Semi-Automatic Financial Events Discovery Based on Lexico-Semantic Patterns

Jethro Borsje
Medior Software Engineer
SSC-I, National Agency of Correctional Institutions
The Dutch Ministry of Justice
P.O. Box 850, NL-2800 AW Gouda, the Netherlands
E-mail: j.borsje@dji.minjus.nl
Fax: +31 (0)88 07 16576

Frederik Hogenboom*
PhD Student
Econometric Institute
Erasmus University Rotterdam
P.O. Box 1738, NL-3000 DR Rotterdam, the Netherlands
E-mail: fhogenboom@ese.eur.nl
Fax: +31 (0)10 408 9031
*Corresponding author

Flavius Frasincar
Assistant Professor
Econometric Institute
Erasmus University Rotterdam
P.O. Box 1738, NL-3000 DR Rotterdam, the Netherlands
E-mail: frasincar@ese.eur.nl
Fax: +31 (0)10 408 9162

Abstract: Due to the market sensitivity to emerging news, investors on financial markets need to continuously monitor financial events when deciding on buying and selling equities. We propose the use of lexico-semantic patterns for financial event extraction from RSS news feeds. These patterns use financial ontologies, leveraging the commonly used lexico-syntactic patterns to a higher abstraction level, thereby enabling lexico-semantic patterns to identify more and more precisely events than lexico-syntactic patterns from text. We have developed rules based on lexico-semantic patterns used to find events, and semantic actions that allow for updating the domain ontology with the effects of the discovered events. Both the lexico-semantic patterns and the semantic actions make use of the triple paradigm that fosters their easy construction and understanding by the user. Based on precision, recall, and F_1 measures, we show the effectiveness of the proposed approach.

Keywords: Lexico-semantic patterns; Update actions; Ontologies; Financial events.

Reference to this paper should be made as follows: Borsje, J., Hogenboom, F., and Frasincar, F. (2010) ‘Semi-Automatic Financial Events Discovery Based on Lexico-Semantic Patterns’, *Int. J. Web Engineering and Technology*, Vol. 6, No. 2, pp.115–140.

Biographical notes: Jethro Borsje obtained a bachelor and a cum laude master degree in informatics and economics at the Erasmus University, Rotterdam, The Netherlands in 2006 and 2007 focusing on economics and ICT. During his bachelor and master degree programme he published research related to the Semantic Web. His master degree research was related to rule based ontology learning. From 2007 until early 2010 he worked as a software engineer for a research oriented firm, where he specialized in applying Semantic Web technologies to the financial domain. Other research interests include machine learning, natural language processing and pattern recognition. Currently he holds a position as an enterprise software engineer at a government agency.

Frederik Hogenboom obtained the cum laude master degree in economics and informatics from the Erasmus University Rotterdam, the Netherlands, in 2009, specializing in computational economics. During his bachelor and master programme, he published research mainly focused on the Semantic Web and learning agents. Currently, he is active within the multidisciplinary field of business intelligence and continues his research in a PhD track at the Erasmus University Rotterdam, the Netherlands. His PhD research focuses on ways to employ financial event discovery in emerging news for algorithmic trading, hereby combining techniques from various disciplines, amongst which Semantic Web, text mining, artificial intelligence, machine learning, linguistics, and finance. Other research interests are related to applications of computer science in economic environments, agent-based systems, and applications of the Semantic Web.

Flavius Frasincar obtained the master degree in computer science from “Politehnica” University Bucharest, Romania, in 1998. In 2000, he obtained the professional doctorate degree in software engineering from Eindhoven University of Technology, the Netherlands. He got the PhD degree in computer science from Eindhoven University of Technology, the Netherlands, in 2005. Since 2005, he is assistant professor in information systems at Erasmus University Rotterdam, the Netherlands. He has published in numerous conferences and journals in the areas of databases, Web information systems, personalization, and the Semantic Web. He is a member of the editorial board of the International Journal of Web Engineering and Technology.

1 Introduction

The days when professional brokers used to be the only ones operating on financial markets are long gone. Today, anyone can buy or sell equities, acting thus as a financial investor, by making use of specific Web-based financial information systems. As financial markets are extremely sensitive to news (Mitchell & Mulherin, 1994; Oberlechner & Hocking, 2004) one needs to continuously monitor which financial events take place.

Its low costs and high rate of adoption, made the Web one of the most popular platforms for news publishing. Unfortunately, the Web is a victim of its own success as thousands of news items are being published daily on different sources and with different content. This makes real-time news analysis a difficult process. In order to alleviate this problem, RSS news feeds aim to summarize and categorize the news information on the Web. Unfortunately, for financial investors interested in specific financial events (e.g., acquisitions, stock splits, dividend announcements, etc.), this categorization is too general to be of direct use in the decision making process. The manual identification of these events is a difficult and time consuming process that prevents the financial investor from promptly reacting on the market.

The Semantic Web provides the right technologies to classify the information in news items and make it available for both human and machine consumption. Being able to identify financial events from news items would help the trader to make a decision whether to react on the financial market. For example, recognizing a buy event such as "Google buys YouTube" in a news item would support the trader's decision to buy shares of YouTube as these will possibly increase in value after the buyout.

In this paper, we investigate how a user acting as a trader can identify the financial events of interest in titles extracted from RSS news feeds. The only requirement that we have for the user is that (s)he should be familiar with the financial domain as captured in an ontology. Due to his interest in buying and selling stocks of certain companies we assume that the user has a minimum knowledge of the financial markets. The user does not have to be familiar with Semantic Web technologies, the domain ontology will be presented in a graphical manner as a tree of concepts and concept relationships. Such an approach should allow the user to describe the events of interest, extract the event instances from news items, and update the domain ontology based on the effects of the discovered event instances.

During the design of our approach special attention is given to the user interface that should allow a simple interaction between the user and the system. Such an interface should enable a simple specification of the events of interest and event triggered-updates for the domain ontology. For this purpose we exploit the triple paradigm due to its intuitiveness and simplicity. Triples are used for defining lexico-semantic information extraction patterns that resemble simple sentences in natural language. In addition, triples are also used to express the event-triggered ontology updates.

In order to experiment with the proposed approach we have implemented a rule engine that allows rules creation, financial event extraction from RSS news feed headlines, and ontology updates. The financial event recognition is a semi-automatic process, where the user needs to manually validate the automatically discovered events before the ontology updates are triggered. In this way, we make sure that the ontology is not modified based on incorrectly discovered events. The effectiveness of the approach is measured by computing the accuracy, error, precision, recall, F_1 measure, and usefulness of the automatically discovered financial events from RSS news feeds.

The contribution of this paper is twofold. The work presented here is an extension of previous work on news personalization (Borsje et al., 2008; Frasincar et al., 2009; Schouten et al., 2010). The research presented in (Borsje et al., 2008; Frasincar et al., 2009) does not make use of lexico-semantic patterns, and merely identifies concepts instead of events. Similarly to this paper, in (Schouten et al., 2010), we do identify events using lexico-semantic patterns. However, here we provide more details on lexico-semantic patterns, i.e., their definition and evaluation, and also, we discuss three new models for event recognition, i.e., strict, relaxed, and hybrid models.

Section 2 continues with presenting related work on semi-automatic event recognition in news items. Section 3 explains the details of our approach, whereas Section 4 introduces the rule engine we have implemented. The rule engine is evaluated in Section 5. Finally, we draw conclusions in Section 6.

2 Related Work

A lot of research has already been done in areas related to (semi-)automatic recognition of financial events in news. For instance, several user interfaces and frameworks have been introduced that are designed for interpreting news feeds. This section continues with discussing some related work that is relevant for our research.

Vargas-Vera and Celjuska introduce a system that recognizes events in news stories (Vargas-Vera & Celjuska, 2004). This system identifies these events by means of information extraction and machine learning technologies and is based on an ontology. This ontology is populated sem-automatically. The system integrates both Marmot, a Natural Language Processing (NLP) tool, and Crystal, which is a dictionary induction tool used for concept learning. Also, a component called Badger is added, which is used for matching sentences with known concept definitions. Vargas-Vera and Celjuska demonstrate that their system works with the KMi Planet news archive of the Knowledge Media Institute (KMi).

StockWatcher (Micu et al., 2008) is an OWL-based Web application that enables the extraction of relevant news items from RSS feeds concerning the NASDAQ-100 listed companies using a customized, aggregated view of news. StockWatcher is able to rate the retrieved news items based on their relevance. Another news interpreter is Hermes (Borsje et al., 2008). In contrast to StockWatcher, this tool is not limited to interpreting a certain segment of financial news, as it also supports decision making in other domains that are highly dependent on news. Hermes aggregates news from several sources and filters relevant news messages using Semantic Web technologies.

PlanetOnto (Kalfoglou et al., 2001) represents an integral suite of tools used to create, deliver, and query internal KMi newsletters. Similar to the approach proposed here, domain ontologies are employed for the identification of events in news items. PlanetOnto uses a manual procedure for identifying information in news items, whereas we aim at semi-automatic information extraction from news items. Furthermore, PlanetOnto uses the ontology language OCML (Motta, 1999) for knowledge representation, while we aim to employ OWL, which is the standard Web Ontology Language (Bechhofer et al., 2004).

SemNews (Java et al., 2006) uses a domain-independent ontology for semi-automatically translating Web pages and RSS feeds into meaningful representations that are presented as OWL facts. For this purpose, OntoSem (Nirenburg & Raskin, 2004) is used, which is an NLP tool that performs lexical, syntactic, and semantic analysis of text. OntoSem has a specific frame-based language for representing the ontology and an onomasticon for storing proper names. In our framework, both the input ontology and the facts extracted from news items are represented in OWL. Our solution proposes to use a semantic lexicon instead of an onomasticon, which is a richer knowledge base that can better support the semantic analysis of text.

Many of the current approaches for automating information extraction from corpora, for instance PANKOW (Cimiano & Staab, 2004) and OntoCoSemWeb (Baazaoui-Zghal et al., 2007), use rules that are based on lexico-syntactic patterns as proposed by Hearst (1992; 1998). An existing ontology is used to extract pairs of related concepts in order to find hyponym and hypernym relations. These relations are found by applying regular expression patterns in free text. As lexico-syntactic rules do not take into account the semantics of the different constructs involved in a pattern we do find such an approach limited. Our solution exploits lexico-semantic patterns, which remove some of the ambiguity inherent to the lexico-syntactic rules. In addition, the proposed rules provide a higher abstraction level than lexico-syntactic rules, making their development and maintenance easier.

When it comes to updating knowledge bases, one could define action rules that are to be executed after patterns have matched. These actions can be expressed in various ways. A suitable candidate for implementation in our learning framework would be the Semantic Web Rule Language (SWRL) (Horrocks et al., 2004). SWRL is a rule language based on a combination of both OWL, and RuleML (Boley et al., 2001). Rules in SWRL are an implication between an antecedent (body), and a consequent (head). The rules can be read as follows: if the conditions in the body hold, then the conditions specified in the head must also hold.

There are some drawbacks regarding the usage of SWRL in our framework. First of all, SWRL can be used to add individuals and property instances to an ontology, but retractions are not allowed. This means it is impossible to remove individuals or instances of properties from an ontology using SWRL. Negation is also not supported at this moment. SWRL can be used in Protégé together with the Jess plugin, and the Racer or Pellet inference engines (Golbreich & Imai, 2004). However, good documentation on the use of SWRL is lacking and Racer is not open source.

Another alternative to express actions is by using a self-defined syntax. This gives us the freedom to fully customize both the storing and execution mechanisms. However, doing so is a tedious job. Besides that, it would be better to adhere to existing standards in order to be able to easily reuse this framework in another context, which brings us to another alternative, i.e., the usage of the triple format. The triple paradigm is compatible with existing Semantic Web standards and helps improving interaction between the user and the system due to its intuitiveness and simplicity.

A triple consists out of a subject, a predicate, and an object. Triples can be connected to each other to form a path expression which makes the use of triples both flexible and expressive. Representing actions in the form of triples facilitates the use of powerful path expressions which enables the use of both simple (single triple) and more complex (multiple triples, path expression) actions. Using this approach an action becomes a sequence of triples. These triples can then be used in SPARQL (Prud'hommeaux & Seaborne, 2008) queries, because a SPARQL query essentially consists of triples.

Human intervention is needed to supervise the ontology learning (i.e., knowledge base updating) process (Aussenac-Gilles, 2005), since NLP is error prone, and therefore humans are needed to validate the result. However, NLP is suitable for pruning large corpora of texts, thereby extracting relevant information. This information can serve as a basis for suggestions on how to identify relevant events and subsequently update a knowledge base with the events effect. These suggestions should then be validated by the user, thereby creating a semi-automated information extraction framework.

3 Event Rules

Our ontology learning framework uses rules to find events, and executes actions based on these events. Event detection is based on automatically scanning news headlines for specific lexico-semantic patterns and validation by the user. The execution of actions after detecting events allows users to use cause-and-effect reasoning in the ontology update process. An event, e.g., one company buys another company, causes one or more changes in the real world. For instance, the products of two companies do no longer compete with one another, the bought company ceases to exist in its old form, etc. Using our rule syntax, users can model these events and the resulting changes, thereby creating an information extraction system able to detect relevant events and correspondingly update the existing knowledge.

We make use of an OWL ontology to store the rules, because this enables easy integration with already existing ontologies. We have formulated a rule syntax, which facilitates the use of classes from other ontologies in the construction of lexico-semantic patterns defining the events. A rule consists of two parts: a *pattern*, which can have multiple lexical representations, and one or more *actions*, which can be executed once the pattern has been matched. In terms of first order logic the pattern is the antecedent, and the actions are the consequents. We now continue to describe the syntax and semantics of patterns and actions.

3.1 Lexico-Semantic Patterns

The lexico-semantic pattern of a rule is used to mine text for the occurrence of a specific event. Such a pattern has a triple format and consists of a *subject*, a *relation*, and an optional *object*. The subject and the object are the syntactic arguments of the relation, which describe the possible participants in the event. In our implementation, the subject and object are OWL classes that reside in the ontology and focus on information extraction and not learning. The OWL individuals (i.e., instances) of these classes are the possible participants in the event. The relation is an OWL individual of the predefined OWL class `kb:Relation`. The object is optional, because there are situations in which only a subject and a relation are enough to trigger an action. An example of such a pattern is shown in Fig. 1.

Figure 1 Lexico-semantic pattern example

```
[kb:Company] kb:goes_bankrupt
```

Multiple lexical representations describing the same event can be derived from the same pattern. These representations are used in the pattern matching process, which is done in several consecutive steps. First of all, the pattern that is associated with a specific rule is retrieved. Then, all the semantic classes are substituted by the participants which they describe (i.e., their instances or concepts). The third step substitutes both the participants (subject and object) and the relation for all the lexical representations by which they are denoted. We illustrate this process with an example pattern, which we call the buy pattern. This pattern is depicted in Fig. 2.

Figure 2 Lexico-semantic pattern for buy events

```
[kb:Company] kb:buys [kb:Company]
```

In this pattern, [`kb:Company`] is the URI of a class in the OWL ontology and we define instances of the class with [and]. Furthermore, `kb:buys` stands for the lexical representation of the predicate and is the URI of an individual of type `kb:Relation` in this ontology. After this pattern has been retrieved from the rules ontology the second step of the process replaces [`kb:Company`] by all class instances. If there are three companies in the ontology, which are `kb:Roche`, `kb:JohnsonAndJohnson`, and `kb:AkzoNobel`, the patterns as demonstrated in Fig. 3 are created.

Figure 3 Resulting patterns of lexico-semantic pattern expansion for subject/object

```
kb:Roche kb:buys kb:JohnsonAndJohnson
kb:Roche kb:buys kb:AkzoNobel
kb:JohnsonAndJohnson kb:buys kb:Roche
kb:JohnsonAndJohnson kb:buys kb:AkzoNobel
kb:AkzoNobel kb:buys kb:JohnsonAndJohnson
kb:AkzoNobel kb:buys kb:Roche
```

This step results in several patterns which are all constructed using OWL individuals, which all have one or more lexical representations. The next step is to substitute the OWL individuals for their different lexical representations. In our case we assume that the companies have only one lexical representation, and that the `kb:buys` relation has two lexical representations: "buys" and "acquires". This step results in the lexical representations of the buy pattern presented in Fig. 4.

All these lexical representations can then be used in mining the corpus for the occurrence of the pattern. If one occurrence of a pattern is found (i.e., we found an instance of the buy event), it is interpreted as an indicator of a possible change in the real world. However, one indicator is not a good basis for event identification. Therefore, we construct a mechanism that gathers several different occurrences of the same pattern. These different occurrences should be from heterogeneous sources – e.g., different news feeds – and they should also be in a specific time span. If, for example, different news feeds contain news items in which the buy

Figure 4 Resulting lexical representations of lexico-semantic pattern expansion for predicate

```
Roche buys Johnson & Johnson
Roche acquires Johnson & Johnson
Roche buys Akzo Nobel
Roche acquires Akzo Nobel
Johnson & Johnson buys Roche
Johnson & Johnson acquires Roche
Johnson & Johnson buys Akzo Nobel
Johnson & Johnson acquires Akzo Nobel
Akzo Nobel buys Johnson & Johnson
Akzo Nobel acquires Johnson & Johnson
Akzo Nobel buys Roche
Akzo Nobel acquires Roche
```

pattern for Roche and Akzo Nobel is recognized, this is a very strong indicator that an event occurred, because the more occurrences of a specific identified event in news headlines, the higher the likelihood the event has actually happened.

Based on identified occurrences, the proposed changes can be shown to the user for validation. It is up to the user to validate a certain identified event based on the news items in which the event was found. If the user confirms the event, our framework automatically executes the appropriate actions. After the execution of the actions the ontology is updated, thereby reflecting the changed reality.

Having an up-to-date ontology improves the information extraction process in the next run. For instance, a company that is not included in the knowledge base yet – e.g., Organon – can be discovered by the pattern in case it is for instance the object of a buy event. After updating the ontology new events can be identified, because Organon is now known as a company, and thus events with Organon as subject are also discovered.

3.2 Update Actions

As stated earlier, the rule-based information extraction framework which we propose uses actions to facilitate the event-triggered ontology updates. One or more update actions are associated with a pattern to form a rule, which can be executed once the pattern is found. The goal of these actions is to enable knowledge engineers and domain experts to express their knowledge in a simple yet expressive way by combining actions with patterns. To be able to make use of these actions, their syntax and semantics must be defined.

We can distinguish between two kinds of update actions: *add actions* and *remove actions*. Both of these actions either apply to an *OWL individual* or an *OWL property*. This basically means that there are four different types of actions: adding an individual, adding a property instance, removing an individual, and removing a property instance. Because of the in Section 2 mentioned drawbacks of alternatives like SWRL (Horrocks et al., 2004) and self-defined syntaxes, as well as the high compatibility with existing Semantic Web standards of the triple paradigm, we opt for the usage of the latter for action rule definition. Update actions are defined as a sequence of triples, which are used in SPARQL (Prud'hommeaux & Seaborne, 2008) queries. For example, removing and creating a property instance can be done analogous to the code (SPARQL templates) depicted in Figs. 5 and 6, respectively.

Figure 5 Simple SPARQL template remove query

```
REMOVE <subject> kb:hasCEO ?x
```

Figure 6 Simple SPARQL template add query

```
CONSTRUCT <subject> kb:hasCEO <object>
```

In order to create more expressive path expressions a WHERE clause can be used, which is exemplified in the code snippet from Fig. 7, which depicts an action where any `kb:competesWith` property instance between subject and object products is removed.

Figure 7 Advanced SPARQL template remove query

```
REMOVE ?x kb:competesWith ?y
WHERE
{
<subject> kb:hasProduct ?x
<object> kb:hasProduct ?y
}
```

These examples are not SPARQL queries, but SPARQL templates to be instantiated at run-time. The `<subject>` and `<object>` are to be replaced with the subject and object matches by the lexico-semantic pattern. This automatically happens when the action is being executed. It should be noted that the actions order is of importance for update execution. The order in which actions are specified is given by the designer so that the desired updates on the ontology are implemented. For example, a rule of thumb is that delete actions should precede insert actions as the opposite ordering of actions would possibly remove the new information from the ontology. No special algorithms are employed in order to determine the correct order of updates, as at the moment, our update actions have limited complexity.

SPARQL can be used for implementing the actions using the convenient triple format. SPARQL supports the use of a CONSTRUCT clause, and a WHERE clause, together with the use of the triple format. Based on these factors SPARQL is ideal for implementing the add actions (i.e., add individuals and add property instances). The problem with a SPARQL implementation lies in the fact that at this moment SPARQL does not support removal operations. This means that it is impossible to use SPARQL for the removal of individuals and properties. There are plans for adding removal functionality to SPARQL in the future by means of SPARQL/Update (Seaborne & Manjunath, 2007), which is already being implemented in Jena (McBride, 2002) and ARQ (Seaborne, 2010). We have added these SPARQL extensions to the SPARQL templates. Recent developments in SPARQL have lead to SPARQL 1.1 (Harris & Seaborne, 2010) and a new SPARQL

1.1 Update (Schenk & Gearon, 2010), which also allow for aggregates, subqueries, negation, expressions in the SELECT clause, and ontology updates. Incorporating these new functionalities into this research is subject of future work.

4 Rule Engine

This section presents our rule engine able to execute the event rules. The rule engine supports several actions: mining text items for patterns, creating an event if a pattern is found, determining the validity of an event by the user, and executing appropriate update actions if an event is valid.

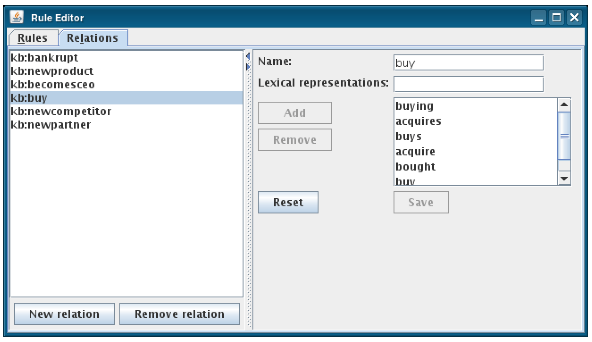
The engine consists of multiple components, i.e., a rule editor, which is described in Section 4.1, an event detector, which is presented in Sections 4.2 and 4.3, a validation environment, which is explained in Section 4.4, and also an action execution engine which is discussed in Section 4.5.

Using the editor, the user can construct the event rules. The event detector is used for mining text items (in our case news item headlines from RSS feeds) for the occurrence of the lexico-semantic patterns of the event rules (i.e., the event detector mines text items for occurrences of events). Using the validation environment, users are able to determine if the found events are valid. They can also modify the events in case the event detector made an error. If an event is validated the action execution engine is used to perform the updates, associated with the rule used for finding the event.

4.1 Rule Editor

The rule editor in our application allows knowledge engineers to construct the event rules which are used in the information extraction process. Using this editor, relations can be created and patterns can be formulated, based on the domain ontology. Fig. 8 shows the user interface which is used when the users want to add or modify relations. Using this interface, relations and their lexical representations (synonyms) can be created and modified. Each relation is an individual of the class `kb:Relation` in our knowledge base. This class contains all the relations for which the user can define event rules.

Figure 8 The rule editor, showing the interface for editing relations

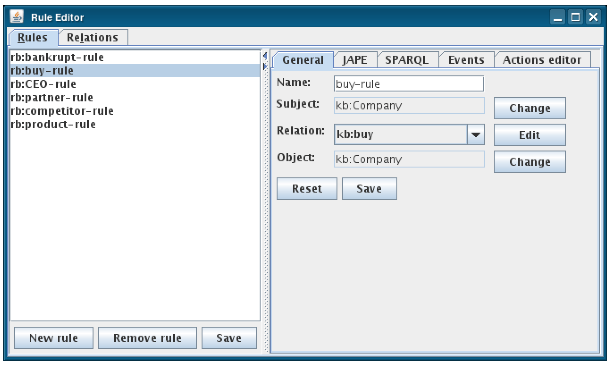


kb:bankrupt
kb:newproduct
kb:becomesceo
kb:buy
kb:newcompetitor
kb:newpartner

Name: buy
Lexical representations:

buying
acquires
buys
acquire
bought
buy

Figure 9 The rule editor, showing the interface for editing lexico-semantic patterns



rb:bankrupt-rule
rb:buy-rule
rb:CEO-rule
rb:partner-rule
rb:competitor-rule
rb:product-rule

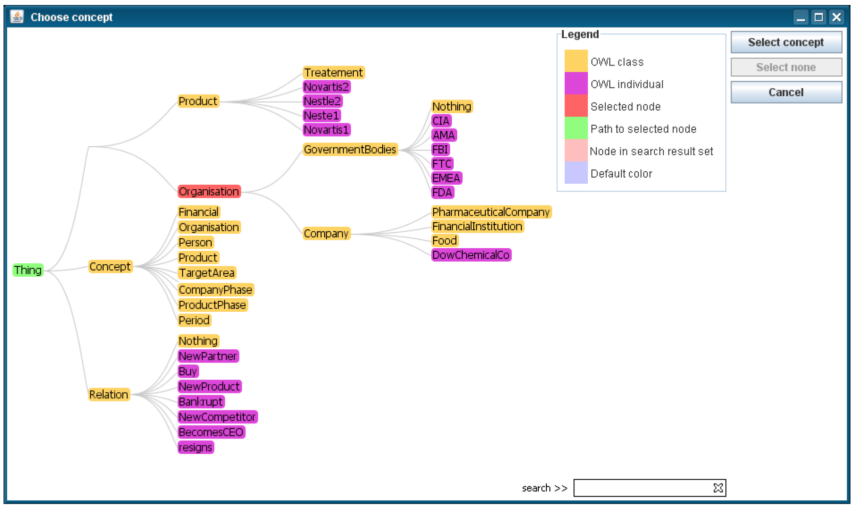
Name: buy-rule
Subject: kb:Company
Relation: kb:buy
Object: kb:Company

Fig. 9 shows the user interface for creating and modifying the lexico-semantic patterns. Through this interface the pattern of the rule can be created by choosing a subject, a relation, and an object. In this figure the buy rule is depicted, which describes how a buy event can be found.

Selecting subjects and objects is done using an interactive tree of the concepts and concept relationships of the domain ontology, as shown in Fig. 10. This tree lets the user browse intuitively through the ontology structure because of its visual support, without requiring an extensive knowledge of ontologies for the user.

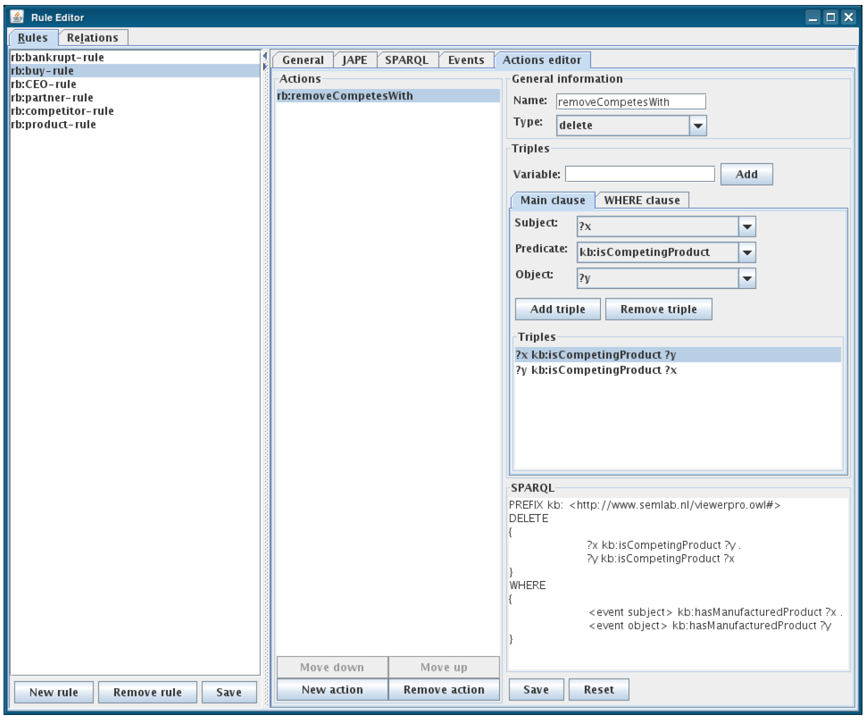
We have also implemented an editor for the update actions that are associated with the event rules. Fig. 11 shows the user interface which is used to create and modify these actions. In this figure the instruction to remove any *competesWith*

Figure 10 The rule editor, showing the interface for choosing concepts



Legend
🟨 OWL class
🟪 OWL individual
🟥 Selected node
🟩 Path to selected node
🟥 Node in search result set
🟪 Default color

Figure 11 The action editor



rb:bankrupt-rule
rb:buy-rule
rb:CEO-rule
rb:partner-rule
rb:competitor-rule
rb:product-rule

rb:removeCompetesWith

Name: removeCompetesWith
Type: delete
Variable:
Subject: ?x
Predicate: kb:isCompetingProduct
Object: ?y

Triples
?x kb:isCompetingProduct ?y
?y kb:isCompetingProduct ?x

```
PREFIX kb: <http://www.semlab.nl/viewerpro.owl#>
DELETE
{
            ?x kb:isCompetingProduct ?y .
            ?y kb:isCompetingProduct ?x
}
WHERE
{
            <event subject> kb:hasManufacturedProduct ?x .
            <event object> kb:hasManufacturedProduct ?y
}
```

property instance between products of the companies is depicted. The action editor allows the user to specify what kind of action (s)he wants to create (add or delete).

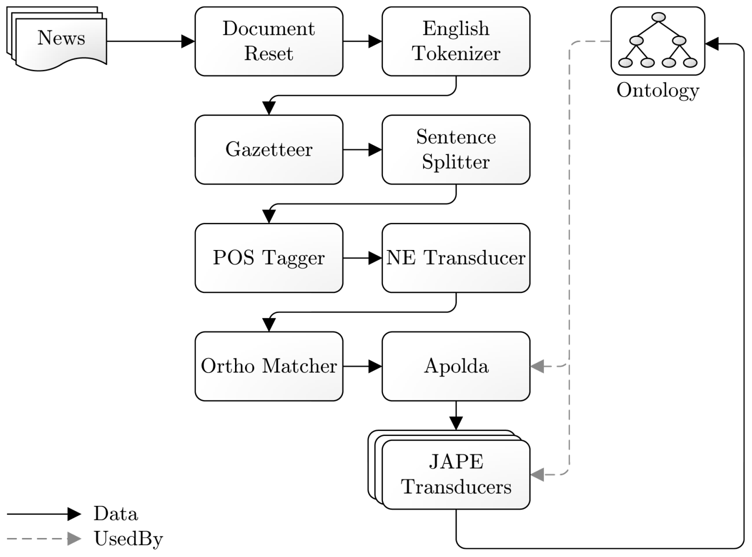
Subsequently, the main clause and the WHERE clause of the action can be created. The main clause contains the triples that are to be created, altered, or removed. These triples contain variables as subject or object, which are defined in the WHERE clause. Together, the main and WHERE clause form the entire instruction set for a specific add or remove action.

4.2 GATE Pipeline

The actual pattern matching for event detection is done using GATE (General Architecture for Text Engineering), which is an environment supporting the research and development of language processing software (Cunningham et al., 2002). GATE provides its users with numerous components (called resources), which can be used to construct a language processing application. It also allows its users to develop their own components or to extend the existing ones. In our framework, each news item will be pushed through this pipeline in order to check whether one of the lexical representations of an arbitrary lexico-semantic pattern of a rule can be matched.

The GATE pipeline, as depicted in Fig. 12 is comprised of several components, some of which are distributed with GATE by default, whereas others are custom-made by third parties or by us. The used default GATE components are Document

Figure 12 The GATE pipeline



Reset, ANNIE English Tokenizer, ANNIE Gazetteer, ANNIE Sentence Splitter, ANNIE Part-Of-Speech (POS) Tagger, ANNIE Named Entity (NE) Transducer, and the ANNIE Ortho Matcher. A third party component is added to these default components, i.e., Automated Processing of Ontologies with Lexical Denotations for Annotation (Apolda) (Wartena, 2010). The last three components of the pipeline are JAPE Transducers that we have created ourselves: Pre-pre-processing JAPE Transducer, Pre-processing JAPE Transducer, and the Pattern Matching JAPE Transducer.

The Document Reset component resets the document to its original state by removing all the annotation sets and their corresponding annotations. An annotation set contains multiple annotations and each annotation has multiple instances. An example of an annotation is “Sentence”, which has all sentences in a text as its instances. Another example is “Organization”, to which all organizations in the text that are identified by GATE are connected as instances. An annotation set is used to group several annotations into one set. These annotation sets can be used to separate annotations which are not related. The Document Reset component is used at the beginning of the pipeline to remove all old annotations.

The ANNIE English Tokenizer component splits the text into simple tokens such as punctuation, spaces, numbers, and words of different types (e.g., words in uppercase and lowercase). The English Tokenizer consists of a normal tokenizer and a JAPE transducer. This transducer is used to concatenate various tokens to create constructs like “50’s”, and “don’t”. We make use of the alternate rule set for the ANNIE English Tokenizer, which is also distributed with GATE. When using this alternate rule set, the tokenizer recognizes hyphenated words better. This means that words like “Sanofi-Aventis” are annotated as one token. Because of the tokenization, more advanced analysis can be performed in components later on in the GATE pipeline, e.g., by Apolda.

The ANNIE Gazetteer is used in order to classify certain types of words, such as for instance cities, organizations, countries, and dates. The gazetteer contains lists of words for which a major type, and an optional minor type are defined. In case words on this list are identified in the text, they are annotated with their corresponding major and minor types. The major type is used for providing general information about the annotation, whereas the minor type provides a more detailed description about the annotation. For example, an organization is a more general form of a company. Once the text is annotated using the gazetteer, JAPE grammars match all organizations by specifying the general major type `kb:Organization`. All companies are matched by specifying the more specific minor type `kb:Company`.

The ANNIE Sentence Splitter is used to split the text into sentences, which are subsequently used by the ANNIE POS-tagger (Hepple, 2000). This tagger is a modified version of the Brill tagger (Brill, 1992). It provides Part-Of-Speech (POS) tags as an annotation for every word or symbol in the text. POS-tags can be used to identify verbs, singular nouns, plural nouns, adjectives, etc.

The ANNIE Named Entity Transducer (or Semantic Tagger) makes use of annotations assigned in earlier phases. Based on these annotations named entities can be found and annotated. For example, the two successive words “Larry”, and “Page” are annotated together, thereby creating the named entity “Larry Page” that is annotated as type `kb:Person`. Using the NE transducer, new organizations, persons, locations, and dates can be found. Companies are a specific type of organization, so they can also be found.

The ANNIE Ortho Matcher (also known as the Orthographic Coreference component or NameMatcher) is used for adding identity relations between named entities found by the Named Entity Transducer. Using the Ortho Matcher strings that refer to the same thing can be found, e.g., “IBM” and “Big Blue” both refer to the company IBM.

Apolda (Wartena, 2010) is a component which has similarities with the ANNIE Gazetteer. The biggest difference lies in the fact that Apolda uses terms from an ontology to classify words instead of gazetteer lists. Apolda searches the text for occurrences of OWL annotation properties – these are the concept lexical representations – of the classes and instances of an ontology. The found matches are annotated with the name of the OWL instance (or class) against which the piece of text is matched. Using this component text can be mined for the occurrences of terms from an OWL ontology.

After annotating the text with Apolda, each term in the text that corresponds to an OWL class or OWL individual is labeled with the corresponding class or individual. However, in our matching process we only want to use OWL individuals, and thus we employ a JAPE transducer for pre-pre-processing, which removes all class annotations from the annotation set.

Each rule is associated with a lexico-semantic pattern, which has multiple lexical representations and is used to identify events in text. In order to be able to find the lexical representations of a lexico-semantic pattern, this pattern is dynamically converted to two JAPE grammars each time a GATE pipeline is being executed. These two JAPE grammars are executed successively in two separate phases.

The grammar in the first phase (the pre-processing phase) is used for copying all relevant annotations to a temporary annotation set. By using a temporary annotation set for the relevant annotations, the next phase automatically discards the non-relevant annotations, because these non-relevant annotations are not copied to the temporary annotation set. Only items that occur in the pattern of the rule are relevant, so for a buy rule (depicted in Fig. 2) only company annotations and annotations that reflect the buy relation are copied to the temporary annotation set.

The second phase is all about pattern matching. The JAPE grammar is employed for matching a pattern based on the annotations in the temporary annotation set created in the previous phase. In this pattern matching phase, the JAPE grammar is used to identify full or partial matches of the lexico-semantic pattern based on the relevant annotations.

4.3 Event Detector

In order to query the news items for occurrences of events, we mine all the news items in the RSS feeds that have not yet been processed by our event detector, which is based on the GATE pipeline. If a lexico-semantic pattern associated with an event rule is found in the text, we create an event instance in case it does not already exist. Provided that an event instance is found, the news item is added to the existing event instance. For example, we might find the three news items from Fig. 13.

Figure 13 Example of possible relevant news items for buy events

Pharmaceuticals giant *AstraZeneca* **buys** *MedImmune* for $15.6 billion
QIAGEN Signs Agreement for the **Acquisition of** *eGene*
AstraZeneca to **buy** *MedImmune*

In the latter figure, italic texts denote companies identified by the user, whereas bold texts denote relations identified by the user. We parse these news items with respect to the buy pattern depicted earlier in Fig. 2.

The pattern matching process first identifies all relevant concepts in a news item, with respect to the subject and the object of a pattern. In the case of the previous three news items and the buy pattern this means that all the relevant companies are found based on their lexical representations stored in the ontology. The news items are also queried for the occurrence of unknown companies, which are not yet stored in the ontology. This is done using the ANNIE Gazetteer, the ANNIE Named Entity Transducer, and the ANNIE Ortho Matcher, which are all GATE components. Using these components, the engine automatically identifies unknown companies, thereby learning at the instance level of the ontology.

After that, the event detector looks for the presence of a relevant relation. In the case of the buy pattern the event detector tries to locate a lexical representation of the buy relation in the news item. If both a subject, and an object together with a relation are found in the news item, a new event is created, if this does not already exist. If the event already exists, the news item is added to the existing event.

Based on the buy pattern previously introduced, our tool identifies two events, depicted in Fig. 14. In the latter figure, the italic texts denote lexical representations of companies as found by the event detector. The non-italic text denotes a company that is not recognized by the event detector. Now the first event is a *full match*, because we have identified both a subject and an object. The second event is a *partial match*, because we were only able to identify a subject (the company *QIAGEN*), and the buy relation (based on its lexical representation "Acquisition of").

Figure 14 Example of identified relevant news items for buy events

AstraZeneca $\overset{buys}{\rightarrow}$ *MedImmune* (with news items #1 and #3)
QIAGEN $\overset{buys}{\rightarrow}$ eGene (with news items #2)

The buy pattern requires both a subject and an object to be valid, so the second event can not be validated automatically at this point, because the object is missing. However, when the user reviews the event, (s)he might recognize *eGene* as being a company based on both his own knowledge and the context of the news item in which *eGene* appears. Based on this event (and the corresponding news item headline) the user can add *eGene* to the ontology by manually validating the event, after which this event has both a subject (*QIAGEN*) and an object (*eGene*), making the event now a full match.

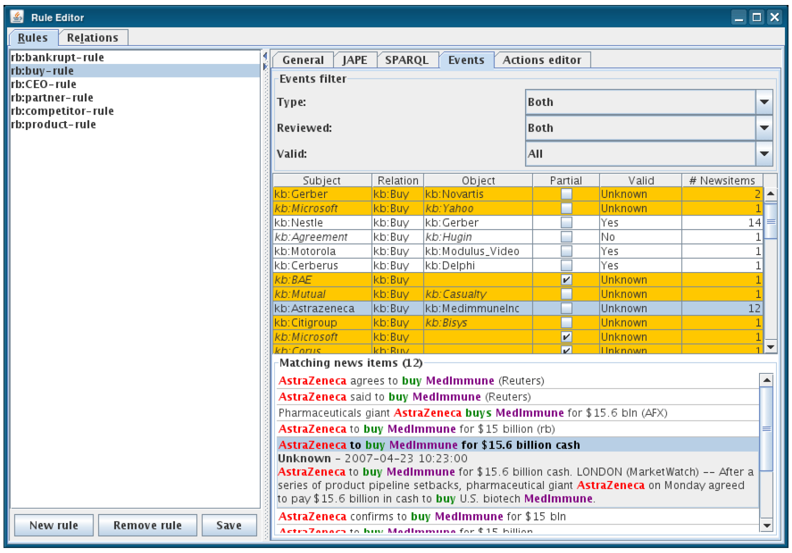
Ideally, *eGene* would be recognized by GATE automatically as an unknown company. However, at this point we are not capable of recognizing all unknown companies automatically. GATE recognizes companies based on Gazetteer lists and Java Annotation Patterns Engine (JAPE) rules (Cunningham et al., 2002). These rules look in the text structure. If, for example, a company is found in an enumeration together with other companies, GATE is capable of recognizing the unknown company as a company. Based on the sentence "Profits Google, Microsoft, and eGene increase." GATE is able to deduce that "eGene" is a company, because of its context. Using the short length RSS news item headlines GATE is not always capable of identifying unknown companies due to the lack of text structure.

4.4 Event Validation

Identified events are stored in an ontology, together with the news items in which they are found. In the validation environment the user is able to review all the identified events. The user is able to validate both full and partial matches. Once an event is validated, the associated actions are executed, thereby modifying the ontology based on the identified event. In case of a validation of a partial match (containing unknown concepts), not only the associated actions are executed, but also the concepts that were unknown up until now are added to the knowledge base automatically.

Fig. 15 gives an overview of the validation environment. A colored row indicates an identified event that has not yet been validated by the user, whereas a white

Figure 15 The validation environment of the rule engine



rb:bankrupt-rule
rb:buy-rule
rb:CEO-rule
rb:partner-rule
rb:competitor-rule
rb:product-rule

Events filter

Type:	Both
Reviewed:	Both
Valid:	All

Subject	Relation	Object	Partial	Valid	# Newsitems
kb:Gerber	kb:Buy	kb:Novartis	☐	Unknown	2
kb:Microsoft	kb:Buy	*kb:Yahoo*	☐	Unknown	1
kb:Nestle	kb:Buy	kb:Gerber	☐	Yes	14
kb:Agreement	kb:Buy	*kb:Hugin*	☐	No	1
kb:Motorola	kb:Buy	kb:Modulus_Video	☐	Yes	1
kb:Cerberus	kb:Buy	kb:Delphi	☐	Yes	1
kb:BAE	kb:Buy		✓	Unknown	1
kb:Mutual	kb:Buy	*kb:Casualty*	☐	Unknown	1
kb:Astrazeneca	kb:Buy	kb:MedimmuneInc	☐	Unknown	12
kb:Citigroup	kb:Buy	*kb:Bisys*	☐	Unknown	1
kb:Microsoft	kb:Buy		✓	Unknown	1
kb:Corus	kb:Buy		✓	Unknown	1

Matching news items (12)

AstraZeneca agrees to buy MedImmune (Reuters)
AstraZeneca said to buy MedImmune (Reuters)
Pharmaceuticals giant AstraZeneca buys MedImmune for $15.6 bln (AFX)
AstraZeneca to buy MedImmune for $15 billion (rb)
AstraZeneca to buy MedImmune for $15.6 billion cash
Unknown - 2007-04-23 10:23:00
AstraZeneca to buy MedImmune for $15.6 billion cash. LONDON (MarketWatch) -- After a series of product pipeline setbacks, pharmaceutical giant AstraZeneca on Monday agreed to pay $15.6 billion in cash to buy U.S. biotech MedImmune.
AstraZeneca confirms to buy MedImmune for $15 bln
AstraZeneca to buy MedImmune for $15 billion

row indicates that the user has either confirmed or rejected the validity of a specific identified event. The italic subjects and objects of identified events are concepts which are unknown at this point; they are discovered in the news items by the GATE pipeline but are not present in the domain specific knowledge base.

4.5 Action Execution Engine

If a user validates an event the appropriate actions should be executed. This is taken care of by the action execution engine which is based on four steps: retrieving the appropriate actions, instantiating the `<subject>` in the actions' SPARQL templates for the actual subjects of the events, substituting the `<object>` in the actions for the actual objects of the events (if there are any), and finally executing the actions using a SPARQL engine.

The actions associated with the event rule whose pattern is matched by the event are first retrieved from the rule base. An example of such an action can be found in Fig. 16. This action is associated with the buy rule. The `kb:competesWith` property instance between two products whose producers do no longer compete with each other (because one bought the other) is removed. It should be noted that the actions need to be ordered correctly in order to prevent erroneous data to be left in the knowledge base.

The template action illustrated in this figure can not be executed directly. Before the action is ready for execution, the `<subject>` and `<object>` must be substituted for the actual subject and object as found in the event. If for example *Akzo Nobel* buys *ICI*, the action will look like the one depicted in Fig. 17 after substituting the subject and object with the appropriate names. This specific action is executed using ARQ (Seaborne, 2010) that supports SPARQL/Update (Seaborne, 2010).

Figure 16 Abstract action before substitution

```
REMOVE ?x kb:competesWith ?y
WHERE
{
<subject> kb:hasProduct ?x
<object> kb:hasProduct ?y
}
```

Figure 17 Action after substitution

```
REMOVE ?x kb:competesWith ?y
WHERE
{
kb:AkzoNobel kb:hasProduct ?x
kb:ICI kb:hasProduct ?y
}
```

5 Evaluation

We now continue with the evaluation of the effectiveness of the rule engine. First, we introduce the data set and performance measures used in Sections 5.1 and 5.2, respectively, after which we discuss the events used in our evaluation in Section 5.3. Finally, Section 5.4 elaborates on our experimental results.

5.1 Data Set

In order to validate the results of the learning process, we compare the (several hundreds of) events identified by the rule engine with those identified by domain experts, based on the same set of 1,736 news items headlines, which are retrieved from several RSS news feeds (scraped from MarketWatch, Yahoo!, and Reuters). Subsequently, a data set of news items received from Reuters via Citigroup is used. All these news items are aggregated into one data set NI, which is used in our experiments on effectiveness of the proposed approach.

It should be noted that the quality of the event detector is greatly dependent on its inputs. This means that without a high quality knowledge base, which contains the proper lexical representations for both the concepts and the relations, the amount of successfully identified events will dramatically decrease. Therefore, we construct a detailed and domain-specific initial knowledge base that is based on the inter-expert agreement of three domain experts, containing approximately fifty distinct classes and hundreds of individuals and properties. The process of developing the ontology is an incremental middle-out approach. First, the most salient concepts are defined and then these are refined using generalization/specialization towards the top/bottom of the ontology. We validate our domain ontology using the OntoClean methodology (Guarino & Welty, 2002). Furthermore, the clarity of the parsed text is also important. In case the text that is being mined is very unstructured, it is hard to match patterns. When using an adequate knowledge base and a well-formed English text, we expect that the event detector performs fairly well.

5.2 Performance Measures

Performance evaluation of the ontology learning process is a non-trivial task in this field of research, as there are no commonly accepted predefined measures (Schutz & Buitelaar, 2005). We use the following (well-known and relevant) measures: accuracy (A), precision (P), recall (R), and F_1. Moreover, we define a measure for usefulness U. Using these measures we evaluate the result of the rule engine E_D against a reference standard E_{RS}, as defined by our domain experts. This reference standard is a set of events created by manually identifying events in the data set. Each event is supported by one or more headlines which denote this event.

Using E_{CI} as the set of correctly identified events and E_T as all identified events by either the event detector or the domain expert ($E_D \cup E_{RS}$), we can denote the accuracy A as:

$$\begin{aligned} A &= \frac{\mid E_{CI} \cup (NI - E_T) \mid}{\mid NI \mid} \\ &= \frac{\mid (E_D \cap E_{RS}) \cup (NI - (E_D \cup E_{RS})) \mid}{\mid NI \mid} \\ &= \frac{\mid NI - (E_D \triangle E_{RS}) \mid}{\mid NI \mid} \ , \end{aligned}$$

where $E_D \triangle E_{RS} = (E_D - E_{RS}) \cup (E_{RS} - E_D)$. Subsequently, we use the precision measure P for determining the amount of correctly classified news items by the event detector. We define P as the ratio of the number of correctly identified items E_{CI} and the number of identified items E_D:

$$P = \frac{\mid E_{CI} \mid}{\mid E_D \mid} \ .$$

The recall measure R is used for calculating the portion of news items in the data set that are correctly classified by the event detector. This ratio of E_{CI} and E_{RS} (i.e., events identified by the domain expert) can be denoted as:

$$R = \frac{\mid E_{CI} \mid}{\mid E_{RS} \mid} \ .$$

There is a trade-off between precision P and recall R, and therefore the F_1 measure is needed. The F_1 measure is applied to compute an even combination of precision and recall. When improving the event detector, the F_1 measure is the measure that should be maximized, as it takes both precision and recall into account. Values of F_1 can be calculated as:

$$F_1 = \frac{2 \times P \times R}{P + R} \ .$$

Finally, we also evaluate the degree of usefulness U of each run. It is determined by dividing the amount of useful news items belonging to events identified by the event detector (E_U) by the amount of news items belonging to all the events identified by the event detector (E_D). Hence:

$$U = \frac{\mid E_U \mid}{\mid E_D \mid} \ ,$$

where $E_U = E_{CI} \cup \{e \in E_D \mid \text{ e is partially valid}\}$, since useful events can either be valid or partially valid. The partially valid events, i.e., valid events which have at least the subject or object from the gazetteer, named entity transducer, or ontology, and the subject or object are missing (as opposed to valid events in which both subject and object are recognized by gazetteer, named entity transducer, or ontology), are quite useful, because the domain expert can complete them and make them valid (with respect to the ontology) with a minimum amount of effort. The domain expert has only to identify the missing valid subject or object.

5.3 Events

In our experiments, we will focus on finding events associated with the previously discussed buy rule (i.e., buy events as defined in Fig. 2), as well as a rule in the political domain, i.e., a rule indicating the visit of a (highly placed) person – a politician – to a country, as depicted in Fig. 18. In order to enable testing on a different domain, we add political concepts (e.g., `[kb:Politician]` and `[kb:Country]`) to our ontology.

Figure 18 Lexico-semantic pattern for visit events

```
[kb:Politician] kb:visits [kb:Country]
```

Hence, we do not only evaluate the engine performance on financial news items extracted from our RSS feeds, but also on political news messages originating from the same sources, in order to be able to assess the applicability of our approach to other domains. We choose to use these events in the evaluation process because they are the most frequently encountered events in our data set. As there are no major structural differences between these events for the field of finance and politics and other events in these fields, we focus on evaluating only the representative events.

We evaluate three different processing runs using our rule engine. In the first run, the rule engine looks for the patterns that are depicted in Figs. 2 and 18 in a strict manner. This means that the rule engine only looks for known companies, persons, and countries, and tokens which are explicitly annotated by GATE as such.

For the second run, we relax the restrictions on the pattern in such a way that the patterns also include tokens that are annotated by GATE as an organization, i.e., not specifically a company, or as unknown. GATE annotates tokens as unknown in case it knows the token is a named entity (e.g., in case the token is capitalized), but it does not know its specific type. Please note that at least the subject or object should be from the gazetteer, named entity transducer, or ontology.

Finally, the third run combines the approaches of the first and second run. The relaxed pattern is applied with the addition that either the subject or the object of the found event is a known company, person, or country from the ontology. This extra condition on either the subject or the object combines the relaxed pattern with a more strict approach.

5.4 Experimental Results

This section evaluates not only the performance of the lexico-semantic patterns, but also of their lexico-syntactic equivalents. Creating lexico-semantic patterns using our tool requires little effort, as the concepts used in the patterns are conveniently described in an ontology. Table 1 displays the pattern creation times in seconds for both the financial and the political patterns. Here, we distinguish between lexico-semantic and lexico-syntactic variants. For both patterns, it holds that equally well-performing lexico-semantic patterns are created about ten times faster than their lexico-syntactic equivalents. This is due to the fact that one has to include all synonyms, instances, etc., in the lexico-syntactic patterns that are automatically incorporated into the lexico-semantic patterns.

The difference in creation time is also reflected in the performance of lexico-semantic patterns when compared to lexico-syntactic patterns. With a given fixed creation time of five minutes, lexico-syntactic patterns seem to perform worse within both domains, as supported by Tables 2 and 3. Table 2 gives an overview of the calculated values for each of the previously defined measures for each run (i.e., using strict, relaxed, and hybrid lexico-semantic patterns). This table shows that applying a hybrid lexico-semantic pattern yields the best results in terms of accuracy, precision, and recall, whereas the usage of a strict pattern results in the worst results. Usefulness of patterns is relatively high for both lexico-syntactic and lexico-semantic patterns in all runs, but overall, lexico-semantic patterns perform better.

For all runs the overall accuracy is very good. The accuracy of the strict and relaxed patterns is almost equal, whereas the accuracy of the hybrid pattern is slightly better. These high and almost equal values are caused by the fact that most of the news item headlines are not associated with a buy event and thus the rule engine has successfully ignored a large portion of these news items in each run. When applying a relaxed pattern the event detector is capable of recognizing more companies than is the case when using a strict pattern. However, this results in some events that are out of the domain, e.g., “Thomson buys Reuters”. Although such an event is correctly recognized we do not want to use it in our information extraction framework, because it learns facts which are not in our domain, i.e., both Thomson and Reuters are not present in our domain specific knowledge base. Therefore we consider these events to be invalid in our context. When using the hybrid pattern, we have the restriction that either the subject or object must be

Table 1 Pattern creation times (in seconds) for lexico-syntactic and lexico-semantic patterns within the financial and political domains

Domain	Pattern	Creation time
Finance	Lexico-Syntactic	528 seconds
	Lexico-Semantic	63 seconds
Politics	Lexico-Syntactic	820 seconds
	Lexico-Semantic	76 seconds

Table 2 Experimental results within the financial domain

	Lexico-Syntactic Patterns			Lexico-Semantic Patterns		
Measure	Strict	Relaxed	Hybrid	Strict	Relaxed	Hybrid
Accuracy	0.92467	0.93065	0.95319	0.95334	0.95392	0.97005
Precision	0.18670	0.37862	0.61124	0.21795	0.38384	0.58065
Recall	0.10839	0.18677	0.20467	0.22667	0.50667	0.48000
F_1	0.13716	0.25014	0.30666	0.22222	0.43678	0.52555
Usefulness	0.65504	0.54137	0.78588	0.73077	0.58586	0.80645

a known company, thereby ensuring that the event is always in our domain. This reduces the error rate of the event detector and thus increases accuracy.

On our financial news, relaxed and hybrid patterns obtain higher precision than strict patterns. This is due to the fact that many company synonyms are not included (e.g., abbreviations or variations of "Incorporated", such as "inc", "inc.", "Inc", "Inc.", ", Inc", ", Inc.", etc.). In our experiments, this occurs frequently, so by relaxing the patterns we increase precision. The hybrid patterns also stand out in precision. The same rationale as for the accuracy applies here. The event detector recognizes more companies with the relaxed pattern, and thus is able to identify events that otherwise would not have been recognized or that would have been partial matches (because the subject or object could not be found). However, because of the restriction that either the subject or the object must be present in the knowledge base, we reduce the amount of invalid matches (due to the recognition of events outside our domain).

Applying the hybrid lexico-semantic pattern does not always result in better values for each measure, as can be derived from the recall measure. Here, the relaxed pattern performs slightly better than the hybrid pattern. When using the relaxed pattern, the event detector is capable of recognizing more events than with the strict pattern. This results in a higher recall for the relaxed run. The hybrid pattern also allows for more events to be identified than the strict pattern, but results in a slightly lower recall because of the restriction that either the subject or the object must be in the knowledge base. However, when combining the precision and recall measures into the F_1 measure, we observe that the hybrid pattern outperforms both the relaxed and strict pattern.

Table 3 Experimental results within the political domain

	Lexico-Syntactic Patterns			Lexico-Semantic Patterns		
Measure	Strict	Relaxed	Hybrid	Strict	Relaxed	Hybrid
Accuracy	0.88389	0.75960	0.82802	0.91304	0.82609	0.86957
Precision	0.76039	0.44105	0.57614	0.77778	0.47368	0.64286
Recall	0.35205	0.48795	0.44010	0.58333	0.75000	0.75000
F_1	0.48128	0.46331	0.49901	0.66667	0.58065	0.69231
Usefulness	0.84240	0.64755	0.73181	0.88889	0.68421	0.78571

Finally, we observe that the identified events (and associated news items) are highly useful when using the hybrid pattern, followed by the strict pattern, whereas the relaxed pattern clearly performs worse than the other two patterns. The reason for the latter observation is that the usage of the relaxed pattern introduces a lot of out-of-domain events, which we consider to be invalid. The hybrid pattern solves this problem due to its restriction that either the subject or the object of an event must be present in the knowledge base.

As shown in Table 3, compared to the financial domain, strict patterns are operating much better within the political domain. Again, lexico-semantic patterns outperform lexico-syntactic patterns with a limited construction time of five minutes. Also, in both cases the hybrid patterns yield promising results. High accuracy and usefulness values, which also holds for the financial domain, are verified by the results presented in Table 3. Differences between the results of both domains can be explained by the characteristics of the knowledge base and the patterns. Strict patterns only identify events when both subject and object (politician and country) can be matched to concepts in the knowledge base. In case of countries and politicians, this easier than with companies.

As our knowledge base contains many (not all) politicians and all countries, there are not many out-of-domain errors here. Generally, politicians are labeled as persons, and all countries are recognized. Therefore, on average, we achieve better scores within the political domain. Mainly, errors occur more often in the relaxed and hybrid patterns, e.g., when "visit" is used as a noun. Furthermore, some errors are made in case of complex terms beginning with country names, e.g., "US Open". Also, the snippet "Iran visit Bashar al-Assad" will not be recognized by any pattern, as the current patterns are not flexible enough to cope with inversions.

On the other hand, relaxed and hybrid patterns do discover events such as "The Queen visits Australia". Even though there is no person to be recognized in this pattern based on names, there is a reference to a person. This makes the event difficult to discover by strict patterns, but relaxed and hybrid patterns are able to identify these. Relaxed and hybrid patterns often fire in case there is no country to be matched, but a city, a person, or a company not known in the knowledge base. Due to this, precision and accuracy tend to be lower than is the case with strict patterns. The relaxed and hybrid patterns often return many events. Because of this, the recall is somewhat higher than the recall resulting from strict patterns.

While our experiments within the financial domain show that precision increases when relaxing patterns, we observe the opposite effect within our political domain. Precision decreases when using relaxed patterns, as many invalid events are recognized. Names and countries do not have many variations, as in news items, persons are frequently referred to using first and last names. Usually, nicknames, popular language, etc., are rarely used in these contexts. Also, countries do not have many different names. Therefore, invalid event recognition becomes more of an issue as the knowledge base is sufficient for providing names to link concepts to politicians and countries. The decrease of accuracies can also be explained by this phenomenon.

In terms of usefulness, strict patterns seem to yield the best results, but hybrid patterns also show similar usefulness. This is caused by the fact that the quality of the relaxed patterns and (to a lesser extent) the hybrid patterns is lower, as they frequently falsely identify events. Also, the obtained results for the political domain show that the usefulness of the hybrid patterns is lower than in de case of the financial domain.

6 Conclusions

In this paper we have proposed the use of lexico-semantic patterns for financial events extraction from RSS news feeds. This approach is implemented in a rule engine with which one can query news feed headlines using patterns. These patterns make use of financial ontologies, which leverage the existing lexico-syntactic patterns to a higher abstraction level. Lexico-semantic patterns are able to identify more events than lexico-syntactic patterns from corpora and can be constructed with considerably less effort. Furthermore, we have developed and presented rules based on lexico-semantic patterns, as well as actions that allow updating of the domain ontology with the effects of the discovered events, causing the domain ontology to reflect the most up-to-date information available on the financial world. In our work, we make use of the triple paradigm that allows an easy construction and understanding of rules by users.

The effectiveness of the proposed approach is measured using metrics, such as the precision, recall, and F_1 measures. We conclude that generally, the tool is effective because of its high accuracy, i.e., it recognizes true negatives well. We verified these results by testing on a different domain as well, i.e., politics. Also, we evaluated the usefulness of the lexico-semantic patterns on both domains, which yielded encouraging results.

For future research, we suggest a couple of directions. First of all, the event-triggered actions which are now used for ontology updates can also be used for other purposes. For example, we could send SMS messages or e-mail trader alerts once certain events are identified, thereby automatically notifying users of the occurrence of real-world events in a real-time manner. Secondly, currently, we have been focusing on processing headlines from RSS feeds instead of entire news items. The reason for this is that titles are usually well formulated and to the point. Moreover, titles can be processed in a limited amount of time. However, the drawback of this approach is that titles do not always contain all contextual information that is needed for proper understanding, which is likely to be solved by processing the entire news item. Therefore, future research into the possibilities of processing entire news items is suggested.

Furthermore, it would be interesting to conduct research on event chains, as usually, events are not isolated but they are part of a chain of events. For example, if we observe a process that eventually results in one company buying another, then in most cases this process is initiated by rumors on the interest of several companies in buying a certain company. If these rumors turn out to be true, the companies might engage in a bidding war. After this bidding war, the winner acquires the company for sale. In most cases, each phase in this process results in a flow of news items, which represent the successive events leading up to the final buy event. It would be interesting to formulate such chains of events in order to monitor the developments of specific domains over time. By identifying these patterns of events, forecasting of events can be done (Milea et al., 2008; Frasincar et al., 2010). If certain events appear, it is likely that other events also appear.

Related future work might include recognizing the difference in the temporal aspects of patterns (e.g., past tense, future tense, etc.) and using this accordingly when updating the knowledge base. Also, one could consider adding full negation support to the patterns. At the current moment, it is the user who decides if an event has been correctly found and it needs to trigger updates to the ontology. Hence, negations and temporal aspects are handled manually by either approving or denying events and their ontology updates. Updating the framework by using SPARQL 1.1 (Harris & Seaborne, 2010) and SPARQL 1.1 Update (Schenk & Gearon, 2010) would be desirable, as these also allow for aggregates, subqueries, negation, expressions in the SELECT clause, and update operations on RDF graphs. Furthermore, it would be worthwhile to investigate event ranking based on evidence. If an event is frequently identified, there is more evidence, and thus the event is more likely to be true than is the case with less evidence.

Additionally, the action editor needs to be improved. Related future research could include automatic pattern discovery, as currently patterns have to be manually formulated by the user. Automating this process would improve the usability of our solution. A possible approach might be to perform generalizations based on co-occurrence analysis. By analyzing which class instances co-occur frequently with certain relations, patterns can be formulated. Users can then validate these patterns and associate actions to them. This approach should result in (semi-)automatically generated rules that can be used in our rule engine.

Finally, the usage of powerful change patterns (Lösch et al., 2009) can be subject to further research. The patterns introduced in this paper operate on text and ontologies, while change patterns operate on graphs and ontologies. Perhaps in future work we could use the output of lexico-semantic patterns as input for expressive change patterns for ontology updating.

Acknowledgement

The authors are partially sponsored by the NWO Physical Sciences Free Competition project 612.001.009: Financial Events Recognition in News for Algorithmic Trading (FERNAT).

References

Aussenac-Gilles, N. (2005), Supervised Text Analyses for Ontology and Terminology Engineering, *in* 'Dagstuhl Seminar on Machine Learning for the Semantic Web (MLSW 2005)', pp. 1–7.

Baazaoui-Zghal, H., Aufaure, M.-A. & Mustapha, N. B. (2007), 'A model-driven approach of ontological components for on-line semantic web information retrieval', *Journal of Web Engineering* **6**(4), 309–336.

Bechhofer, S., van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D. L., Patel-Schneider, P. F. & Stein, L. A. (2004), 'OWL Web Ontology Language Reference – W3C Recommendation 10 February 2004'. Obtained through the Internet: `http://www.w3.org/TR/owl-ref/`, [accessed 1/5/2010].

Boley, H., Tabet, S. & Wagner, G. (2001), Design Rationale of RuleML: A Markup Language for Semantic Web Rules, *in* '1st Semantic Web Working Symposium (SWWS 2001)', pp. 381–401.

Borsje, J., Levering, L. & Frasincar, F. (2008), Hermes: a Semantic Web-Based News Decision Support System, *in* '23rd Annual ACM Symposium on Applied Computing (SAC 2008)', ACM, pp. 2415–2420.

Brill, E. (1992), A Simple Rule-Based Part of Speech Tagger, *in* '3rd Conference on Applied Natural Language Processing (ANLP 1992)', Association for Computational Linguistics, pp. 152–155.

Cimiano, P. & Staab, S. (2004), 'Learning by Googling', *SIGKDD Explorations Newsletter* **6**(2), 24–33.

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

Frasincar, F., Borsje, J. & Levering, L. (2009), 'A Semantic Web-Based Approach for Building Personalized News Services', *International Journal of E-Business Research* **5**(3), 35–53.

Frasincar, F., Milea, V. & Kaymak, U. (2010), *Semantic Web Information Management: A Model-Based Perspective*, Springer, chapter tOWL: Integrating Time in OWL, pp. 225–246.

Golbreich, C. & Imai, A. (2004), Combining SWRL Rules and OWL Ontologies with Protégé OWL Plugin, Jess, and Racer, *in* '7th International Protégé Conference'. Obtained through the Internet: `http://protege.stanford.edu/conference/2004/abstracts/Golbreich.pdf`, [accessed 1/10/2010].

Guarino, N. & Welty, C. A. (2002), 'Evaluating Ontological Decisions with OntoClean', *Communications of the ACM* **45**(2), 61–65.

Harris, S. & Seaborne, A. (2010), 'SPARQL 1.1 Query Language – W3C Working Draft 1 June 2010'. Obtained through the Internet: `http://www.w3.org/TR/sparql11-query/`, [accessed 1/10/2010].

Hearst, M. A. (1992), Automatic Acquisition of Hyponyms from Large Text Corpora, *in* '14th Conference on Computational Linguistics (COLING 1992)', Vol. 2, pp. 539–545.

Hearst, M. A. (1998), *WordNet: An Electronic Lexical Database and Some of its Applications*, MIT Press, chapter Automated Discovery of WordNet Relations, pp. 131–151.

Hepple, M. (2000), Independence and Commitment: Assumptions for Rapid Training and Execution of Rule-Based POS Taggers, *in* '38th Annual Meeting of the Association for Computational Linguistics (ACL 2000)', Association for Computational Linguistics, pp. 278–285.

Horrocks, I., Patel-Schneider, P. F., Boley, H., Tabet, S., Grosof, B. & Dean, M. (2004), 'SWRL: A Semantic Web Rule Language Combining OWL and RuleML – W3C Member Submission 21 May 2004'. Obtained through the Internet: `http://www.w3.org/Submission/SWRL/`, [accessed 1/5/2010].

Java, A., Finin, T. & Nirenburg, S. (2006), Text Understanding Agents and the Semantic Web, *in* '39th Hawaii International Conference on Systems Science (HICSS 2006)', Vol. 3, IEEE Computer Society, p. 62b.

Kalfoglou, Y., Domingue, J., Motta, E., Vargas-Vera, M. & Shum, S. B. (2001), myPlanet: An Ontology-Driven Web-Based Personalised News Service, *in* 'Workshop on Ontologies and Information Sharing (ONTOL2) collocated with 17th International Joint Conference on Artificial Intelligence (IJCAI 2001)', pp. 140–148.

Lösch, U., Rudolph, S., Vrandečić, D. & Studer, R. (2009), Tempus Fugit: Towards an Ontology Update Language, *in* '6th European Semantic Web Conference on the Semantic Web (ESWC 2009)', Vol. 5554 of *Lecture Notes In Computer Science*, Springer, pp. 278–292.

McBride, B. (2002), 'Jena: Semantic Web Toolkit', *IEEE Internet Computing* **6**(6), 55–59.

Micu, A., Mast, L., Milea, V., Frasincar, F. & Kaymak, U. (2008), *Semantic Knowledge Management: an Ontology-based Framework*, IGI Global, chapter Financial News Analysis Using a Semantic Web Approach, pp. 311–328.

Milea, V., Frasincar, F. & Kaymak, U. (2008), Knowledge Engineering in a Temporal Semantic Web Context, *in* '8th International Conference on Web Engineering (ICWE 2008)', IEEE Computer Society Press, pp. 65–74.

Mitchell, M. L. & Mulherin, J. H. (1994), 'The Impact of Public Information on the Stock Market', *Journal of Finance* **49**(3), 923–950.

Motta, E. (1999), *Reusable Components for Knowledge Modelling: Case Studies in Parametric Design Problem Solving*, Vol. 53 of *Frontiers in Artificial Intelligence and Applications*, IOS Press.

Nirenburg, S. & Raskin, V. (2004), *Ontological Semantics*, Language, Speech, and Communications series, MIT Press.

Oberlechner, T. & Hocking, S. (2004), 'Information Sources, News, and Rumors in Financial Markets: Insights into the Foreign Exchange Market', *Journal of Economic Psychology* **25**(3), 407–424.

Prud'hommeaux, E. & Seaborne, A. (2008), 'SPARQL Query Language for RDF – W3C Recommendation 15 January 2008'. Obtained through the Internet: `http://www.w3.org/TR/rdf-sparql-query/`, [accessed 1/5/2010].

Schenk, S. & Gearon, P. (2010), 'SPARQL 1.1 Update – W3C Working Draft 1 June 2010'. Obtained through the Internet: `http://www.w3.org/TR/sparql11-update/`, [accessed 1/10/2010].

Schouten, K., Ruijgrok, P., Borsje, J., Frasincar, F., Levering, L. & Hogenboom, F. (2010), A Semantic Web-Based Approach for Personalizing News , *in* '25th Annual ACM Symposium on Applied Computing (SAC 2010)', ACM, pp. 854–861.

Schutz, A. & Buitelaar, P. (2005), RelExt: A Tool for Relation Extraction in Ontology Extension, *in* '4th International Semantic Web Conference (ISWC 2005)', Vol. 3729 of *Lecture Notes in Computer Science*, Springer, pp. 593–606.

Seaborne, A. (2010), 'ARQ, a SPARQL Processor for Jena'. Obtained through the Internet: `http://jena.sourceforge.net/ARQ/`, [accessed 1/5/2010].

Seaborne, A. & Manjunath, G. (2007), SPARQL/Update: A language for updating RDF graphs, Technical Report HPL–2007–102, Digital Media Systems Laboratory, HP Laboratories Bristol.

Vargas-Vera, M. & Celjuska, D. (2004), Event Recognition on News Stories and Semi-Automatic Population of an Ontology, *in* '3rd IEEE/WIC/ACM International Conference on Web Intelligence (WI 2004)', IEEE Computer Society, pp. 615–618.

Wartena, C. (2010), 'Apolda'. Obtained through the Internet: `http://apolda.sourceforge.net/`, [accessed 1/5/2010].