Personalizing News Services Using Semantic Web Technologies

Flavius Frasincar
Jethro Borsje
Frederik Hogenboom
Econometric Institute, Erasmus University Rotterdam, the Netherlands

ABSTRACT

This chapter describes Hermes, a framework for building personalized news services using Semantic Web technologies. The Hermes framework consists of four phases: classification, which categorizes news items with respect to a domain ontology, knowledge base updating, which keeps the knowledge base up-to-date based on the news information, news querying, which allows the user to search the news with concepts of interest, and results presentation, which shows the news results of the search process. Hermes is supported by a framework implementation, the Hermes News Portal, a tool that enables users to have a personalized access to news items. The Hermes framework and its associated implementation aim at advancing the state-of-the-art of semantic approaches for personalized news services by employing Semantic Web standards, exploiting and keeping up-to-date domain information, using advanced natural language processing techniques (e.g., ontology-based gazetteering, word sense disambiguation, etc.), and supporting time-based queries for expressing the desired news items.

INTRODUCTION

The simplicity, availability, reachability, and reduced exploitation costs have made the Web one of the most common platforms for information publishing and dissemination. This is particularly true for news agencies that use Web technologies to present emerging news regarding different types of events as for example business, cultural, sport, and weather events. Most of this information is published as unstructured text that is made available to a general audience by means of Web pages.

The heterogeneity of the Web audience and the diversity of the published information asks for more refined ways of delivering information that would enable users to access news items that interest them. For this purpose the Really Simple Syndication (RSS) (Winer, 2003) standard was developed that publishes information in a semi-structured format that supports machine processing. This format is based on metadata that (1) associates news items with channels (feeds) that have properties like categories (e.g., business, sport, politics, etc.), title, publication date, etc., and (2) describes news items by means of their properties as categories (e.g., online business, business system, Internet marketing, etc.), release time, title, abstract, link to the original published information, etc.

Most of the annotations supported by the RSS feeds are coarse-grained providing general news information. Fine-grained information, as for example the financial events depicted in news, is at the moment not available. Also, the current annotations are only partially processable by machines as the tags do not have unique semantic meaning associated to them and thus have different interpretations. Being able to understand the semantic content of a news item would enable a fine-grained categorization of this information, thus better supporting the users (casual users, media analysts, stock brokers, etc.) information needs.

In order to make the Web data not only machine readable but also machine understandable the World Wide Web Consortium proposes the Semantic Web (Berners-Lee, Hendler, & Lassila, 2001), a sequence of technologies that allow for self-describing content. On the Semantic Web metadata is defined using semantic information usually captured in ontologies. Some of the most popular formats to describe ontologies on the Semantic Web are RDF(S) (Klyne & Carroll, 2004) (Brickley & Guha, 2004) and OWL (Bechhofer et al., 2004).

A special class of users who make daily use of (emerging) news is that of stock brokers. Because news messages may have a strong impact on stock prices, stock brokers need to monitor these messages carefully. Due to the large amounts of news information published on a daily basis, the manual task of retrieving the most interesting news items with respect to a given portfolio is a challenging one. Existing approaches such as Google Finance or Yahoo! Finance are developed to meet these personalization needs by supporting automatic news filtering on the Web.

Current approaches to news filtering are able to retrieve only the news that explicitly mention the companies involved, failing to deliver indirect information which is also deemed relevant for the considered portfolio. For example, for a portfolio based on Google shares, such systems fail to deliver news items related to competitors of Google, such as Yahoo! or Microsoft, which might have an indirect influence on the share price of Google. Exploiting the semantic contextual information related to companies such as its competitors, CEO's, alliances, products, etc., enables a more comprehensive overview of relevant news with respect to a certain portfolio.

Existing news filtering systems are not able to cope with delivering news items satisfying temporal constraints. The time aspect is of utmost importance when, for example, one considers the fact that news items usually have an immediate impact on stock prices, or when one desires to do a historical analysis of past news and stock price evolutions. Being able to exploit the timestamps associated to news items enables retrieving only news that obey user-determined time-related constraints.

Another limitation of existing news personalization services is that they fail to consider the dynamicity of the current world. In an economic context companies are created, disappear, or are bought by other companies, CEO's change positions or sometimes even companies, companies develop new products while other products become obsolete, etc. For a proper representation of the domain it is important that we cope with these changes and allow our world representations to dynamically change. It is only in this case that the user will be able to properly specify their interests and receive up-to-date news information that matches these interests.

In this chapter we propose the Hermes framework, a semantics-based approach for retrieving news items related, directly or indirectly, to concepts of interest from a domain ontology. In addition these news items might need to satisfy temporal constraints. For illustration purposes we focus here on the NASDAQ stock market domain (Kandel & Marx, 1997), but the genericity of our approach makes it applicable also to other domains, as, e.g., tourism or scientific domains. The Hermes News Portal (HNP) is an implementation of the Hermes framework, which allows the user to specify queries for the concepts of interest and temporal constraints, and retrieve the corresponding news items.

For HNP we make use of Semantic Web technologies like OWL (Bechhofer et al., 2004) for formally defining the semantics of the concepts of interest in the ontology. We employ natural language processing (NLP) technologies as, e.g., lexical analysis, gazetteering, word sense disambiguation, etc., for

indexing news items based on ontology terms. The most popular Semantic Web query language SPARQL (Prud'hommeaux & Seaborne, 2008) is used for expressing queries using the previously identified concepts. In order to simplify the representation of temporal constraints we propose time-related extensions to SPARQL. HNP is a generic platform that could easily be applied to other domains than the financial one.

Compared to our pervious work on Hermes (Borsje, Levering, & Frasincar, 2008) (Frasincar, Borsje, & Levering, 2009) (Schouten et al., 2010), this chapter provides a complete description of Hermes as available at the current moment. It also enhances our previous presentations on the Hermes framework with additional details on the knowledge base updating and results presentation steps (e.g., refinement of the information extraction patterns, definition of concept importance in a news item with respect to a query, .etc.). Based on our latest insights in designing the framework, this chapter suggests additional future work that has not been previously identified.

The structure of the chapter is defined as follows. The first section discusses related approaches for personalized news services. The second section presents the Hermes framework identifying the proposed methodological steps. The third section describes HNP, an implementation of the proposed framework. The fourth section discusses future research directions. The last section concludes the chapter.

BACKGROUND

Among the methods that aim at personalizing news information we distinguish two types: non-semantic approaches and semantic approaches. In the followings we will present short descriptions of three non-semantic methods: SeAN, YourNews, and NewsDude, and three semantic methods: MyPlanet, SemNews, and QuickStep. For each presented method we give the differences compared to our approach and at the end of this section we highlight the main contributions of the Hermes framework.

Server for Adaptive News (SeAN) (Ardissono, Console, & Torre, 2001) enables a personalized access to news servers on the Web. The generated views are composed of sections, as in newspapers, on which customized news items are embedded. The news items are viewed as complex entities in which attributes define different components, e.g., title, abstract, text, photos, videos, commentaries, etc. The system employs a user model initialized using orthogonal stereotypes (interests, domain expertise, cognitive characteristics, and life styles) for which the user is asked to provide input and is further updated using rules that exploit the user behavior with the application. Taking into account the user model, the system builds a presentation based on relevant news items, each news item being shown at an appropriate level-of-detail (based on the user model). Differently than SeAN, our framework uses standard Semantic Web technologies for representing knowledge and employs NLP techniques for automatic annotation of news items, instead of using a manual approach.

YourNews (Ahn, Brusilovsky, Grady, He, & Syn, 2007) proposes an open and editable user model for personalizing news items. The user model is open in the sense that users can view the list of keywords stored in the individual profiles. The user model is also editable as it allows users to add/delete keywords from their associated profiles. As an additional feature which also contributes to the transparency and control over adaptation, YourNews shows the key terms present in news items. The representation of news items is given by weighted vectors of terms (Salton, 1971), where the weights are computed using TF-IDF (Salton & McGill, 1983). The visited news items are used for building a weighted term vector which is the user model. The similarity between a news item and the user model is defined by the cosine metric between their associated vectors. This measure allows the system to

recommend news items that are considered relevant for the user. Despite the users' interest to view and edit their profiles, there is a decrease in performance (e.g., precision, recall, etc.) for recommended items compared to the same system using a closed user model (where the user is not able to view/edit the user model). While YourNews uses a keyword-based approach for modeling news items and user interests, Hermes employs a semantic approach based on ontology concepts for modeling similar aspects.

NewsDude (Billsus & Pazzani, 1999) is a personal news agent. It uses a two step solution for personalization, at the beginning it employs the user's short term interests to discover relevant news items, and, if the result set is empty result, it filters news based on the user's long term interests. For the short term model construction, NewsDude employs TF-IDF in combination with Nearest Neighbour (NN), which is able to represent user's multiple interests and takes in consideration the changing user's short term interests. Long term interests or user's general interests are modeled using a Naïve Bayes classifier. Compared to NewsDude, Hermes uses a semantic instead of lexical representation of the available data, and an unsupervised classification algorithm instead of a supervised one, eliminating thus the need of a training step.

MyPlanet (Kalfoglou, Domingue, Motta, Vargas-Vera, & Shum, 2001) aims at providing users with news items relevant for their topics of interest. MyPanet is an extension of PlanetOnto, an integrated suite of tools used to create, deliver, and query internal newsletters of the Knowledge Media Institute (KMi). Similar to our approach an ontology is used for classifying news items and allowing the user select his topics of interest. Nevertheless, the classification process is based on the heuristics of cue phrases attached to ontology concepts, while we have a more systematic approach to classification by employing NLP techniques (e.g., exploiting the WordNet term synonyms, performing word sense disambiguation, etc.) that improve classification results. Also, we support semi-automatic ontology updates, a feature missing from MyPlanet. In addition, our implementation is based on the standard ontology language OWL instead of the specific ontology language, OCML (Motta, 1999), used in myPlanet. We also did choose to present the ontology as a graph instead of a tree as it allows the user to have a more comprehensive overview of the ontology structure.

SemNews (Java, Finin, & Nirenburg, 2006) proposes a framework for understanding and querying news items. As in Hermes, the monitored news are made available by RSS feeds. The news items are analyzed by OntoSem (Nirenburg & Raskin, 2001), SemNews' natural language processing engine. OntoSem converts the textual representation of news into Text Meaning Representation (TMR), a specific format for knowledge representation. The TMR descriptions are subsequently converted to OWL and published on the Web. The OWL news representation can be used for querying using RDQL (Seaborne, 2004), one of the precursors of the SPARQL query language. Differently than SemNews, Hermes uses a semantic lexicon (e.g., WordNet) for performing word sense disambiguation, and allows for a more intuitive way of building queries by letting the user make his selections in a graphical way.

Quickstep (Middleton, Shadbolt, & Roure, 2004) is a recommender system for academic papers. Academic papers are classified by means of the boosted IBk classifier (Aha, Kibler, & Albert, 1991) which employs the k-Nearest Neighbour (k-NN) algorithm applied on a vector space encoding of papers. Paper topics are represented in an ontology and are also used for describing user interests. In addition this ontology is used to solve the user or item cold-start problems, by setting the initial user interests using the topics of authors' previous papers. The recommendation is based on the similarity between the user's topic of interest and the paper's topics. Differently than Quickstep, which considers only type relationships, Hermes takes in account also other ontology relationships (e.g., part-of, domain specific

relationships, etc.) in the personalization process. Also, we used an unsupervised method for classification instead of supervised one, thus not requiring a training phase.

The contributions that Hermes brings to building personalized news services compared to existing approaches are sixfold. First, Hermes makes a strict distinction between the framework (Hermes framework) and its implementation (HNP), allowing for possible different technologies (as these evolve) to be used with the same framework. Second, Hermes uses an advanced NLP methodology (e.g., tokenization, part-of-speech tagging, word sense disambiguation, etc.) for news understanding employing a semantic lexicon (e.g., WordNet). Third, the implementation is based on the most up-to-date Semantic Web standards (OWL and SPARQL). Fourth, we allow news querying using temporal constraints by providing temporal extensions to SPARQL. Fifth, the user is provided with a graphical query interface to specify the concepts of interest in a direct (using concept selections) or indirect manner (using relationship selections). Sixth, and the last contribution of Hermes, is the process of updating the domain knowledge based on real-time world changes (as given in news), a feature that is missing from the investigated ontology-based approaches for news personalization.

HERMES FRAMEWORK

The Hermes framework proposes a sequence of steps to be followed in order to build a personalized news service. The input for the constructed system comprises RSS news feeds and the output consists of news items fulfilling user needs. The Hermes framework is centered around a domain ontology which is used for indexing news items and helping the user formulate queries based on his concepts of interest. In addition, the user can specify temporal constraints that news items need to satisfy. The resulting news items are sorted based on their relevance for the user queries.

For illustrative purposes we chose a financial domain example, i.e., a personalized news service which can help the stock brokers in their daily decisions. More precisely we opted for portfolios based on stocks of NASDAQ companies. For this purpose we developed a domain ontology, which captures companies, products, competitors, CEO's, etc. In addition we have developed a news ontology able to store news items and their metadata such as title, abstract, time stamp, etc.

The domain ontologies are developed by domain experts (in the example these are the financial domain experts). 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. As the news information can contain additional concepts not known a priori, the ontology needs to be regularly maintained by the domain experts. We validated our domain ontology using the OntoClean methodology (Guarino & Welty, 2002).

As news items might come from different RSS feeds, it is possible that the same news item has been published through different channels. After aggregating the news items one needs to remove the duplicate ones. In order to speed-up this process we employed the heuristics to use only the title for identifying news items, even so we acknowledge that in few cases news items might have the same title and still have different content, i.e., they represent different news items.

The architecture of the Hermes framework is described in Figure 1 and is composed of four main steps: *news classification*, *knowledge base updating*, *news querying*, and *results presentation*. *News classification* is responsible for indexing the news items based on ontology concepts. *Knowledge base updating* keeps the knowledge base (ontology instance) up-to-date. *News querying* consists of two substeps: *query formulation*, i.e., helping the user build the query that expresses the items of interest,

and *query execution*, i.e., computing the results of query evaluation. In the last step, *results presentation* the computed news items are presented based on their relevance to the user interests.

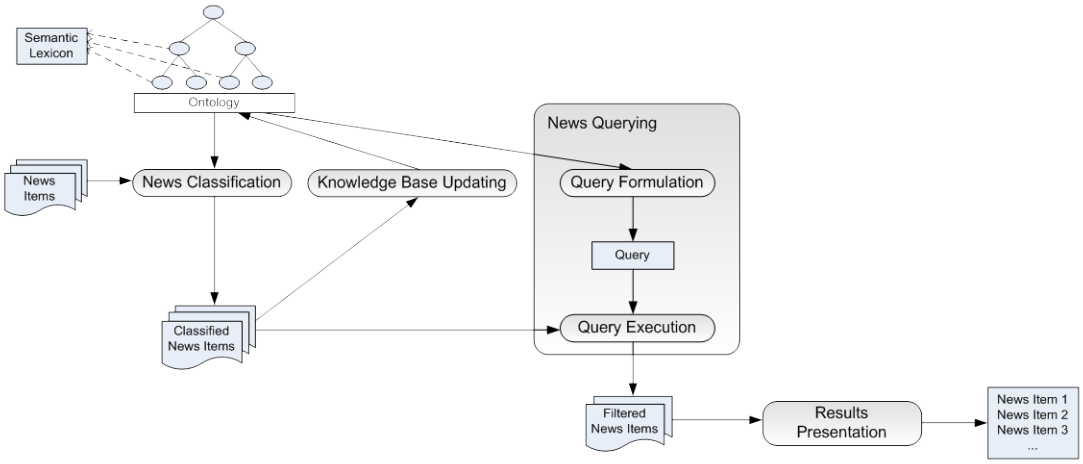

Figure 1 Hermes architecture

News Classification

The ontology concepts used for news items classification are classes and individuals from the domain ontology. Concepts are linked to synsets, i.e., sets of synonyms, from a semantic lexicon, which identify their unique meaning. The synonyms stored in a synset in the semantic lexicon are used as lexical representations of the associated concept. In this case a lexical representation has a sense associated with it, i.e., the one given by the corresponding synset. As the ontology is specific to a domain, while the semantic lexicon is domain independent, we associate additional domain specific lexical representations to the concepts in our ontology. The lexical representations are composed only of word lemmas (the canonical word form appearing in dictionaries).

In addition, for classes without subclasses and individuals we decided to consider the hyponyms associated to their corresponding semantic lexicon synset. For these classes and individuals, the domain expert, who devised the ontology, is probably not interested in more refined definitions of these. However, the lexical representations of these concepts can be enlarged by considering also the corresponding hyponyms synsets from the semantic lexicon.

The classification approach is *ontology-centric*, in the sense that the ontology concepts are loaded one at-a-time and their lexical representations are matched against the news items. This approach is opposed to a *news items-centric* one where the words in the news items are matched against the lexical representations of the concepts from the ontology. We opted for an ontology-centric approach in order to speed-up the classification process as in this case we need to traverse the ontology only once. The number of concepts in the ontology is considerably larger than the number of words in the news items (for batch processing of news items).

First the tokenization, sentence splitting, part-of-speech tagging, and morphological analysis are performed. The tokenization precedes sentence splitting as sentence splitting needs the identification of the punctuation signs from tokenization (as ".", ",", etc.). Morphological analysis follows part-of-speech tagging because the lemma of a word depends on its part-of-speech tag. For example "reading" as a verb has lemma "read" but as noun it has the lemma "reading". In this way, all words in a news item are reduced to their canonical form, a form shared also by the lexical representations of concepts stored in

the domain ontology. For lexical representation identification we use the maximal group of words (compound words) found in sequence in a news item that stand for a concept's lexical representation. For example "European Central Bank" would be identified as a compound word representing the ECB concept, e.g., a longer match supersedes a shorter match.

Each time a lexical representation of a concept is matched a *word sense disambiguation* procedure takes place. As the same lexical representation can belong to different concepts (present or not in the ontology) this procedure checks if the match corresponds to the meaning of the found concept. If the check is positive a *hit* is stored in the ontology, i.e., a link between the news item and the corresponding concept is defined. The hit also stores the found lexical representation, as classification evidence.

For word sense disambiguation we use a variant of the SSI algorithm (Navigli & Velardi, 2005). In this process we also consider lexical representations for concepts that are not stored in the ontology but are present in the semantic lexicon as these are relevant when computing the sense of a found lexical representation from the ontology. These lexical representations help in better determining the context of a sentence and thus computing the sense of an ontology-based lexical representation.

The algorithm determines, per news sentence, the sense of a lexical representation (*lex*) by computing the sum of the distances between one of the senses of the considered lexical representation (s_j) and the senses of the previously disambiguated lexical representations from the context sentence (sc_i). The sense corresponding to the smallest sum is the chosen one (*selectedSense*). The algorithm starts with monosemous lexical representations (i.e., lexical representations which correspond to only one concept) and in case that such representations do not exist a guess is made by picking the most common sense for one of the found lexical representations. These senses are added to the context (*I*) of the sentence. For each remaining polysemous lexical representations two steps take place: a disambiguation step to find the correct sense and an insertion step which adds the newly computed sense to the context. In this way the context gets enlarged with new senses that model the meaning of the sentence. Formula (1) specifies what sense is selected for each lexical representation.

$$selectedSense(lex) = \arg \min_{s_j \in senses(lex)} \sum_{sc_i \in I} d(s_j, sc_i) \qquad (1)$$

The distance between senses is defined as the length of the shortest path between the corresponding synsets ($sense_i$ and $sense_j$) in the semantic lexicon graph. The graph is based on the hypo/hypernyms relationships stored in the semantic lexicon. Because nouns and verbs have different hierarchies for these relationships, we apply the algorithm separately for these two part-of-speeches. Path lengths larger than a predefined constant (e.g., 4 or 5) are not used (for these cases the distance is considered infinite). In this way we employ only semantically close concepts that truly help in the disambiguation procedure and also improve the speed of the process by not using arbitrarily large paths. Formula (2) shows how to compute the distance between two senses.

$$d(sense_i, sense_j) = length(shortestPath(synset(sense_i), synset(sense_j))) \qquad (2)$$

As the distances between synsets are not changing, one can pre-compute these, thus further reducing the time needed for the disambiguation step. The Hermes framework can be used with other methods for computing the similarity between concepts as for example string metrics (e.g., Levenshtein, Editex, etc.) or lexical co-occurrences in a corpus (e.g., pairwise mutual information, Google distance, etc.).

Nevertheless, many of these methods are less precise than the graph-based method used here, as they do not take in account senses, comparing only the lexical representations of concepts.

Knowledge Base Updating

As the real world is continuously changing it is important for our framework to keep its internal world representation up-to-date. For this purpose we employ the information contained in news items as a source of possible changes in the real-world. In the Hermes framework news play a dual role (1) they are used to provide the user with the desired information, and (2) they are employed as a source of new information to be used for updating the knowledge base. Regarding updates, we recognize concepts known in the knowledge base in news items and detect changes related to these concepts that happen in the real-world (e.g., concepts can disappear, concept relationships might change, etc.).

In the Hermes framework the world changes are triggered by *events* that are modeled in the ontology and are detected in news items. Events have associated alternative lexico-semantic patterns for their detection, and a sequence of actions to be used for updating the knowledge base. Note that one event type (e.g., `kb:newCEO`, which represents the appointment of a new CEO) can correspond to many event instances detected in news items each triggering the associated list of actions. The l*exico-semantic patterns* associated to an event combine lexical arguments with ontology concepts (which have attached lexical representations). The list of actions associated to an event represent an ordered set of actions each being an insert or delete operation to be performed on the knowledge base. The lexico-semantic patterns communicate with the actions by means of variables bounded to ontology concepts during information extraction (event rules patterns execution). The events and their associated information (lexico-semantic patterns and actions) are maintained by the domain experts.

The knowledge base updating step is composed of five substeps: event rules patterns construction, i.e., building the alternative lexico-semantic patterns associated to events, event rules actions construction, i.e., building of action lists associated to events, event rules patterns execution, i.e., extraction of the event-related information (event instances) from news, event validation, i.e., allowing the domain expert to validate extracted event instances before updates, and event rules actions execution, i.e., updating the knowledge base with the event information.

Event Rules Patterns Construction

Each event (type) has a set of lexico-semantic patterns (alternatives) that can trigger the detection of an event instance in a news item. The lexico-semantic patterns are based on triples (subject, predicate, object), each triple element being possibly annotated with variables. As triple elements one can use a knowledge base instance (concepts that represent instances) or a knowledge base concept (concepts that represent types).

We differentiate between types and instances by enclosing types in the square brackets ("`[`" and "`]`"). A type is a placeholder for all lexical representations of all instances of this type, while a concept instance stands only for the lexical representations associated to this particular instance. For example the type `[kb:Company]` represents all lexical representations of all company instances, i.e., "IBM", "International Business Machines", "EBay", "E-bay", "Ebay", etc., while the instance `kb:IBM` represents all lexical representations of IBM, i.e., "IBM", "International Business Machines", etc. Variables are represented by names prefixed with the "$" symbol. In addition to concepts and variables, lexico-semantic patterns can contain lexical presentations enclosed in quotes as for example "is ruined".

There are two types of lexico-semantic patterns: SP patterns, i.e., patterns that have only a subject and a predicate, and SPO patterns, i.e., patterns that have a subject, a predicate, and an object. An example of an SP pattern associated to the `kb:Bankruptcy` event is:

```
$c:[kb:Company] kb:GoesBankrupt
```

which possibly matches "WorldCom goes bankrupt", "WorldCom filed for Chapter 11", "WorldCom is ruined", etc. An example of an SPO pattern associated to the `kb:newCEO` event is:

```
$p:[kb:Person] kb:BecomesCEO $c:[kb:Company]
```

which possibly matches "Steve Ballmer appointed CEO of Microsoft", "Steve Ballmer becomes news Chief Executive Officer of Microsoft", "Steve Ballmer is the new CEO of Microsoft", etc. As can be noted from these examples, predicates, as instances of the property meta-class, are also concepts and thus have lexical representations associated to them.

Event Rules Actions Construction

Each event (type) has a list of update actions (ordered set) that can fire when an event instance is found in a news item and has been validated by the user. These actions are used to update the knowledge base by deleting or adding concept relationships. There are two types of update actions: insert triples, i.e., add new triples to the knowledge base, and delete triples, i.e., remove triples from the knowledge base. For the `kb:newCEO` event the action list contains two actions given in the following order:

```
DELETE $c kb:hasCEO *
INSERT $c kb:hasCEO $p
```

In the actions list, the order of actions is important as it gives the execution order and the update results are sensitive to this order. Performing last a delete action would remove the newly inserted CEO leaving the company with the CEO undefined. The symbol '*' is a placeholder which stands for any concept (in the knowledge base). It is used to delete triples for which some of the components are not known in the query creation process.

Event Rules Patterns Execution

The event rules patterns execution is used for extracting information, more precisely events, from the news textual information. Each match represents an event instance. Variables are bound to the matched ontology concepts in news and are used in the event rules actions execution for possibly inserting the event instance in the knowledge base but more importantly they are used to implement the world state changes triggered by the found events.

Using the previous example, once the `kb:newCEO` event instance is found (triggered by the found relation `kb:BecomesCEO`), the variable `$p` is instantiated with the found `kb:Person` and the variable `$c` is instantiated with the found `kb:Company`. These variables are used in the event rules action execution to state that the found person is the new CEO of the found company. Note that for each found `kb:newCEO` event, the previous variables have different values unless the same event instance has been found multiple times.

Event Validation

As the event detection is not flawless, these events need to be validated by the domain expert. The validation step is necessary in order not to pollute the knowledge information with incorrect information. Polluting the ontology will have a negative impact in the next information extraction run. It is the domain expert who can acknowledge if the event has been properly extracted and if it represents a fact or an opinion (only facts are valid events).

The manual event validation process makes our framework semi-automatic, all the other steps besides this one being fully automatic. We assume that the domain expert has configured the system with the right input information (the domain ontology and the RSS feeds). It is the domain expert, and not the user, who controls the internal data of the system and thus is responsible for any changes made to the domain ontology.

Event Rules Actions Execution

After an event instance has been discovered, as a result of its validation, the Hermes framework will update the ontology with the event-triggered new information. The event instances are processed in the order they are found in the news items, and for each event instance the associated list of actions is executed in the specified order. In the previous example it is important to first delete the old CEO from the knowledge base before inserting the new CEO. Swapping these two actions would make a company not have a CEO defined in the ontology.

By processing the events in the order they are found (chronological order) we make sure that the newest events are processed last. In this way, in case of conflicting updates, we will retain the most up-to-date ones in the knowledge base. After executing the actions in the proper order we have the event effects captured in the knowledge base. With an updated knowledge base the system is ready to classify the new news items as well as offer concepts for queries that better reflect reality.

News Querying

The user expresses the topics of interests by posing queries using concepts from the domain ontology. In addition the user can express constraints that the timestamps associated to news items need to satisfy. The news querying step consists of two substeps: query formulation, i.e., supporting the user to build queries, and query execution, i.e., computing the results of query evaluation.

Query Formulation

In order to assist the query construction process we present to the user the ontology graph. We decided to show a graph-based representation of the ontology instead of a tree-based representation, as it gives more insight in the overall structure of our domain. For example a graph representation captures more relationship types instead of a singular relationship type, often the subsumption relationship, from a tree-based representation. The user needs to understand the different relationship types in order to be able to build his query.

By using the ontology graph the user can select the *direct concepts of interest*. For each concept of interest, the framework considers all its lexical representations, which the user does not need to remember during the query specifications. In addition he is able to specify concepts of interest using a keyword search facility through the graph. For this purpose the input keyword is checked for possible inclusion in the lexical representations of concepts. If such inclusions are found the corresponding concepts are being returned as possible direct concepts of interest. It is the task of the user to accept these as direct concepts of interest or to reject them.

One of the important functionalities of the Hermes approach is that it allows for the selection of concepts indirectly linked to the selected ones, concepts which are not a priori known to the user. We call these concepts *indirect concepts of interest*. For the selection of the indirect concepts of interest the user can state the type of the relationships that links the direct concepts of interest to the indirect concepts of interests or leave this undefined in which case all relationship types are being considered.

Suppose that the user has selected the direct concept of interest `Google` from the NASDAQ domain ontology. The user also specifies that he is interested in news related to the competitors of Google by selecting the `kb:hasCompetitor` relationship. That means that `Yahoo!`, `Microsoft`, and `EBay` will be selected as indirect concepts of interests, without the user having to know the exact names of Google's competitors. All this background information is being extracted from the ontology.

The direct, indirect, and keyword-based search concepts of interest, as well as the other concepts from the ontology need to be emphasized in a graph by using for example different colors. In this way the user is able to know, by analyzing the graph, why a certain concept is being highlighted. As the size of the graphs can be very large the user is provided with zooming/panning facilities for visualizing the ontology.

The original graph of the domain ontology is also called the *conceptual graph*. Based on the user selection, a new graph is generated, the so-called *search graph* that contains only the concepts and concept relationships relevant for the query. The user can go back and forth between the two graphs performing new selections and thus updating the search graph with new concepts.

The search graph is given by the subgraph of the conceptual graph that models the user interests. It contains all the user concepts of interests (directly or indirectly selected by the user) and the relationships among them. The search graph has disjunctive semantics with respect to the included concepts which means that the user is interested in *any* of the search graph concepts to be found in news items.

Another crucial functionality of the Hermes approach is that it allows the specification of temporal constraints for news items. As news items appear at a certain moment in time and have certain time validity, it is important to be able to restrict the timestamps associated with the news. For this purpose the user can employ time comparison/arithmetic operators and retrieve the current time in order to build complex time expressions. In addition the system provides predefined temporal constraints such as: last day, last week, last two weeks, last three months, last quarter, last half year, and last year. The temporal conditions that model these constraints have conjunctive semantics as they need to be fulfilled in the same time.

Query Execution

Based on the previously selected concepts and specified temporal constraints the system can support the generation of the corresponding query in a semantic query language. This translation process involves mapping concepts and temporal constraints to query restrictions. After that, the user can trigger the query evaluation and the relevant news items are retrieved. The order of the retrieved news items is not relevant, at this stage.

Results Presentation

The results presentation is responsible for displaying the news items that match the user query. The results presentation is composed of two substeps: news sorting, i.e., sorting the news based on their relevance to the user query, and news presentation, i.e., displaying the news items in a visual appealing manner and explaining their relevance to the user query.

News Sorting

The results returned from query evaluation are presented in the order of their relevance for the user query. For this purpose, for each returned news item a *relevance degree* is computed based on all the hits between the news item and the query concepts. News items with a high relevance degree are placed at the top of the retrieved news items list.

Based on previous work (Micu, Mast, Milea, Frasincar, & Kaymak, 2008) the relevance degree is defined as a weighted sum of the number of hits ($n(c_i)$), where the weights (w) depend on hits location (*title* or *body* of a news item). From our experimental results we have determined as acceptable values for w_{title} to be *2* and for w_{body} to be *1*. Formula (3) presents how to compute the relevance degree.

$$relevanceDegree(news\ item) = \sum_{\substack{c_i\ found\ in\ title \\ c_i \in Q \cap news\ item}} w_{title}\ n_{title}(c_i) + \sum_{\substack{c_i\ found\ in\ body \\ c_i \in Q \cap news\ item}} w_{body}\ n_{body}(c_i) \qquad (3)$$

News items that have the same relevance degree are sorted in descending order based on the associated timestamps (the most recent news items are presented first).

News Presentation

For each news item a summary is given containing the title, source, date, and few lines from the news item. Also, the icon of the most important query concept found in each news item is displayed. Formula (4) shows how to compute the *importance degree* for each query concept c_i found in a news item:

$$importanceDegree(c_i, news\ item) = n_{title}(c_i) + n_{body}(c_i) \qquad (4)$$

The most important query concept in a news item is the one with the highest importance degree. If several query concepts have the same importance degree in a news item, an arbitrary query concept from these ones is chosen as the most important concept.

In addition to presenting the relevant news items, the system shows the query concepts in order to provide cues of the current query for which the results are computed. Also, for each returned news item, the found lexical representations stored in the hits are emphasized in the news item text, thereby offering to the user an explanation of why a certain news item is considered to be relevant.

HERMES NEWS PORTAL

The Hermes News Portal (HNP) is an implementation of the Hermes framework, which allows the user to specify queries on the considered domain using temporal constraints and subsequently retrieve the relevant news items. The presentation of HNP follows closely the steps proposed by the Hermes framework. Note that HNP is one of the possible implementations of the Hermes framework, with specific design choices, query/programming languages, and libraries used.

Operating on the Semantic Web we chose as ontology language OWL due to its expressivity and standard status. We did not opt for RDFS because OWL specific features, as for example relationship

inverses (`hasCompetitor` has as inverse `isCompetitorOf`), are exploited in the conceptual graph. Lacking a true OWL query language we used SPARQL (Prud'hommeaux & Seaborne, 2008) as the query language, an RDF query language that we extended with time-related functionality. This functionality is provided by implementing comparison/arithmetic time operators and functions for retrieving current time information.

The chosen implementation language is Java due to the availability of powerful libraries for manipulating, reasoning with, querying, and visualizing OWL ontologies. For manipulating and reasoning with OWL ontologies we used Jena (Jena Development Team, 2010a). For querying we employed ARQ (Jena Development Team, 2010b), the SPARQL implementation available in Jena. For updating the ontology we have used SPARQL/Update (Seaborne et al., 2008) also supported by ARQ. For visualizing ontologies we adapted the generic graph visualization library Prefuse (The Berkeley Institute of Design, 2010) for visualizing OWL graphs (Borsje & Giles, 2010). For part-of-speech tagging we used the Stanford parser (The Stanford Natural Language Processing Group, 2010). As a semantic lexicon we employed WordNet (Princeton Cognitive Science Laboratory, 2010), the largest database available online for the English language. JWI (Finlayson, 2008) was used for the morphological analysis (finding lemmas of words) and the communication with WordNet. As natural language processing pipeline (e.g., tokenization, sentence splitting, gazetteering, etc.) we have used the A Nearly New Information Extraction System (ANNIE) pipeline from General Architecture for Text Engineering (GATE) (Cunningham, Maynard, Bontcheva, & Tablan, 2002) and for information extraction from news items we have used the Java Annotations Patterns Engine (JAPE) language and its implementation, also part of GATE.

We illustrate the HNP by means of the following user query: *retrieve all news items related to Google or one of its competitors that appeared in the last three months*. For this query we will go through all the different phases of Hermes: news classification, knowledge base updating, news querying, and results presentation. In the current HNP the news items duplicates removal has not been yet implemented.

News Classification

The news classification step is responsible for indexing news items based on the domain ontology. We present this process by means of the news item example depicted in
Figure 2. The first line describes the title, the second line shows the timestamp, and the remaining text represents the content of the news item.

> *Google to broker print ads in newspapers*
> 6 November 2006 17:41 CET
>
> SAN FRANCISCO (Reuters) - Google Inc. is set to begin helping customers buy advertisements in 50 U.S. newspapers in a test of how the Web search leader can extend its business into offline media, the company said on Sunday.

Figure 2 News item example

The news classification step starts by identifying lexical representations of the ontology concepts in the news item. First the tokenization, sentence splitting, part-of-speech tagging, and morphological analysis take place. Then, the concepts from the ontology are traversed once and their lexical representations are matched against the content of the news item. The following lexical representations “Google”, “extend”, and “company” are found. Next, per sentence, for each of the lexical representations with multiple senses, the word sense disambiguation procedure takes place in order to identify the used

senses. The noun “Google”, having only one sense, does not undergo the word sense disambiguation procedure and is mapped to the `Google` concept.

For “extend” and “company” a word sense disambiguation procedure is needed. In this process we do consider also lexical representations of concepts outside the ontology, as for example the nouns “customer” and “business”, or the verb “buy” that do appear in the news item. We select the sense that yields the smallest sum of similarities to previously disambiguated lexical representations. “extend” is determined as representing the `extend-verb-#1` concept with lemma `extend`, part-of-speech tag `verb`, and the first sense from WordNet. For “company” the corresponding concept is `company-noun-#1`. “customer”, a lexical representation outside the ontology, is determined as having the sense `customer-noun-#1`.

After identifying an ontology concept in a news item, a hit is stored. This hit is defined as a link between the news item and the concept together with the found lexical representation. For this purpose we decided to model a hit as an instance of the `Relation` class, which uses different properties for storing the involved news item, concept, and found lexical representation. This modeling choice is based on a best practice for modeling N-ary relations on the Semantic Web (Noy & Rector, 2006). A news item can have multiple hits associated, enabling thus its multiple classification.

Knowledge Base Updating

The knowledge base updating step makes sure that the knowledge base has an accurate representation of the real-world in HNP. It is composed of five substeps: event rules patterns construction, event rules actions construction, event rules patterns execution, event validation, and event rules actions execution, discussed below. In order to support the domain expert for events definition we have developed an event editor in the HNP.

Event Rules Patterns Construction

In the first step, event rules patterns constructions, the lexico-semantic patterns associated to event types are defined. These lexico-semantic patterns are used to discover event instances in news items. The lexico-semantic patterns have a subject, a relation, and an optional object. The subjects, relations, and objects are concepts from the knowledge base (instances or types), and are interpreted as placeholders for all the associated lexical representations (of instances or of all instances associated to a certain type).

Figure 3 shows the definition of a lexico-semantic pattern in the event editor. In this example the `rb:buy-rule` (the prefix `rb` stands for rule base, to differentiate from the prefix `kb`, which stands for knowledge base) is used to detect a buy event. The subject is of type `[kb:Company]`, the object is of type `[kb:Company]`, and the predicate is `kb:Buys`. At the current moment, HNP does not use a different notation for types than for individuals, as we have specified in the framework, assuming that when the user employs a type concept, it is meant a type and not an instance (i.e., instance of the meta-concept class). In the future we plan to update the HNP with the latest framework features, including the most recent pattern language developments.

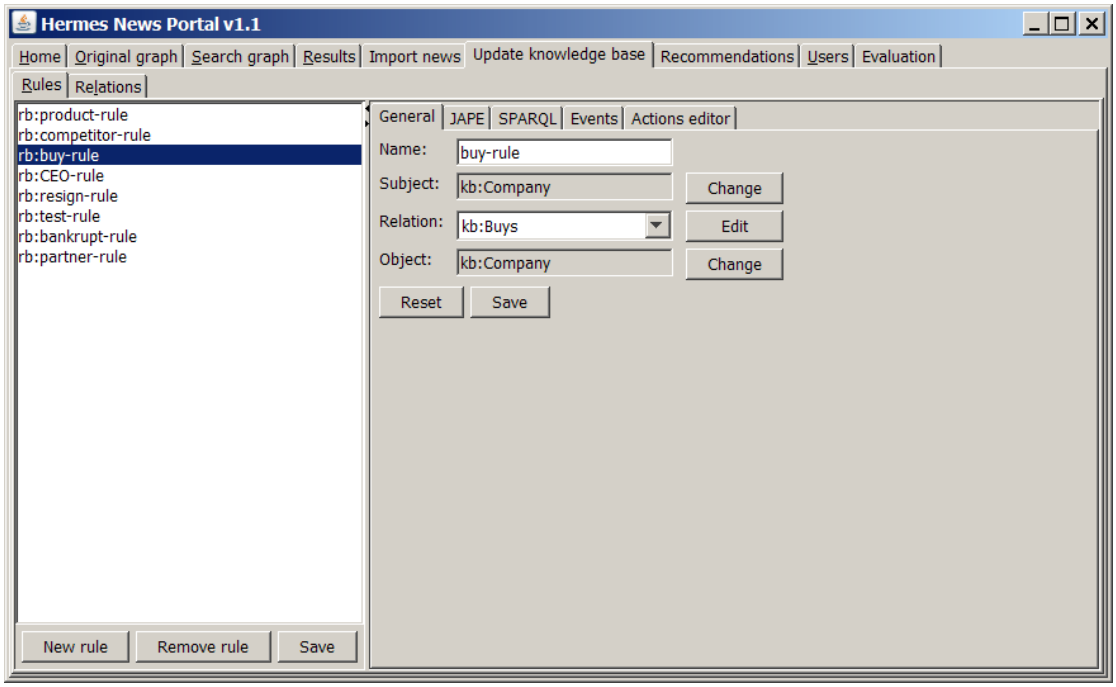

Figure 3 The event editor: lexico-semantic patterns example

The subject and the object can be changed by selecting new concepts. Figure 4 shows the selection of concepts. These concepts are selected from the graph representation of the knowledge base. The lexical representations associated to these concepts are defined in the knowledge base. The current version of HNP allows the definition of relations and their lexical representations. For example, Figure 5 shows the `kb:Buys` relation and its associated lexical representations: “purchase”, “acquire”, “acquires”, etc.

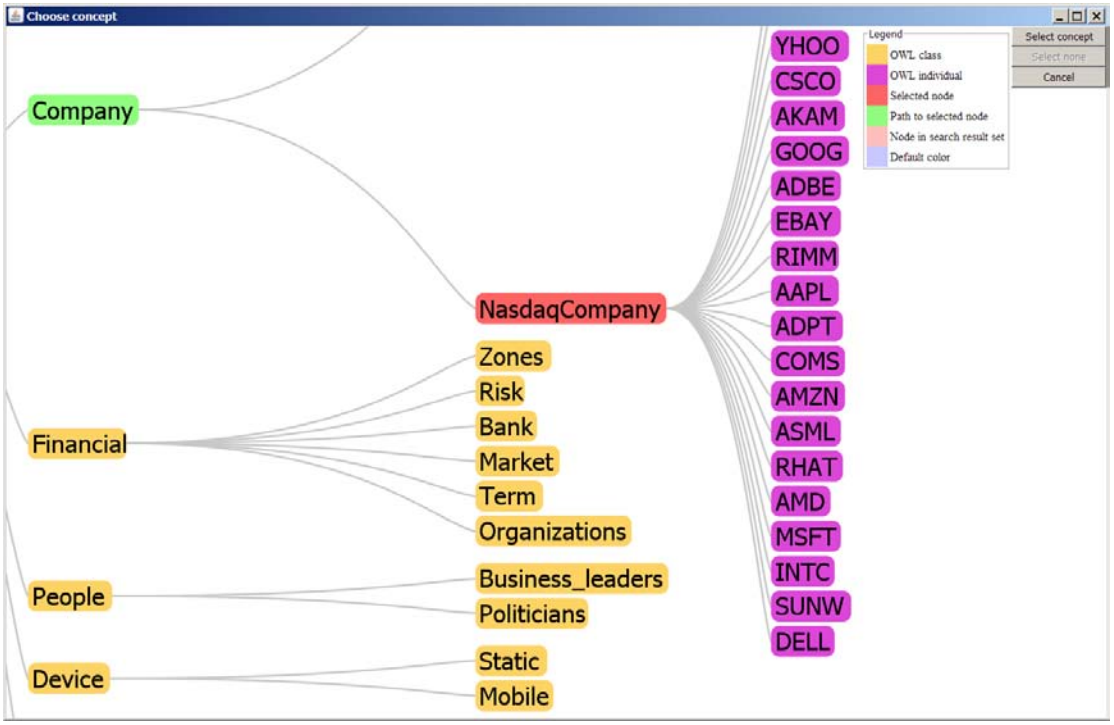

Figure 4 The event editor: concepts example

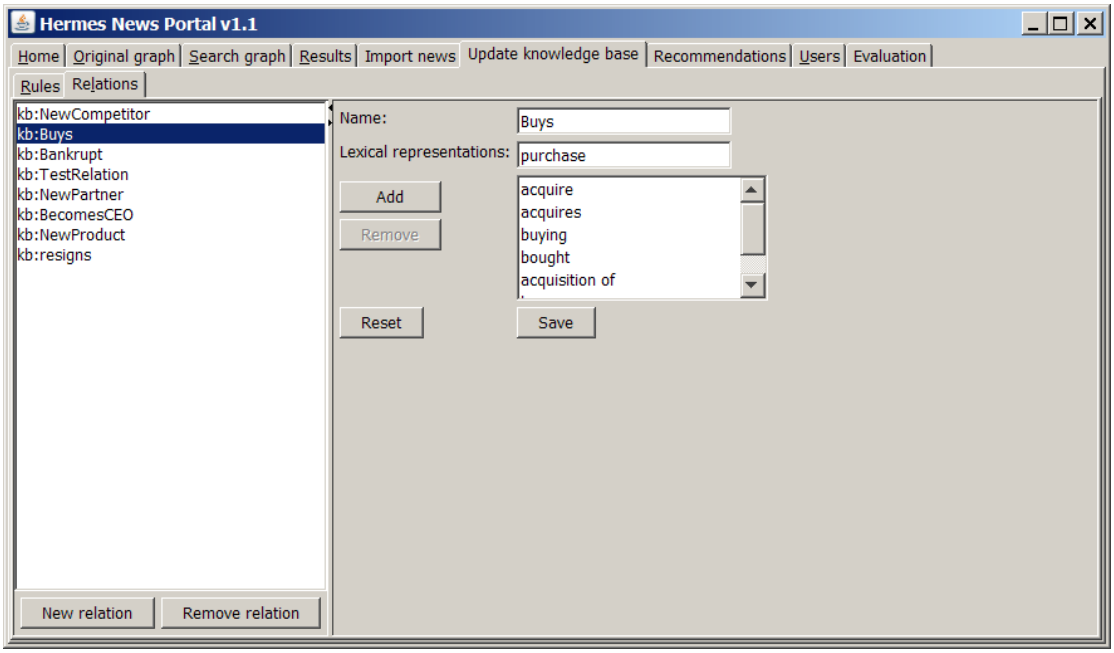

Figure 5 The event editor: relations example

Each event has a list of actions associated which are used to update the knowledge base with the event-based information. The actions are insert triples or delete triples in the knowledge base. Each action can contain multiple insert or delete operations. Figure 6 presents the action editor of the HNP. For example `rb:buy-rule`, which represents the new buy event, has one action associated `rb:removeCompetesWith`. The action removes one triple with the event subject (`<event subject>`, which in this case is a company), the predicate `kb:hasCompetitor`, and event object (`<event object>`, which in this case is a company), and another triple where the subject and object are swapped. The meaning of these actions is that after a company buys another company they cease to compete with each other.

The implementation does not yet make use of user-defined variables to pass information from the information extraction step to the knowledge base updating step, and is based on a more rigid approach employing the predefines variables `<event subject>`,`<event predicate>`, and `<event object>`. The user-defined variables approach would allow the definition, in the future, of complex information extraction patterns based on regular expressions (e.g., optional, repetition, sequence, etc.) where more than three entities are used. Also, the actions are expressed using SPARQL templates and do not make use yet of the action language defined in Hermes, which is implementation-independent (please note that "?" represents the optional operator in regular expressions, and we reserve this symbol for this functionality in a future HNP version).

As can be noticed from Figure 6 we have created a graphical editor for SPARQL insert and delete queries, which allows users to specify the query type, the query triples, as well as inspect the constructed query. In order to support the domain expert during the query creation process for the subjects, predicates, and objects one makes selections from drop down lists filled with existing concepts from the knowledge base, or one of the predefined variables in the framework `<event subject>`, `<event predicate>`, or `<event object>`.

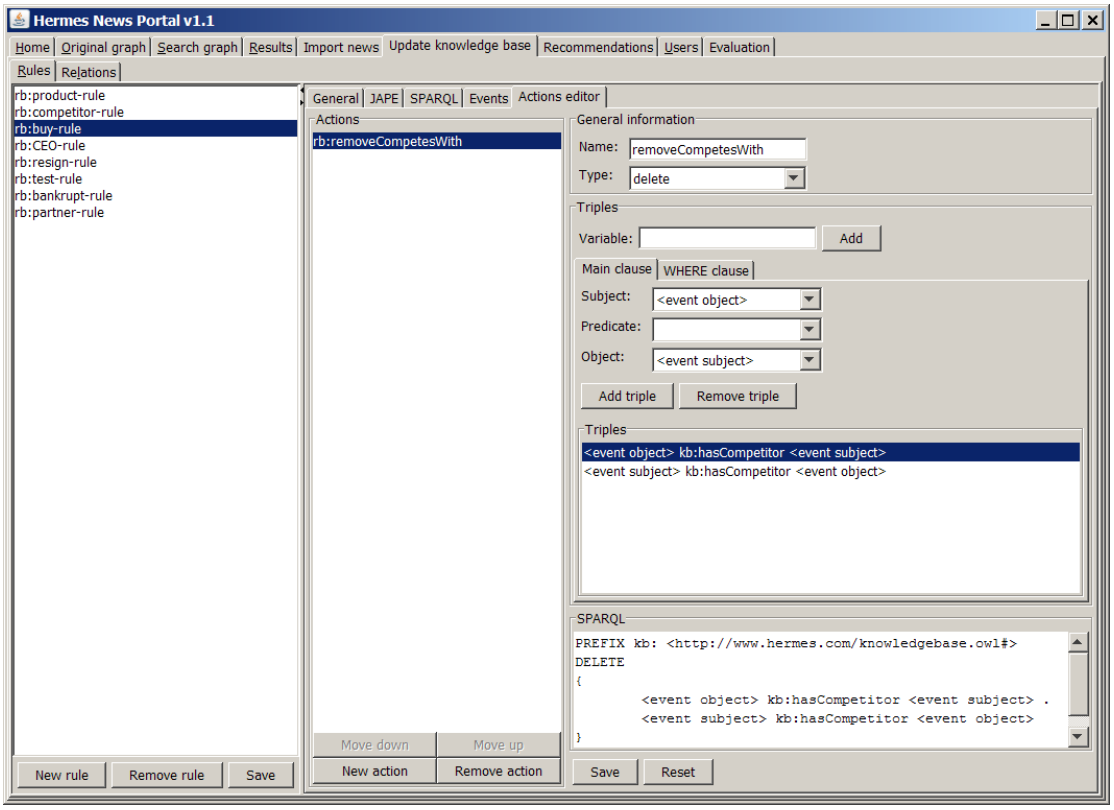

Figure 6 The rule editor: actions example

Event Rules Patterns Execution

In event rules patterns execution, the lexico-semantic patterns are executed on the news items and possibly yield to event instances being detected. For example for the `kb:Buy` event, this amounts to finding the subject and object and binding them to the predefined `<event subject>` and `<event object>` variables, respectively. The implementation of the information extraction patterns is based on Java Annotation Patterns Language (JAPE) rules.

The JAPE rules have two parts: the left-hand side, which describes the patterns to be matched, and the right-hand side, which specifies the actions to be performed when a pattern is matched. The left-hand side language is based on regular expressions. The right-hand language allows for annotations of the matched piece text. For advanced actions (e.g., deleting temporary annotations, manipulating annotations, complex control expressions, etc.) one can make use of Java code on the right-hand side.

For the running example, the JAPE rules will find the buyer and the buyee. In our domain these are both companies that are present in the knowledge base. It is important that these two companies are present in the ontology in order for the `kb:Buy` event to be discovered. After annotating the corresponding lexical representations with the corresponding knowledge base concepts, the event instance is discovered and ready for validation.

Event Validation

The knowledge base updating step is a semi-automatic process, the HNP allows the domain expert to validate the event (instance) before triggering the knowledge base updates based on the event information. The user is presented with the subject, relation (predicate), and optional object, as well as the news item(s) in which the event originated. The domain expert has the options to declare the event:

valid, invalid, or unknown (unknown is used when the domain expert is not able to decide the validity of the event and defers the event for future processing). Figure 7 shows the event validation in the HNP. If the user validates the event the knowledge base is updated in the next substep. In the running example the user has to validate if Sun Microsystems has bought MySQL as suggested in the displayed news item.

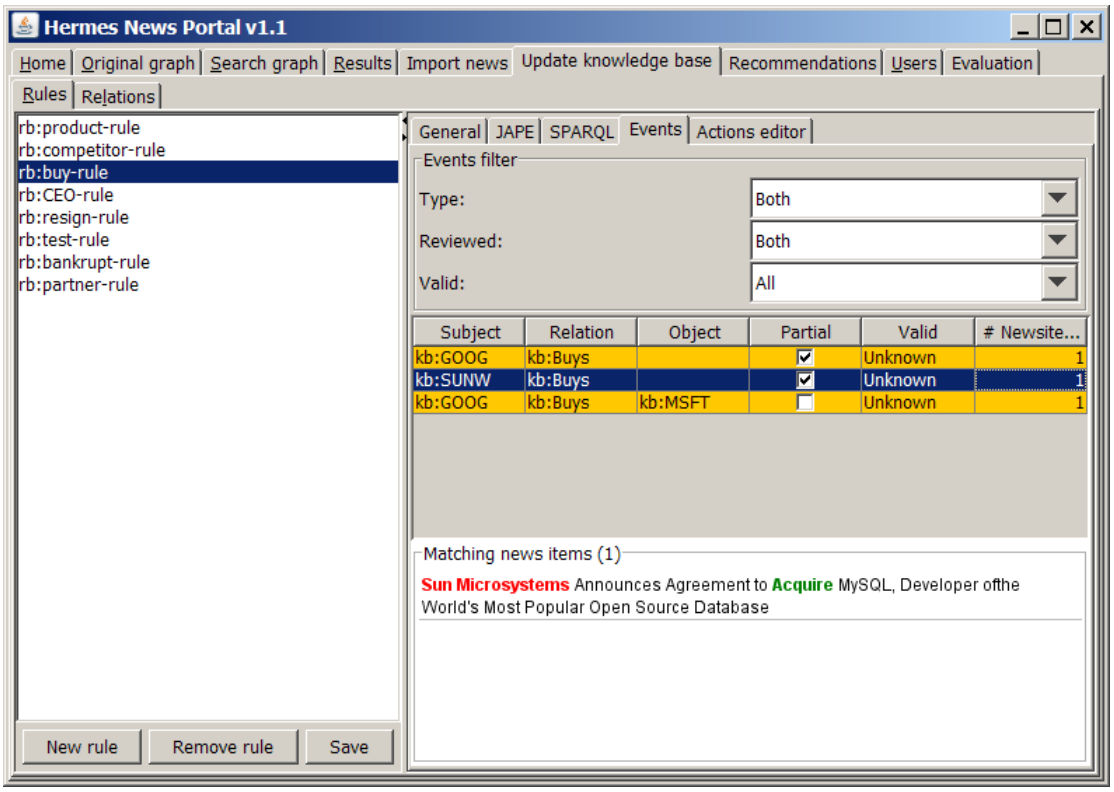

Figure 7 The rule editor: events validation example

Event Rules Actions Execution

The last step of knowledge base updating is the execution of the actions associated with the events found and validated by the user. HNP executes the actions in the order of the found events, and for each event in the specified action order. The actions are SPARQL queries that update the knowledge base with event-related information. At the end of this substep one should have a valid knowledge base with a proper representation of the reality at a given moment.

News Querying

The news querying is based on the graphical interface of the HNP. It is composed of two substeps: query formulation and query execution.

Query Formulation

Figure 8 shows the conceptual graph from which the user can select concepts of interest. Once a user selects a concept, the control panel gets activated by means of which the user can add to the search graph his concepts of interest.

The concepts of interest can only be the current concept, all related concepts including the current one, all related concepts excluding the current one, or all related concepts by means of specified

relationships including/excluding the current concept. The node info panel shows information regarding the selected concept.

The local name of the different concepts is displayed using ovals. In order to emphasize the different types of concepts we use the following coloring scheme for ovals. The selected concepts are displayed in red, the concepts related to the selected one are shown in green, and the ones returned by the keyword search are presented in pink. The other concepts are displayed in yellow for classes and magenta for individuals.

In the example from Figure 8 the user has directly selected the `kb:Google` concept. In the node info panel all the information related to the `kb:Google` concept is displayed: name, competitors, CEO, etc. Then, the user can select the indirect concepts by specifying the `kb:hasCompetitor` relationship between the direct concept and the indirect ones. The user also specifies that the `kb:Google` concept should be kept in the search graph.

After selecting the concepts of interest the user can visualize the search graph. The user can refine its query by deleting some of the concepts, resetting the search, or adding new concepts from the conceptual graph to the search graph. After a number of iterations the user has added all the concepts of interests to the search graph.

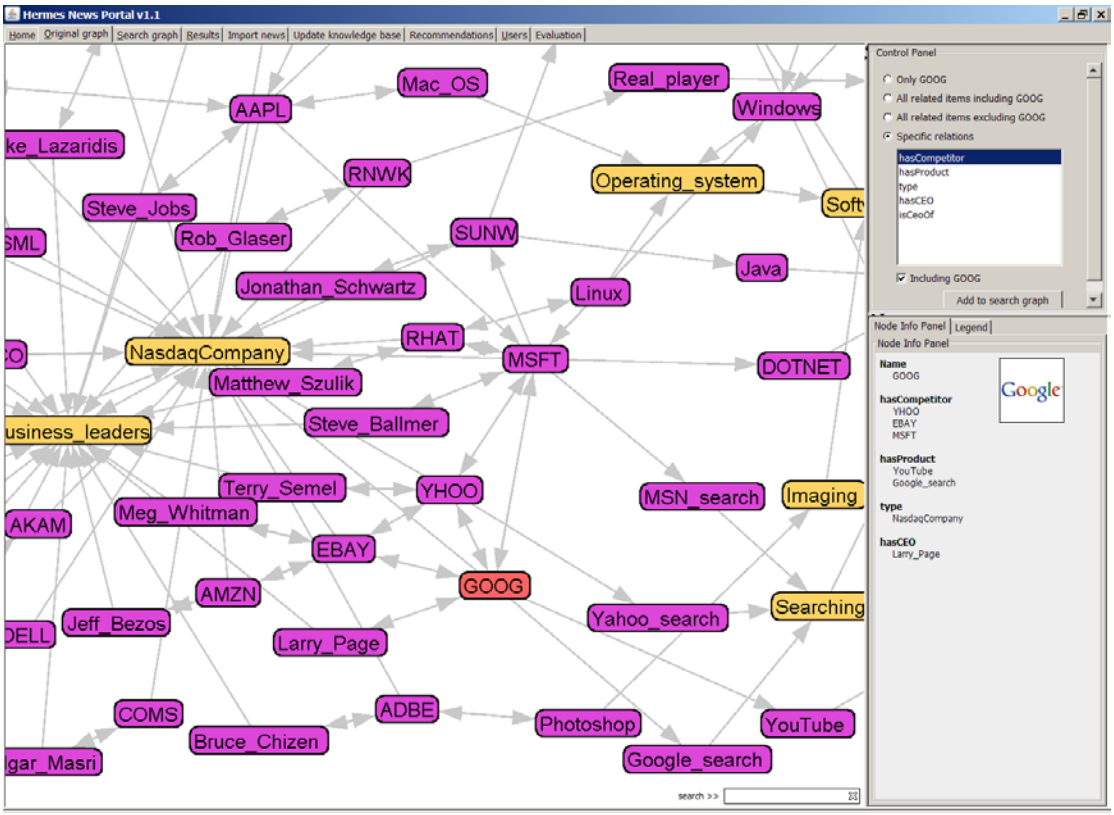

Figure 8 Conceptual graph example

Figure 9 shows the search graph, which presents the selected concepts, in this case `kb:Google` and its competitors: `kb:Microsoft`, `kb:Yahoo!`, and `kb:EBay`. In the time constraint panel the user can specify the desired time restriction. The user can choose between predefined temporal constraints as

the past hour, past day, past week, past two weeks, past three months, past quarter, past half year, and past year, and specific time/date constraints.

As before the node info panel displays information about the currently selected node. However, differently than in the previous situation, only the information given by the specified relationships, in this case the `kb:hasCompetitor` relationship, is displayed.

In this example the user has specified that the news items have to be between 1st of September 2009 and 1st of December 2009, where the current day is 1st of December 2009 (the *last three months* in the user query). Alternatively the user could have selected the past three months option from the predefined temporal constraints.

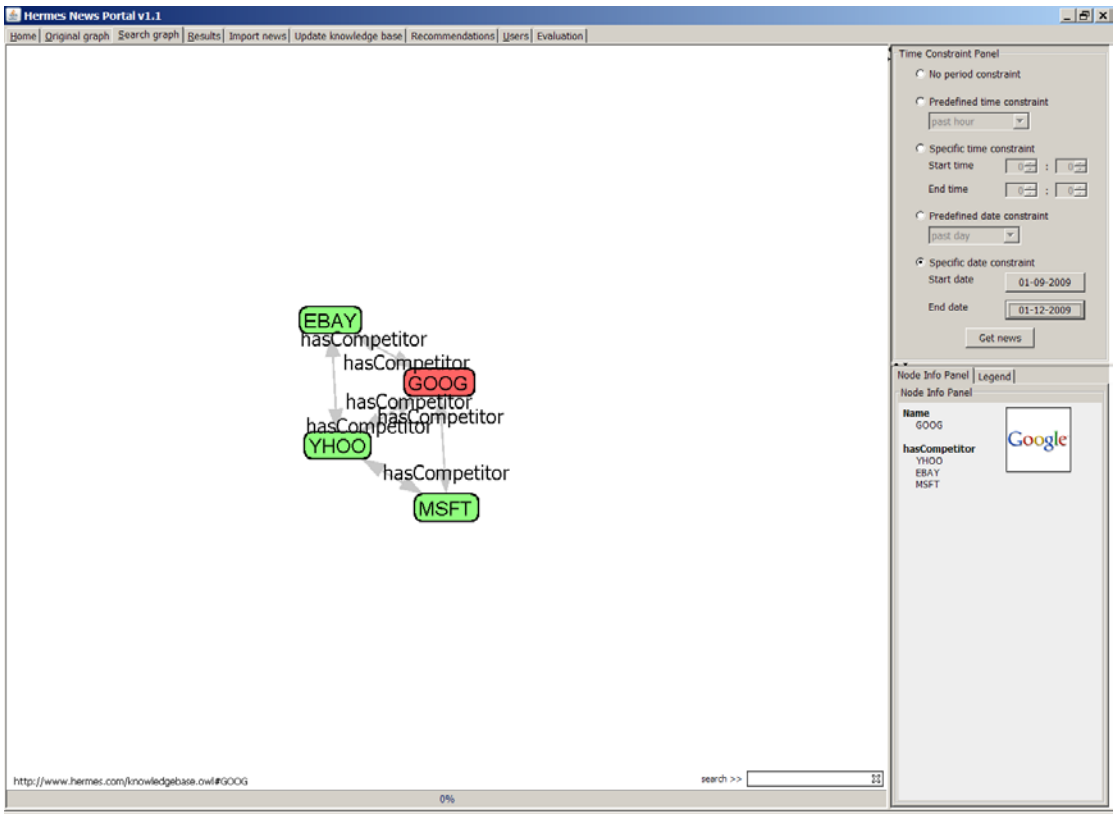

Figure 9 Search graph example

Query Execution

For each search graph a SPARQL query is generated. This query is a SELECT query as it retrieves the news items in which any of the search graph concepts are present. The disjunctive semantics of the search graph with respect to its embedded concepts is naturally specified as an 'or' filter in the SPARQL query. Also, we have decided to use filters to specify the time restrictions that news timestamps need to satisfy. Due to the conjunctive semantics of the time restrictions we modeled them as an 'and' filter in the SPARQL query.

Figure 10 shows the SPARQL query corresponding to the search graph given in Figure 9. This query is hard-coded with XML Schema `xsd:dateTimes` specifying the desired temporal boundaries of the interval in which the timestamps of the desired news items need to be contained.

The first part of the SPARQL query defines that the returned news items should be related to the concepts of interest. The second part of the query is composed of two filters. The first SPARQL filter defines the concepts of interest to be `kb:Google`, `kb:Microsoft`, `kb:Ebay`, and `kb:Yahoo`!. The second SPARQL filter specifies that the timestamp of news items should be between 1st of September 2009 and 1st of December 2009, where 1st of December 2006 is the current day. Both dates are specified using XML Schema `xsd:dateTime` format.

```
PREFIX kb: <http://hermes-news.org/news.owl#>
SELECT ?title
WHERE {
   ?news kb:title ?title .
   ?news kb:time ?date .
   ?news kb:relation ?relation .
   ?relation kb:relatedTo ?concept .
   FILTER (
   ?concept = kb:Google ||
   ?concept = kb:Microsoft ||
   ?concept = kb:Ebay ||
   ?concept = kb:Yahoo
   ) .
   FILTER (
   ?date > "2009-09-01T00:00:00.000+02:00" &&
   ?date < "2009-12-01T00:00:00.000+01:00"
   )
}
```

Figure 10 SPARQL query example

In order to ease the specification of temporal constraints in queries we have extended SPARQL with custom functions. We call the SPARQL language extended with time functions tSPARQL. Please note that SPARQL does naturally support such extensions, tSPARQL being backwards compatible with SPARQL. Figure 11 shows the signature of the time functions that we have added. These functions relate to retrieving the current `xsd:date` and `xsd:time`, the current `xsd:dateTime`, adding/subtracting to a `xsd:dateTime` instance a `xsd:duration`, and subtracting two `xsd:dateTime` instances (`xsd:date`, `xsd:time`, `xsd:dateTime`, and `xsd:duration` are defined by XML Schema).

These functions relate to retrieving the current `xsd:date` and `xsd:time`, the current `xsd:dateTime` instance, adding/subtracting to a `xsd:dateTime` instance a `xsd:duration`, and subtracting two `xsd:dateTime` instances.

```
    xsd:date currentDate()
    xsd:time currentTime()
xsd:dateTime dateTime-add(xsd:dateTime A, xsd:duration B)
xsd:dateTime dateTime-subtract(xsd:dateTime A, xsd:duration B)
xsd:duration dateTime-subtract(xsd:dateTime A, xsd:dateTime B)
```

Figure 11 Custom time functions

Figure 12 depicts the same query as in Figure 10 but now written in tSPARQL. Differently than in the previous case the tSPARQL query is not hard-coded with times and dates, but makes use of custom functions and durations. The semantics of the query is closer to its representation, in our current example that is retrieving the news items *that appeared in the last three months*.

The tSPARQL query uses the `dateTime-substract()` to determine the `xsd:dateTime` of three months ago, `now()` is used to obtain the current `xsd:dateTime`, and it specifies that the news timestamps should be between these two `xsd:dateTime`s. `P0Y3M` is an XML Schema `xsd:duration` constant that specifies a period with 0 number of years and 3 months.

After its creation, the tSPARQL query is executed. As a result, the news items that match the query constraints (concepts of interest and temporal constraints) are being returned. The order of the results is not relevant here.

```
PREFIX kb: <http://hermes-news.org/news.owl#>
SELECT ?title
WHERE {
   ?news kb:title ?title .
   ?news kb:time ?date .
   ?news kb:relation ?relation .
   ?relation hermes:relatedTo ?concept .
   FILTER (
   ?concept = kb:Google ||
   ?concept = kb:Microsoft ||
   ?concept = kb:Ebay ||
   ?concept = kb:Yahoo
   ) .
   FILTER (
   ?date > kb:dateTime-subtract(hermes:now(), P0Y3M) &&
   ?date < kb:now()
   )
}
```

Figure 12 tSPARQL query example

Results Presentation

The results presentation is based on the graphical interface of the HNP. It is composed of two substeps: news sorting and news presentation.

News Sorting

Figure 13 shows the results after the query execution. The news items are sorted based on their relevance degree with the user query. The relevance degrees have been displayed as relative percentages with respect to the best matching news item. The most relevant news items are presented at the top of the results list.

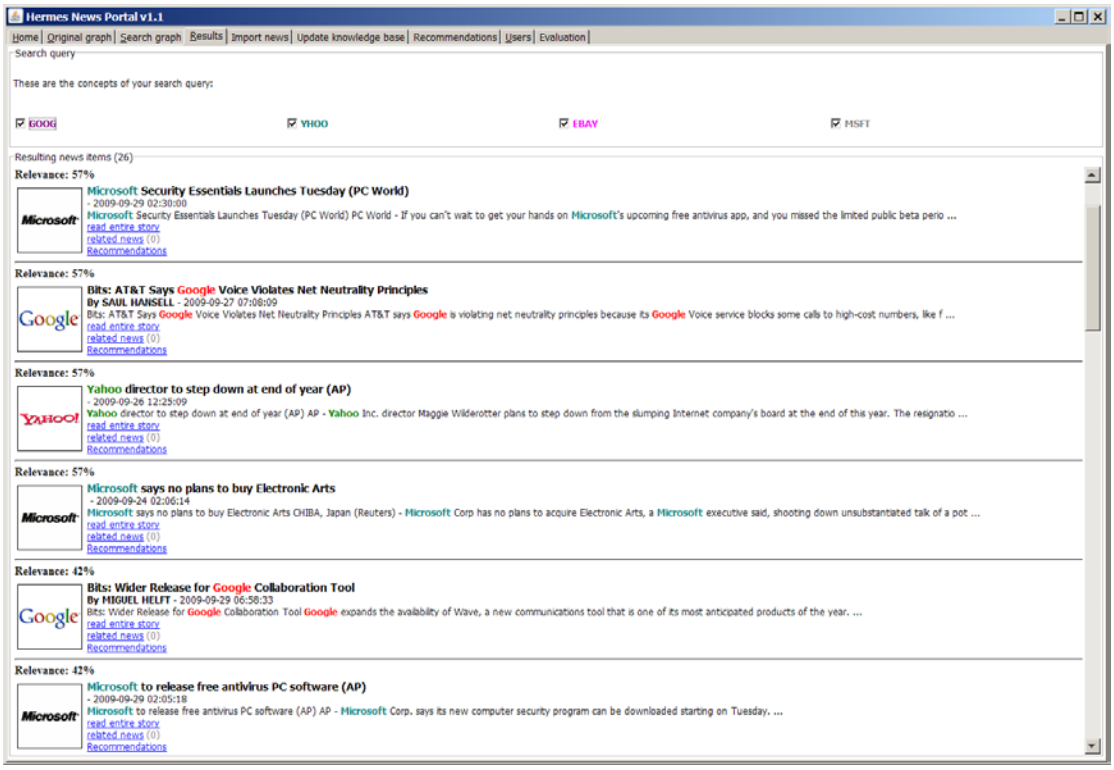

Figure 13 Results presentation example

News Presentation

Figure 13 also lists the concepts of interest from the search graph (top). In addition, the system shows the found lexical representations of the concepts of interests in the returned news items using different colors. The user is able to deselect some of the concepts of interest in order to refine his query and thus limit the result set.

For each news item the system shows a summary containing the title, source, date of publication, and few beginning lines from the news item. Its also shows the icon of the most important query concept found in a news item. The icons correspond often to logos of companies that the user is interested in, and are retrieved from the knowledge base.

EVALUATION

In order to evaluate the performance of the implementation we measured the concept identification precision. Precision was defined as the number of concepts correctly identified in the news items divided by the number of concepts identified in news items. We define recall as the number of concepts correctly identified in the news items divided by the number of concepts that should have been identified in news item. For our current implementation, for concept identification, the precision is 85% and the recall is 81% for a given repository of around 200 news items. For event identification, which is based on a sequence of 2 or 3 concepts (contained in a pattern), we obtained a precision of 62% and recall of 53%.

Precision is based on cumulative errors through our application pipeline based on the HNP's part-of-speech recognition, morphological analysis, and word sense disambiguation algorithm. Despite using only WordNet as a semantic lexicon (compared to other approaches which combine several semantic

lexicons, some domain specific (Navigli & Velardi, 2005)) we obtained high values for precision as many of our concepts' lexical representations are named entities (names of companies, CEO's, locations, etc.) that usually have only one meaning. The high value of recall can be explained by the fact that we do not aim at providing meaning for each (compound) word in news messages but only for the ones that correspond to lexical representations of ontology concepts. The meanings of the news' (compound) words that are not present in the ontology are used only to help the disambiguation process of found concept lexical representations. In addition to the application pipeline, recall is also influenced by the lexical representations of the concepts of interest. The concept lexical representations present in news items but missing from the ontology influence negatively the recall.

A different metric for the performance evaluation is the latency of a news item in the concept identification phase. The obtained average latency time is around 1s which represents the time needed to process a news item from tokenization to concept recognition. The bottleneck lies in the disambiguation step for which distances between synsets need to be computed. The sysnset distances are pre-computed (a limit of 4 for the shortest path length between synsets is used).

Regarding usability we have asked 9 users (students at Erasmus University Rotterdam following a course on Semantic Web technologies including RDF(S), OWL, and SPARQL) to find news items for 3 given natural language queries in two ways: (1) using the Hermes implementation and (2) a SPARQL engine. Most students were able to correctly build the search graph and specify the temporal constraints, as well as the corresponding SPARQL query. All queries were faster specified using the Hermes framework than using SPARQL. Note that we do not claim that it is easier to use Hermes instead of SPARQL for querying RDF graphs, but for expressing a certain set of RDF queries (the ones supported by the search graph with temporal restrictions, which we consider typical for news querying) Hermes seems to be easier to use than SPARQL. Among the features mostly appreciated by students in Hermes were the graphical representations of the conceptual graph, the predefined time functions, and the visual cues employed for emphasizing concepts in returned news items.

Compared with traditional keyword-based search engines for news items (e.g., YourNews, SeAN, Google News, Yahoo! News, etc.) our semantic approach benefits from better precision, as it is able to disambiguate (compound) words, query expressive power, because it allows the selection of indirect concepts (i.e., concepts not directly related to the items of interests), and the support for temporal constraints. A quantitative comparison with non-semantic based approaches is difficult to achieve due to the query limitations that these systems have and the impossibility of using the same news items as inputs in the compared systems.

In the tradeoff between expressivity and usability we decided to keep our queries simple with intuitive semantics so that a broad range of users (casual users, media analysts, stock brokers, etc.) should be able to use Hermes. Nevertheless, we acknowledge that an expert user might need more query expressivity (e.g., optional graph patterns, disjunctive semantics for temporal constraints, etc.) which contributes to the increase in complexity (and thus to the decrease in usability) of the framework. For this purpose we plan to extend the Hermes framework in the future with additional powerful functionality that would enable the generation of a news personalization service family (services targeting novice, average, or expert users).

FUTURE RESEARCH DIRECTIONS

As future work, we would like to extend the Hermes framework by employing multiple semantic lexicons and adding new domain specific concepts to the ontology that are not captured in existing semantic

lexicons. Some domain specific concepts (e.g., domain neologisms) are used in news items while current semantic lexicons, which are not up-to-date, do not include them. We also plan to exploit the structure of the domain ontology in order to compute the similarity between concepts. In this way we are better equipped in determining concept (synset) similarity. For the knowledge base updating we would like to be able to discover new concepts (that go beyond the possibilities of a gazetteer) in news items and add them to our ontology. Also, we would like to distinguish between facts and opinions during event recognition and present to the user for validation only fact-based events, as these represent the true source of real-world updates.

At the current moment the lexico-semantic patterns are defined by the domain expert. We would like to automatically extract patterns for detecting events by using machine learning techniques (e.g., genetic algorithms (Castellanos, Gupta, Wang, & Umeshwar, 2010)). The expressivity of the information extraction language can be extended by using regular expressions (e.g., repetition, sequence, optional, etc.) over ontology concepts. For this we plan to get inspired from the design of current relational and ontology update languages (Losch, Rudolph, Vrandecic, & Studer, 2009). Also, we would like to further refine the expressivity of our query language by allowing the use of Boolean operators (AND, OR, and NOT) as well as pattern-based (sequence-based) queries.

In addition we would like to explore the possibilities to redefine the domain ontology as a time-based representation, where instances have a certain time validity associated with them (Milea, Frasincar, & Kaymak, 2008) (Frasincar, Milea, & Kaymak, 2010). These temporal extensions to the ontology would enable us to better reason with the temporal contextual information available for our domain. Regarding the news presentation we would like to include timelines that would allow user to visualize results in a temporal dimension. Also, for news presentation we would like to experiment with the use of snippets of the news message that match the user query instead of the current few beginning lines from the returned news items.

Another direction that we would like to pursue is that of semantic adaptation of news items based on a user model. The user preferences now represented in the (temporary) search graph would be represented in a (stored) user model which is continuously adapted based on user behavior. In order not to bother the user with already seen content, we would like to be able to filter news items that provide new information by using a novelty control mechanism (Gabrilovich, Dumais, & Horvitz, 2004). We believe that our semantic approach can be successfully applied for modeling dissimilarities between news items and thus be able to recommend only news carrying novel content. In a different scenario, by measuring the similarities between news items, we would be able to recommend news items related to the same story but issued at different moments in time and/or by different institutions.

Regarding HNP we would like to implement news items duplicates removal, timestamps-based sorting of relevant news items, and the user-defined variables communication between information extraction and knowledge base updating -as proposed in the Hermes framework. Additionally, we also wish to implement the previously proposed extensions to the Hermes framework: enriching our knowledge sources with multiple lexicons and domain-specific concepts, adding a user model and employing it to adapt system functionality, and filtering news that provide novel content. Also, we would like to test the usage of data structures for fast data access (e.g., hash maps) for ontology access in a news-centric approach where the concept lexical representations are identified during a single news item traversal. Having a constant access time to concept lexical representations and taking in consideration that the number of lexical representations in a news item is smaller than in an ontology might reduce the time needed for the news classification step (for run-time processing of news items).

Additionally, we would like to conduct a more extensive evaluation procedure of the Hermes implementation. Based on detailed questionnaires and measuring the time spent on building queries given in natural language, we can obtain more empirical evidence on the system usability. The accuracy of the proposed relevance degree (based on concept identification) could be determined by measuring the access order and reading time of news item in the result list (accessing the first items first and spending substantial time for reading them are good indications that the returned items are relevant). In addition, we want to experiment with other domains (e.g., politics, sports, etc.) and analyze the precision, recall, and latency of our implementation for these new fields. The genericity of our approach only asks for the definition of a new domain ontology and domain-specific news feeds that need to be plugged into our implementation.

CONCLUSION

The Hermes framework proposes a sequence of steps to be followed for building personalized news services. The input for these systems comprises RSS news feeds and the output are news items fulfilling user needs. The Hermes approach is based on a domain ontology used for classifying news items and to support the user define his concepts of interest. In addition to concepts and their relationships, the domain ontology stores for each entity its associated lexical representations, some being retrieved from a semantic lexicon. Also, the user can specify temporal constraints that the news item needs to obey. For this we have propose a number of custom functions to support the user during the query creation process. The knowledge base is updated in a semi-automatic fashion using news information. In order to facilitate the user browsing of results we have ordered the news items based on their relevance to the user query.

The Hermes News Portal (HNP) is an implementation of the Hermes framework. The domain ontology is specified in OWL and as a query language we used SPARQL. As a semantic lexicon we employed WordNet, one of the most popular English dictionaries available online. For the natural processing pipeline we have used ANNIE from GATE extended with our own ontology gazetteer and word sense disambiguation engine. For representing temporal constraints we have extended the SPARQL language with temporal functions. JAPE from GATE was used for implementing the lexico-semantic patterns. For the knowledge base updates we employed SPARQL/Update.

Differently than Google News and Yahoo! News, Hermes is able to exploit the background information stored in ontologies for retrieving user's items of interest. In this way the user does not need to explicitly define all the instances involved in the query by making use of the concept relationships for specifying his concepts of interest. In addition to the concepts of interest, the user is able to specify temporal constraints in his query. Another key feature of Hermes is the word sense disambiguation procedure, which is not used in related approaches as SeAN, YourNews, NewsDude, MyPlanet, SemNews, or QuickStep. The word sense disambiguation step increases the accuracy of news classifications, by making sure that the found lexical representations indeed correspond to the meaning of the domain ontology concepts. Hermes is also the only personalization framework that to our knowledge allows for semi-automatic update of its knowledge base.

ACKNOWLEDGEMENTS

The authors are partially supported by the NWO Physical Sciences Free Competition Project 612.001.009: Financial Events Recognition in News for Algorithmic Trading (FERNAT) and EU funded IST-STREP Project FP6-26896: Time-Determined Ontology-Based Information System for Real Time Stock Market Analysis (TOWL). Also, we would like to thank Kim Schouten, Philip Ruijgrok, Leonard Levering,

Wouter Rijvordt, Maarten Mulders, and Hanno Embregts for their contribution to the Hermes framework.

REFERENCES

Aha, D. W., Kibler, D., & Albert, M. K. (1991). Instance-Based Learning Algorithms *Machine Learning, 6*(1), 37-66.

Ahn, J.-w., Brusilovsky, P., Grady, J., He, D., & Syn, S. Y. (2007). Open User Profiles for Adaptive News Systems: Help or Harm? In *16th International Conference on World Wide Web (WWW 2007)* (pp. 11-20). New York, NY: ACM.

Ardissono, L., Console, L., & Torre, I. (2001). An Adaptive System for the Personalized Access to News. *AI Communications, 14*(3), 129-147.

Bechhofer, S., Harmelen, F. v., Hendler, J., Horrocks, I., McGuinness, D. L., Patel-Schneider, P. F., et al. (2004). *OWL Web Ontology Language Reference* W3C Recommendation 10 February 2004.

Berners-Lee, T., Hendler, J., & Lassila, O. (2001). The Semantic Web. *Scientific American, 284*(5), 34-43.

Billsus, D., & Pazzani, M. J. (1999). A Personal News Agent that Talks, Learns and Explains. In *Third International Conference on Autonomous Agents (Agents 1999)* (pp. 268-275). New York, NY: ACM.

Borsje, J., & Giles, J. (2010). OWL2Prefuse. from http://owl2prefuse.sourceforge.net/index.php

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)* (pp. 2415-2420). New York, NY: ACM.

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

Castellanos, M., Gupta, C., Wang, S., & Umeshwar. (2010). Leveraging Web streams for Contractual Situational Awareness in Operational BI. In *International Workshop on Business intelligencE and the WEB (BEWEB 2010)*. New York, NY: ACM.

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

Finlayson, M. (2010). The MIT Java Wordnet Interface (JWI). from http://www.mit.edu/~markaf/projects/wordnet/

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). tOWL: Integrating Time in OWL. In R. D. Virgilio, F. Giunchiglia & L. Tanca (Eds.), *Semantic Web Information Management: A Model-Based Perspective* (pp. 225-246). Berlin Heidelberg, Germany: Springer.

Gabrilovich, E., Dumais, S., & Horvitz, E. (2004). Newsjunkie: Providing Personalized Newsfeeds via Analysis of Information Novelty. In *13th International Conference on World Wide Web (WWW 2004)* (pp. 482-490). New York, NY: ACM.

Guarino, N., & Welty, C. A. (2002). Evaluating Ontological Decisions with OntoClean. *Communications of the ACM 45*(1), 61-65.

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, pp. 62.62). Washington, DC: IEEE Computer Society.

Jena Development Team. (2010a). A Semantic Web Framework for Java (Jena). from http://jena.sourceforge.net/

Jena Development Team. (2010b). A SPARQL Processor for Jena (ARQ). from http://jena.sourceforge.net/ARQ/

Kalfoglou, Y., Domingue, J., Motta, E., Vargas-Vera, M., & Shum, S. B. (2001). *myPlanet: An Ontology-Driven Web-Based Personalized News Service.* Paper presented at the Workshop on Ontologies and Information Sharing (IJCAI 2001).

Kandel, E., & Marx, L. M. (1997). NASDAQ Market Structure and Spread Patterns. *Journal of Financial Economics, 45*(1), 61-89.

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

Losch, U., Rudolph, S., Vrandecic, D., & Studer, R. (2009). Tempus Fugit. In *6th European Semantic Web Conference (ESWC 2009)* (pp. 278-292). Berlin Heidelberg, Germany: Springer.

Micu, A., Mast, L., Milea, V., Frasincar, F., & Kaymak, U. (2008). Financial News Analysis Using a Semantic Web Approach. In A. Zilli, E. Damiani, P. Ceravolo, A. Corallo & G. Elia (Eds.), *Semantic Knowledge Management: An Ontology-Based Framework* (pp. 311-328). Hershey, Pennsylvania: IGI Global.

Middleton, S. E., Shadbolt, N. R., & Roure, D. C. D. (2004). Ontological User Profiling in Recommender Systems. *ACM Transactions on Information Systems, 22*(1), 54-88.

Milea, V., Frasincar, F., & Kaymak, U. (2008). Knowledge Engineering in a Temporal Semantic Web Context. In *The Eighth International Conference on Web Engineering (ICWE 2008)* (pp. 65-74). Washington, DC: IEEE Computer Society Press.

Motta, E. (1999). *Reusable Components for Knowledge Modelling: Case Studies in Parametric Design Problem Solving* (Vol. 53). Amsterdam, the Netherlands: IOS Press.

Navigli, R., & Velardi, P. (2005). Structural Semantic Interconnections: a Knowledge-Based Approach to Word Sense Disambiguation. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 27*(7), 1063-1074.

Nirenburg, S., & Raskin, V. (2001). Ontological Semantics, Formal Ontology, and Ambiguity. In *Formal Ontology in Information Systems (FOIS 2001)* (pp. 151-161). New York, NY: ACM.

Noy, N., & Rector, A. (2006). *Defining N-ary Relations on the Semantic Web*: W3C Working Group Note 12 April 2006.

Princeton Cognitive Science Laboratory. (2010). A Lexical Database for the English Language (WordNet). from http://wordnet.princeton.edu/

Prud'hommeaux, E., & Seaborne, A. (2008). *SPARQL Query Language for RDF*: W3C Recommendation 15 January 2008.

Salton, G. (1971). *The SMART Retrieval System—Experiments in Automatic Document Processing*. Upper Saddle River, NJ: Prentice-Hall.

Salton, G., & McGill, M. J. (1983). *Introduction to Modern Retrieval*. New York, NY: McGraw-Hill.

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)* (pp. 854-861). New York, NY: ACM.

Seaborne, A. (2004). *RDQL - A Query Language for RDF*: W3C Member Submission 9 January 2004.

Seaborne, A., Manjunath, G., Bizer, C., Breslin, J., Das, S., Davis, I., et al. (2008). *SPARQL Update: A language for Updating RDF Graphs*: W3C Member Submission 15 July 2008.

The Berkeley Institute of Design. (2010). The Prefuse Visualization Toolkit. from http://prefuse.org/

The Stanford Natural Language Processing Group. (2010). The Stanford Parser: A Statistical Parser. from http://nlp.stanford.edu/software/lex-parser.shtml

Winer, D. (2003). *RSS 2.0 Specification*: Harvard Law School.

KEY TERMS AND DEFINITIONS

Domain ontology: a formal shared specification of a domain conceptualization (includes both schema and instance of the represented domain).

Knowledge base: a domain ontology instance (includes also instance types but does not contain type definitions).

Concept: a class or instance from the domain ontology.

Conceptual graph: the graph representation of the knowledge base.

Search graph: a subgraph of the conceptual graph used to define the search query.

News classification: assigning concepts from the domain ontology to news items.

Knowledge base updating: adding new instances and/or relationships to the domain ontology.

Results presentation: displaying of the news items matching the search query.

Lexico-semantic patterns: information extraction patterns that make use of lexical representations and/or concepts.

tSPARQL: SPARQL extended with temporal operators.