a log-linear POS-tagger developed by the University of Stanford (Klein & Manning, 2003). One of the basic principles behind this tagger is the fact that unknown words need more attention and better support for the tense forms of verbs (Toutanova & Manning, 2000). The Stanford POS-tagger is easy to use in a Java environment, by making use of the Javadoc available on the Stanford website.

For the Morphology tasks of StockWatcher there are different tools available, which provide an API to the features of WordNet: JWordNet (Johar, 2004), Java WordNet Library (JWNL) (Didion, 2007), Java Interface to WordNet (JWI) (Finlayson, 2007), and WordNet Web Application (WNWA) (Bou, 2007). From the first three packages, JWNL provides the best features, giving access to the full functionality of the WordNet libraries. The JWI package can only retrieve a word's lemma, but is unable to provide synonyms, while the JWordNet package makes use of an outdated WordNet library. The last package, WNWA, offers a web interface to the libraries of WordNet, while the other tools need the WordNet dictionary installed locally.

STOCKWATCHER

The focus of this section is on the design of StockWatcher. After presenting the choice for the different building blocks of the application, the focus falls on the architecture of StockWatcher and the different processing phases.

Choosing the Right Components

At a first look, GATE seems to be the best choice for the development of the NLP component of StockWatcher. It offers the right tools for the requirements, supports Java

applications, and is a popular choice in the scientific circles as far as NLP is concerned. However, a main disadvantage of the GATE plugin is the problem of a cumbersome integration with existing applications. OntoSem has a lot to offer as well, however it is not the same concept of plug and play as GATE. Moreover, OntoSem is intended to represent meaning, while StockWatcher needs a lexical reference system for synonyms. For example, “watch” returns only one synonym in OntoSem: “observe”, while WordNet's dictionary has found four direct synonyms, thus recommending the latter as the more complete choice.

The Stanford POS-tagger is easy to use, works fast and reliable, and is fully designed for Java environments. For the morphology tasks, JWNL gives the best results, supporting some of the latest libraries available for WordNet, and giving access to the full functionality that the libraries have to offer. Above that it is designed for Java, making it easier to integrate when compared to the GATE morphological plugin. Finally, the Java String Tokenizer, a standard library available in Java, has the necessary functionality to fulfill the tasks concerning the division of paragraphs into sentences and words.

System Architecture

An overview of StockWatcher’s architecture is depicted in Figure 1, where the workflow model has been separated in three main phases. The first phase is the Data Extraction process. This part of the model is responsible for extracting information concerning the companies supported by our application, and storing it in our local database. The first step in creating StockWatcher was to mine various websites for information. Once the information was found, we employed various techniques, such as HTML Wrappers, to extract this information. The main information sources were *Nasdaq.com* and *Hoovers.com*. Both websites make use of the same *id_keys*, which are used here to uniquely identify a company. Making use of these *id_keys*, we were able to query for specific HTML websites that offer the relevant company information. Furthermore, both sources provide detailed HTML information tags, making it easy to identify the required information. The first website, *Nasdaq.com*, has a certain amount of information on all the companies active on the technology stock market. However, this information is not always complete, so by making use of the *id_keys* found on Nasdaq.com, we queried *Hoovers.com* for additional information. This website provides company information with detailed business reports and industry profiles. For the data storage we employed the Microsoft Access database management system (DBMS). This system provides a simple and flexible solution for our data storage problems, consisting of SQL query support and easy Java integration.

Once the database has been populated, users can compose their own portfolio consisting of the NASDAQ-100 companies present in the database. After this step, an ontology corresponding to the particular user portfolio is generated automatically. All this takes place in the Ontology Creation phase. The ontology was created with the help of Protégé-OWL (Knublauch, Fergerson, Noy, & Musen, 2004), a platform with a powerful graphical user interface, facilitating easy and reliable development of ontologies. In the ontology we distinguish between three major classes: companies, industries, and persons. The first class,

Companies, has two subclasses: companies that represent the user's portfolio and companies that represent competitors of the companies in the user's portfolio. A set of Java modules has been developed by making use of the Jena API. In this way we were able to create, edit and remove instances in our original ontology with the help of Java. With all these components in place, a user can access our website and select a portfolio. Once the selection is finished, StockWatcher produces a personalized ontology for this user. In Figure 2 we illustrate how an individual, in this case Google, is being represented in the ontology.

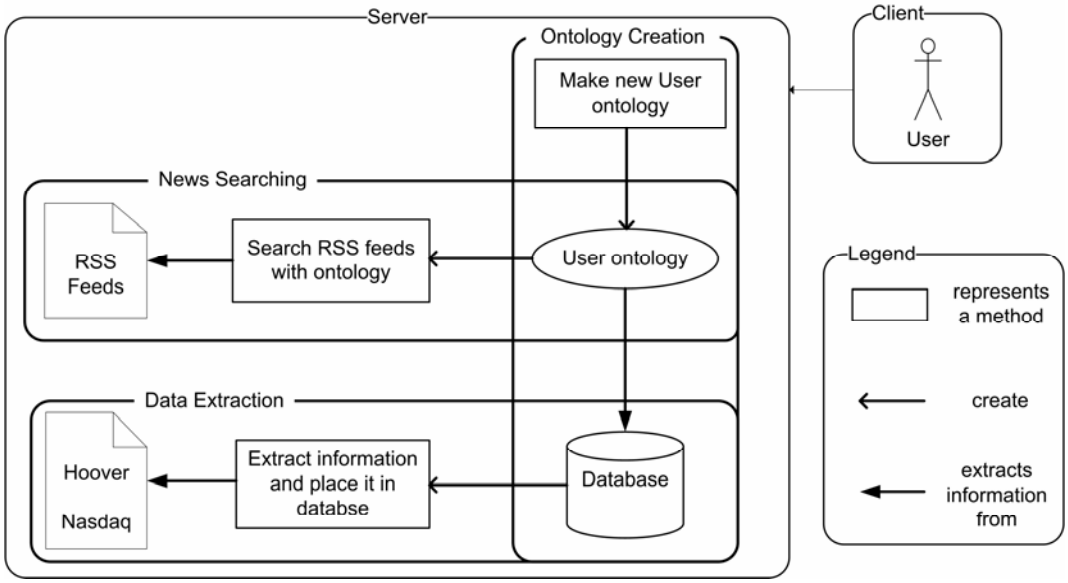

Figure 1: StockWatcher Architecture.

The search for relevant news items may start as soon as the ontology is complete and the RSS feeds are set. To filter out irrelevant matches, and make the application as reliable as possible, a ranking system has been implemented. Based hereon, the news items receive a score based on how many times a word appears in the article. In Figure 3 we present the algorithm employed by StockWatcher for finding the relevant news items for a company. As at this point we are dealing only with proper names (names of people and companies), no NLP is required. The criteria words in the algorithm represent resources like the ones displayed in Figure 2. The system also takes into account where the match is found, making distinctions between the title and the body of a news article. A match in the title receives a score of 2 while a match in the text receives a score of 1. A threshold was established, such that only news items with a score of 2 or higher are presented. This improved the relevance of the results significantly. To search the RSS feeds we employ a Java based platform - Informa (Schmuck, 2007).

Ontology Editing and Querying

For the application to be able to recognize economic events it was necessary to create a knowledge base containing this information. As an ontology is already present in the architecture of StockWatcher, the most efficient way to fulfill this objective is expanding the existing ontology with economic events. The relevant economic concepts are created and categorized in groups. Afterwards, the groups are further sorted by the effect they would normally have on the stock price.

```
<rdf:Description rdf:about="GOOG">
   <data:hasPrimaryIndustry rdf:resource="Internet_Search_&_Navigation_Services"/>
   <data:hasCompetitor rdf:resource="http://competitor/Yahoo!"/>
   <data:hasCompetitor rdf:resource="http://competitor/MSN"/>
   <data:hasCeo rdf:resource="http://person/George_Reyes"/>
   <data:hasCeo rdf:resource="http://person/Sergey_Brin"/>
   <data:hasCeo rdf:resource="http://person/Eric_E._Schmidt"/>
   <data:hasCompanyName>Google Inc.</data:hasCompanyName>
   <data:hasTradeName>GOOG</data:hasTradeName>
   <rdf:type rdf:resource="STOCKWATCHER.owl#CompanyPortofolio"/>
 </rdf:Description>
```

Figure 2: Representation of the Google Individual in the Ontology.

A majority of the economic events are composed from observations gathered from news articles. Furthermore, a glossary offered by the NASDAQ site, containing key events that influence share prices, proved to be useful in further completing the knowledge base (NASDAQ). The following groups of events are identified and added to the ontology: (i) analyst forecasts, (ii) contracts, (iii) earnings, (iv) results, (v) sales, (vi) stocks and shares, (vii) acquisitions, (viii) collaborations, (ix) company performance, (x) new products.

Most of the events enumerated above can have a positive or a negative influence on the stock price. This gives birth to a further division of some of the existing classes in two subclasses: *PositiveEvents* and *NegativeEvents*. (e.g., *ContractsGood* and *ContractsBad* for the *Contracts* class). The groups ‘acquisitions’, ‘collaborations’ and ‘new products’ are assumed here to only have positive influences on the stock price of the company involved, and thus no further differentiations are required.

Text Mining

Making use of the economic events described in the domain ontology, StockWatcher is able to identify news items that can influence stock prices. In order to accomplish this, the NLP tools discussed in the first part of this section have been deployed in our application,

creating an NLP pipeline. This consists of a tokeniser, a part of speech tagger and a morphology plugin. The first phase of this process is the categorization of the news items.

```
FOR all criteria words available
   FOR all the news items available
      split the news item title into words
      FOR all the words in the title
         IF there is a criteria word in the title THEN
            update score with 2 points
         END IF
      END FOR
      split the news item text into words
      FOR all the words in the text
         IF there is a criteria word in the text THEN
            update score with 1 point
         END IF
      END FOR
      IF news item score > 1 THEN
         add to an array of relevant news items
      END IF
   END FOR
END FOR
```

Figure 3: The Search Algorithm.

Once a user has selected his/her portfolio, StockWatcher attempts to extract all relevant news items concerning this portfolio from various RSS feeds. As soon as the news items have been properly identified and categorized, they are processed one by one by the text processor. First, the ontology is queried, extracting all economic events available in the knowledge base. This list of events, together with the list of news items corresponding to the portfolio is redirected to the text processor. The text processor is responsible for all NLP operations required by StockWatcher. Figure 4 illustrates the architectural model of our software.

Once the events have been passed on to the text processor, the application attempts to extract all the synonyms available in the WordNet dictionary. The first step in this process consists of determining the part of speech of every event. This is done by the Stanford POS-tagger that, when queried, returns whether the word is a verb, noun, adjective or adverb. With this information, the index of that word can be resolved from the WordNet dictionary. Once the index is available, all senses of a word can be found, resulting in a full synonym list.

The next step consists of the transformation of news articles. This transformation is required in order to identify economic events. First, the news article has to be split into words. This happens with the standard tokeniser offered by Java, which facilitates several methods for this purpose. Once the words have been separated, they are put through the Stanford POS-tagger. The words, together with their proper part of speech position, are then analyzed by the WordNet Morphological plugin (Bou, 2007). This plugin returns the

lemma (canonical form of a word) for every word. This refers to the headword or heading used for words in any dictionary. Once all the words have been processed, the news article is rebuilt with the new lemmas.

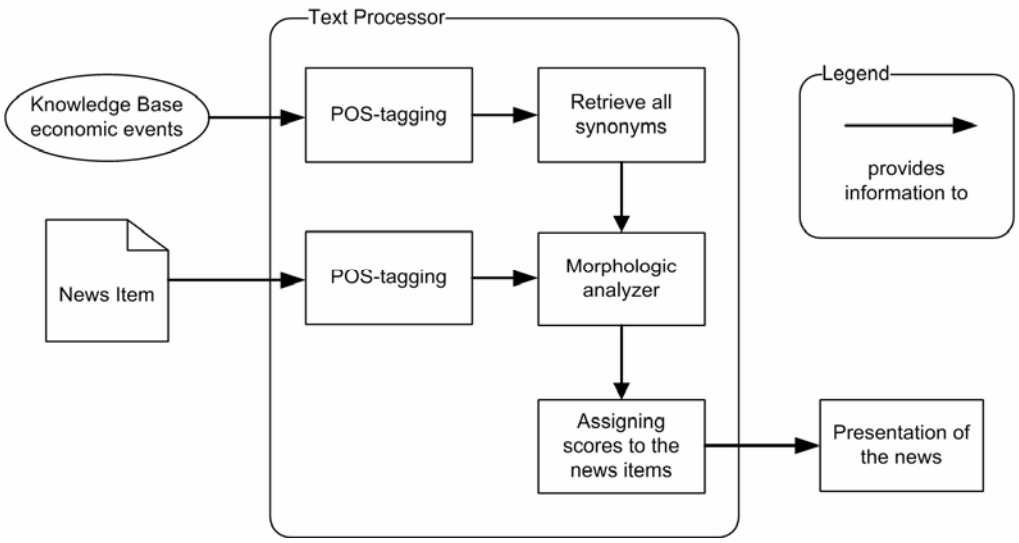

Figure 4: The NLP in StockWatcher.

The final phase of the text processing consists of finding the events in the transformed news articles. For this purpose we introduce a heuristic for the rating system, assigning scores to the news item depending on the position where the event was found. The score can be positive or negative. The positions are divided in two categories, *title* and *body* of the article. If there is a match in the title, this counts for 5 points. A match in the normal text counts for 1 point. Once the entire article has been processed, the total score is determined. This total score is important for two reasons: (i) the total score of an article reflects whether the article will be categorized as positive or negative, and (ii) articles with a total score between -1 and +1 are ignored. From extensive testing we concluded that single text hits resulting in +1 or -1 points are also not significant for our outcomes. In Figure 5 we present the heuristic for the transformation and the rating of items.

Additionally, a dynamic chart was implemented. This chart presents the company's stock price, consisting of a month prior to the user's accesses to StockWatcher. In this way the user can see how the price has progressed and compare it to the outcomes of the news items categorization. The chart is updated dynamically every time StockWatcher is started, and includes the latest stock price available, as shown in Figure 6.

ANALYSIS OF THE RESULTS

In this section we provide a preliminary analysis of the results produced by StockWatcher. We start off by taking a look at the reliability with which news items are categorized as being positive or negative in a concrete case.

```
FOR all the news items
   split the news item title into words
   FOR all the words in the title
      retrieve part of speech of the word
      retrieve lemma of the word
   END FOR
   split the news item text into words
   FOR all he words in the text
      retrieve part of speech of the word
      retrieve lemma of the word
   END FOR
   FOR all the positive events available
      FOR all the lemma words in the title
         IF there is a positive event match THEN
            update score with 5
         END IF
      END FOR
      FOR all the lemma words in the text
         IF there is a positive event match THEN
            update score with 1
         END IF
      END FOR
   END FOR
   FOR all the negative events available
      FOR all the lemma words in the title
         IF there is a negative event match THEN
            update score with -5
         END IF
      END FOR
      FOR all the lemma words in the text
         IF there is a negative event match THEN
            update score with -1
         END IF
      END FOR
   END FOR
END FOR
```

Figure 5: The Transformation and Rating of News Items.

The number of news items being categorized in a positive or negative group depends largely on whether there is an economic event identifiable within the item. After running extensive tests, we present some of the outcomes produced by StockWatcher. To start with we take a look at news about the company itself. In this case Microsoft was chosen, and eight news items were found, from which four items were categorized as being positive and the rest neutral. We discuss one of the articles categorized as positive, and evaluate whether the categorization went correctly.

Microsoft advances cited in report; Date: Tue, Jun 12, 2007

by Paul Krill
InfoWorld
June 12, 2007

Research 2.0, in its May technology research report, found that Microsoft has made "significant advances" with .Net and Windows Vista-driven technology improvements. .Net, Research 2.0 said, has advantages compared to platforms such as the open source LAMP (Linux, Apache, MySQL, PHP) stack. "Perhaps the most striking value of .Net is the

efficiency it affords over other platforms. It's the ability to get to the job at hand vs. building scaffolding to get started," the report stated. Microsoft's Silverlight technology, for running multimedia applications in a browser, will fuel market gains versus Adobe Flash and AJAX, Research 2.0 said. The report also covers the latest SOA findings. Research 2.0 predicts users will keep experimenting with SOA during the next few years but that will peak in 2010 or 2011. SOA will be embraced as mainstream technology by 2015. SOA packaged applications are predicted to dominate application market revenue flows within a decade, disrupting application providers Oracle, SAP and Salesforce.com. The services industry is leading SOA adoption progress; portal- and infrastructure-based approaches to SOA developments are being explored. The Research 2.0 report, which was funded by the Research 2.0 itself, can be downloaded here. The company issues the report to entice investors and businesses to buy the company's other reports, a representative for Research 2.0 said.

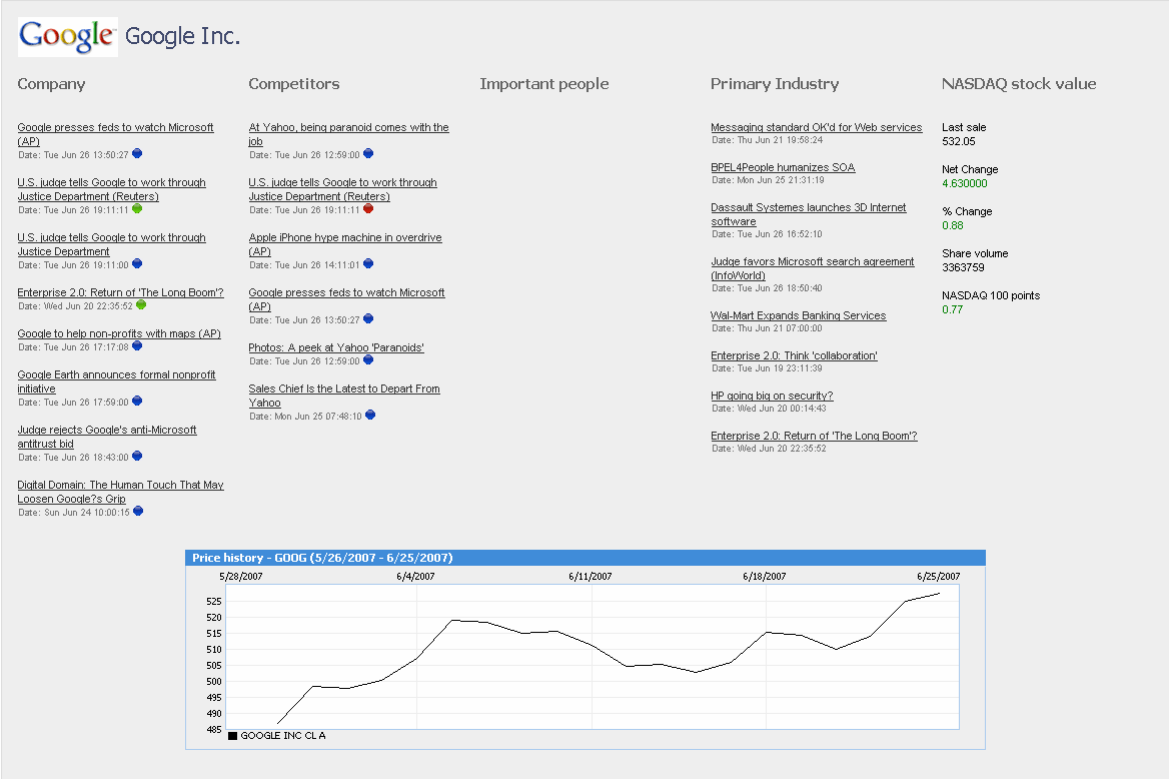

Figure 6: StockWatcher Output.

This news item summarizes a report released by Microsoft in May, concerning the significant advances in their research. The most important topics were the advantages represented by own products in comparison to competitors products (.Net versus Linux,

Apache, MySQL, PHP for example). These advantages would later turn into market share, resulting in bigger profits – thus a positive effect on the stock price. In this article there were only two hits generated in the text: *buy* and *market gain*. The first one was found in the phrase "businesses to buy the" and the second one in "fuel market gains versus". Finally, this article was categorized properly as positive.

The results obtained in this particular case are further supported by the evidence obtained after performing experiments on a larger sample. For this purpose, we have analyzed a dataset of news collected over a longer period of time, stretching across 30 days, from September 1st, 2007 to September 30th, 2007. The list of companies for which the analysis was performed comprised the following entities: Apple, Dell, Google, Microsoft and Yahoo. We have limited ourselves to these companies based on the frequency of news relating to them, and considered this representative enough for the purpose of this preliminary testing. The average number of news per company per day was around 4 for these companies, though exceeding this number on several occasions. However, not all news messages have been categorized as positive or negative, as around half of the total sample has been categorized as neutral (meaning no effect or that the effect cannot be assessed). The other 50% that has been categorized presented interesting results. From the 'postive sample', 30% were incorrectly categorized as positive (false positive), and thus the true-positive percentage amounted 70%. The number are similar in the case of the news with a negative effect: here, around 25% of the news were classified as being negative while they should have been classified as positive (false-negative), while 75% were correctly classified as negative (true-negative).

CONCLUSIONS AND FURTHER RESEARCH

The most important conclusion that can be drawn at the end of this chapter is that StockWatcher is able to classify news reliably. The average percentage of 72.5% correctly classified news items shows that StockWatcher may successfully be employed for the purpose of providing quick, accurate overviews about the market situation at a certain time and as relevant to specific, customizable portfolios. The uniqueness of this application in the financial trading tools landscape helps further positioning StockWatcher as a pioneer in its area. The main problem addressed by this tool relates to providing a way to keep the huge amount of news available today manageable for typical users. The focus in this case in on the financial domain, where the relevance of news items is determined based on a customizable user portfolio.

Additionally, the speed by which the impact of news messages on stock prices can be assessed by a human is negatively affected by the number of news messages available. By automating this process, a very intuitive way is provided of assessing, in one look, the general feeling of the market regarding specific companies. Additionally, less experienced investors, or investors with less experience in a particular domain, may use this tool for approaching markets still less known to them. This is all possible due to the underlying domain ontology where different aspects relevant to this issue have carefully been defined.

Placing StockWatcher in a larger context, we can regard this application as a support tool in the algorithmic trading environment. Algorithmic trading refers to the automated execution of financial trades. Despite its relevance and status as 'buzz word', very little has been published in this domain. Trading algorithms are generally characterized by lack of transparency regarding the underlying strategy, expected costs and corresponding risks, and how the algorithm will adapt to changing market conditions such as prices and liquidity (Kissell, 2006). In this matter, StockWatcher differs from this type of algorithms through a total transparency, being freely available, and most of all because of addressing a unique problem: assessing the impact of news messages on stock prices.

Different approaches were available for StockWatcher's Natural Language Processing component. The different tools available employ different tactics to achieve the same goal: making text understandable to a machine. On the other hand, a pattern emerges: tokenizing the text first, part of speech tagging and finding the lemma for every word. This foundation was also put in practice in our application. For this purpose, several tools were selected, in order to fit our requirements. These tools are the Stanford POS-tagger and the Java WordNet Library. We used both of them in our text processing system, where news items are categorized by positive or negative influences to the stock price. At the same time a knowledge base has been developed, containing multiple economic events.

For the field of natural language processing we distinguished three different segments: speech, grammar and meaning. While speech is of no relevance to StockWatcher, and the grammar part is already implemented, meaning is the only segment that is missing from the big picture. Although the application is still reliable in correctly categorizing the news items, implementing tools to extract the meaning of the identified events would help make StockWatcher more accurate.

Points of improvement relate to the user interface which could be improved, for example by linking the news items and the price history chart (for example the charts offered by Google Finance). In this way, the user would be able to directly relate the different news items to price fluctuations. Moreover, it would function as a benchmark tool for the prediction and classification abilities of our application. If StockWatcher would identify a news item as a positive influence on the stock price, the price fluctuation on the chart should confirm or contradict this.

In addition, extending the knowledge base with new economic events would benefit the application the most. The scoring system could be improved by assigning the ratings based on the economic event, for instance an economic event involving the profits of the company has more influence on the stock price than an event concerning a collaboration. This gives the opportunity to design a more reliable system from an economic point of view.

From the results generated by StockWatcher, we can observe that matches found in the title of an article provide a high degree of certainty that the article is categorized correctly.

From this we can conclude that a title match is more reliable for categorizing the news item, compared to text matches. Furthermore, the position of the match could also be of interest. For example a news article with the title “Microsoft grabs a higher market share than Linux” can get processed by StockWatcher. Based on our current algorithm, this news item would get categorized for two companies: Microsoft and Linux. However, it represents important economical news only for Microsoft. For a correct categorization, StockWatcher needs to calculate the position of the word “Microsoft” in the sentence (first in this case) and compare it to the position of “Linux” (eight), for example.

As a final point, StockWatcher could benefit from the introduction of time constraints in the ontology. The economic events could then also be described based on the moment in time the news message is referring to, how long the effect is likely to last, etc. For example, some economic events can influence the stock price only for a certain period. Based on the data stored in the time property of the economic event and the date of the news article, StockWatcher could check if the match found is still valid or the effect has already passed.

In conclusion, the Semantic Web based application StockWatcher in combination with Natural Language Processing tools has proven to be an efficient way of identifying and categorizing news items by the type of news concerned and their influence on the stock price.

ACKNOWLEDGEMENT

The authors are partially supported by the EU funded IST STREP Project FP6 - 26896: *Time-determined ontology-based information system for realtime stock market analysis (TOWL)*. More information is available on the official website of the TOWL project (http://www.towl.org).

REFERENCES

Berners-Lee, T., Hendler, J., & Lassila, O. (2001). The Semantic Web. *Scientific American*, *284*(5), 28-37.

Black, W., McNaught, J., Vasilakopoulos, A., Zervanou, K., & Rinaldi, F. (2005). *CAFETIERE: Conceptual Annotations for Facts, Events, Terms, Individual Entities and RElations* (Parmenides Technical Report No. TR-U4.3.1), http://www.nactem.ac.uk/files/phatfile/cafetiere-report.pdf.

Bou, B. (2007). WordNet Web Application [Electronic Version]. Retrieved June 2007 from http://wnwa.sourceforge.net/.

Brickley, D., & Guha, R. V. (2004). *RDF Vocabulary Description Language 1.0: RDF Schema* (W3C Recommendation 10 February 2004)

Carroll, J. J., & Stickler, P. (2004). RDF triples in XML. *13th International World Wide Web Conference*, (412-413), ACM Press.

Chowdhury, G. G. (2003). Natural language processing. *Annual Review of Information Science and Technology*, *37*(1), 51-89.

Cunningham, D. H., Maynard, D. D., Bontcheva, D. K., & Tablan, M. V. (2002). GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. *40th Anniversary Meeting of the Association for Computational Linguistics*, (168-175), Association for Computational Linguistics.

Cunningham, H. (2000). *JAPE - A Java Annotation Patterns Engine* (Research Memorandum CS–00–10): Department of Computer Science, University of Sheffield, http://www.dcs.shef.ac.uk/~diana/Papers/jape.ps.

Cunningham, H., & Scott, D. (2004). Software Architecture for Language Engineering. *Natural Language Engineering*, *10*(3-4), 205-209.

Didion, J. (2007). The Java Wordnet Library.

Finlayson, M. A. (2007). MIT Java WordNet Interface. Retrieved August, 2007, from http://www.mit.edu/~markaf/projects/wordnet

Java, A., Finin, T., & Nirenburg, S. (2006a). SemNews: A Semantic News Framework. *Twenty-First National Conference on Artificial Intelligence*, (1939-1940), AAAI Press.

Java, A., Finin, T., & Nirenburg, S. (2006b). Text Understanding Agents and the Semantic Web. *39th Hawaii International Conference on System Sciences*, (62-71), IEEE Computer Society.

Java, A., Nirneburg, S., McShane, M., Finin, T., English, J., & Joshi, A. (2006). Using a Natural Language Understanding System to Generate Semantic Web Content. *International Journal on Semantic Web and Information Systems, 3*(4), 50-74.

Johar, K. (2004). JWordNet Browser. Retrieved August, 2007, from http://www.seas.gwu.edu/~simhaweb/software/jwordnet

Kim, S., Alani, H., Hall, W., Lewis, P. H., Millard, D. E., Shadbolt, N., et al. (2002). Artequakt: Generating Tailored Biographies with Automatically Annotated Fragments from the Web. *Workshop on Semantic Authoring, Annotation & Knowledge Markup*, (1-6), CEUR.

Kissell, R., Malamut, R. (2006). Algorithmic decision-making framework. *Journal of Trading, 1*(1), 12-21.

Klein, D., & Manning, C. D. (2003). Fast exact inference with a factored model for natural language parsing. *Advances in Neural Information Processing Systems*, (3–10), MIT Press.

Klyne, G., & Carroll, J. J. (2004). *Resource Description Framework (RDF): Concepts and Abstract Syntax* (W3C Recommendation 10 February 2004)

Knublauch, H., Fergerson, R. W., Noy, N. F., & Musen, M. A. (2004). The Protégé OWL Plugin: An Open Development Environment for Semantic Web Applications. *3rd International Semantic Web Conference*, (229–243), Springer.

Licklider, J. C. R., & Clark, W. (1962). On-Line Man-Computer Communication. *Spring Joint Computer Conference*, (113-128), National Press.

McShane, M., Zabludowski, M., Nirenburg, S., & Beale, S. (2004). OntoSem and SIMPLE: Two Multi-Lingual World Views. *ACL 2004: Second Workshop on Text Meaning and Interpretation*, (25-32), Association for Computational Linguistics.

Milea, V., Frasincar, F., Kaymak, U., & di Noia, T. (2007). An OWL-Based Approach Towards Representing Time in Web Information Systems. *Workshop on Web Information Systems Modelling (WISM 2007)*, (791-802), Tapir Academic Press.

Miller, G. A. (1995). WordNet: A Lexical Database for English. *Communications of the ACM, 38*(11), 39.

Miniwatts Marketing Group. (2007). Internet World Stats [Electronic Version]. Retrieved June 2007 from http://www.internetworldstats.com/stats.htm.

NASDAQ. NASDAQ Glossary. Retrieved August, 2007, from http://www.nasdaq.com/reference/glossary.stm

Patel-Schneider, P. F., Hayes, P., & Horrocks, I. (2004). *OWL Web Ontology Language Semantics and Abstract Syntax* (W3C Recommendation 10 February 2004)

Schmuck, N. (2007). Informa: RSS Library for JAVA. Retrieved August, 2007, from http://informa.sourceforge.net/index.html

Sekine, S., & Grishman, R. (1995). A corpus-based probabilistic grammar with only two non-terminals. *Fourth International Workshop on Parsing Technologies*, (216-223), ACL/SIGPARSE.

Slabber, N. J. (2007). The Technologies of Peace [Electronic Version]. *Harvard International Review*. Retrieved June 2007 from http://hir.harvard.edu/articles/1336.

Toutanova, K., & Manning, C. D. (2000). Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. *Proceedings of the 2000 Joint SIGDAT conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, (63-70), Association for Computational Linguistics Morristown.