Knowledge Engineering in a Temporal Semantic Web Context

Viorel Milea, Flavius Frasincar, and Uzay Kaymak
Faculty of Economics, Erasmus University Rotterdam
P.O. Box 1738, 3000 DR Rotterdam, The Netherlands
{*milea, frasincar, kaymak*}*@few.eur.nl*

Abstract

The emergence of Web 2.0 and the Semantic Web as established technologies is fostering a whole new breed of Web applications and systems. These are often centered around knowledge engineering and context awareness. However, adequate temporal formalisms underlying context awareness are currently scarce. Our focus in this paper is two-fold. We first introduce a new OWL-based temporal formalism - TOWL - for the representation of time, change, and state transitions. Based hereon we present a financial Web-based application centered around the aggregation of stock recommendations and financial data.

1. Introduction

Information on the Web is mostly textual in nature. Its character is descriptive and meaningless without its most skilled interpreter: the human brain. If the goal of automating the aggregation of vast amounts of information is to be achieved, then this information should be described in a machine-readable way enabling applications to at least simulate some understanding of the data being processed. The emergence of Web 2.0 [12] and the Semantic Web [2] as established technologies is fuelling a transition from a web of data to a web of knowledge. In turn, this knowledge rich environment is fostering a whole new breed of Web applications and systems, centered around knowledge aggregation and context awareness. Focussing on the latter, it can rightfully be stated that enabling context awareness involves the existence of adequate temporal formalisms - currently very scarce in a Semantic Web context. This results in adhoc (and often not reusable) solutions for dealing with temporal aspects on the Web.

One of the domains with a prominent temporal aspect, which forms the focus of our current research, is the financial one. More specifically, we seek to explore the area of engineering Web applications for automated trading, an area far too little investigated in such a context. One of the motivations behind this choice finds its origins in the fact that, in 2006, automated trading accounted for no less than a third of all share trades in America. By 2010, according to consultancy firm Aite Group, it will make up more than half the share volumes and a fifth of options trades [3].

The main drive of this increased demand for automated support in financial markets can be accounted to different factors. One of the most obvious reasons underlying this trend is the tight correlation between the timing of a trade and the generated return. A number of aspects play a part herein, such as the time required for the aggregation of the relevant information on the human side; this period introduces an unavoidable lower bound on the time of the trade relative to the moment when it becomes profitable based on the available signals. Another factor that fuels the involvement of automated systems on the decision-side of financial markets consists of the discovery of arbitrage opportunities. The increased efficiency of mature financial markets, as well as the large number of assets and derivatives available, make the identification of market inefficiencies highly challenging. Automating at least parts of this process should result in an increased coverage of the market, as well as in an increased quantitative expectation from this type of speculative returns. Finally, we underline the human aspect as one of the main drives behind the migration from trader-driven investments to computer(-aided) trades. We refer here mostly to error proneness, especially in conditions of stress or extreme situations.

Although seemingly not directly related to automated trading, the Semantic Web may come to meet the increased technological demands emerging in the world of trading [10, 15]. In achieving this purpose, we deem it necessary to provide extensions to current Semantic Web languages, thus making the latter more suitable for the knowledge we seek to represent. One such extension is presented in this paper and concerns a temporal ontology language based on OWL-DL - the TOWL language. This language stands at the basis of the financial application we present in this paper, and forms one of the key ingredients allowing the aggregation of historical stock recommendations and financial

978-0-7695-3261-5/08 $25.00 © 2008 IEEE
DOI 10.1109/ICWE.2008.10

data.

This paper is organized as follows. In Section 2 we present an overview of known approaches towards temporal representations on the Semantic Web. In Section 3 we introduce our temporal extension of OWL-DL: the TOWL language. Section 4 presents the TOWL-based financial application developed in the context of the current research both in terms of its key features as well as the results generated. We wrap up in Section 5 with some conclusions regarding the main ideas presented in this paper and possible future work.

2. Related Work

The focus of this section is on previous approaches regarding temporal Semantic Web extensions. The first approach we present, in Section 2.1 regards the extension of RDF with temporal labeling of triples. In Section 2.2 we give an overview of an OWL ontology of time, while in Section 2.3 we report a previous effort of integrating a perdurantist approach towards change in OWL, again in the form of an ontology.

2.1. Temporal RDF

A rather extensive approach towards extending ontology languages with a temporal dimension is reported in [4]. This work is similar to our current goal as it concerns the ability to represent temporal information in ontologies, but differs in that the language considered is the Resource Description Framework (RDF). The concrete result of this approach consists of temporal RDF graphs with underlying temporal semantics allowing for temporal entailment.

The extension presented in [4] with regard to RDF consists of a vocabulary extension focused on temporal labeling. This vocabulary extension enables the labeling of triples with intervals, thus allowing for the representation of the time period during which the triple is true. Building further upon this idea, the authors introduce the concepts of *graph slices* and *graph snapshots*. Given a time t, graph slices can be seen as subgraphs consisting of all triples with temporal label t, while graph snapshots consist of all triples holding true at t.

This labeling approach is suitable for the representation of both transaction and valid time, as known from temporal databases. Additionally, it preserves the spirit of the distributed and extensible nature of RDF, while keeping object proliferation limited in scenarios where changes are frequent [4]. The time used is a discrete and linearly ordered domain, again presenting resemblance with temporal databases.

However, the approach presented in [4] is mainly focused on representing different versions of RDF graphs, as they hold at different moments in time. In other words, temporal RDF is centered around representing and tracking change. For our current purpose, although relevant, we deem this to be insufficient, as we seek represent state transitions next to the ephemeral traits of entities.

2.2. OWL-Time

The initial purpose behind the design of a time ontology was to represent the temporal content of Web Pages and the temporal properties of Web Services (DAML-Time) [5]. The latest version of this ontology is represented in OWL, and is currently a W3C working draft, as of September 27th, 2006.

The time ontology is built around the *TemporalEntity* class and the relations describing its individuals. Temporal entities may be of two types: *Instant* (point-like moments in time) and *Interval* (time descriptions having a duration, represented as ordered pairs of *Instant* individuals). Additionally, the ontology contains classes meant to describe different other temporal concepts, such as duration (*DurationDescription*), dates and times (*DateTimeDescription*), temporal units (*TemporalUnit*) and alternative representations for the days of the week (*DayOfWeek*).

The main relations describing individuals of type *Instant* are the functional properties *begins* (the beginning, same as the end in case of an instant) and *ends* (the end, same as the beginning in case of an instant). The range of the properties are objects of type *InstantThing*. The actual time belonging to these individuals can be expressed as of the internal type *CalendarClockDescription* or as *xsd:dateTime* of XML Schema. Intervals exhibit, amongst others, the same properties as instants, with the same flexibility in the representation of the actual date and/or time. Additionally, the property *inside* may be defined in the case of individuals of type *Interval*, and describing a relation between an *Instant* and an *Interval*, equivalent to asserting that some individual of type *Instant* is within the bounds of the individual of type *Interval*.

In the context of designing a temporal language, the expressiveness offered by OWL-Time addresses mainly the issue of the representation of time in its qualitative nature. However, the main building block of this approach is OWL-DL, resulting in a limited representational power even at this level. A very simple example relates to the semantics of an interval - the left bound of the interval should always be smaller than the right bound of that same interval - which cannot be represented in OWL-DL. Despite this, the approach also exhibits a number of interesting features, such as the reference to XML Schema for the representation of dates and times, and the use of intervals as the main building block for more complex constructions.

2.3. An OWL Ontology for Fluents

Perdurantism, or four-dimensionalism, relates to the view of entities as having more than only their spatial parts: they also have a temporal part. For the current purpose we focus on perdurantism and the corresponding view on the temporal parts of an object: "... something persists iff, somehow or other, it exists at various times; something perdures iff it persists by having different temporal parts, or stages, at different times, though no part of it is wholly present at more than one time" [6].

An approach related to incorporating perdurants through the use of timeslices and fluents in OWL-DL is presented in [14], where the authors develop a reusable ontology for fluents in OWL-DL. The fundamental building blocks of this representation are timeslices and fluents. Timeslices represent the temporal parts of a specific entity at clear moments in time and the concept itself is then defined as all of its timeslices. Fluents are nothing more than properties that hold at a specific moment in time, may this time be an instant (point-like representation of time) or an interval. The ontology for fluents as defined in [14] is reproduced in Figure 1.

```
Ontology(4dFluents
  Class(TimeSlice)
  DisjointClasses(TimeSlice TimeInterval)
  Property(fluentProperty Symmetric
    domain(TimeSlice)
    range(TimeSlice))
  Property(tsTimeSliceOf Functional
    domain(TimeSlice)
    range(complementOf(TimeInterval))
  Property(tsTimeInterval Functional
    domain(TimeSlice)
    range(TimeInterval)))
```

Figure 1. The original fluents ontology as presented in [14].

The approach in [14] comes to address a relevant issue, namely the representation of change in OWL ontologies. However, as in the case of OWL-Time, relying solely on OWL-DL for such a goal limits in several ways the power of the approach. The semantic limitation regarding temporal aspect translates to little to no reasoning support in such contexts.

3. The TOWL Language

In this section we formally introduce the TOWL language [11]. The concepts discussed are introduced both in abstract as well as description logics (DL) syntax. A further differentiation is made between syntax and semantics of the concrete domain and the syntax and semantics of fluents and timeslices. Finally, we differentiate between TBox and ABox syntax and semantics, and present these separately.

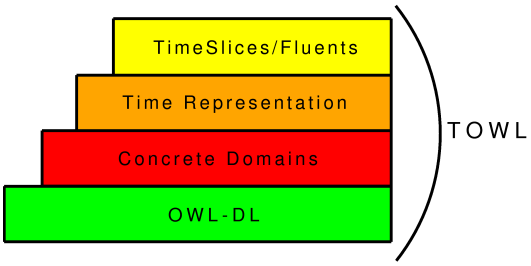

Figure 2. The TOWL layer cake.

An overview of the layers introduced by TOWL as an extension to OWL-DL is given in Figure 2. In what follows we present the different layers in more detail, introducing the syntax and semantics belonging to each. It should be noted that some of the layers do not necessarily introduce a semantic extension, as in the case of timeslices and fluents where it can be argued that this layer concerns syntactic sugaring.

3.1. Concrete Domains

The first step towards increasing the expressiveness of OWL-DL in a temporal fashion consists of extending the capabilities of the language regarding concrete domains. OWL-DL does offer a certain level of support for concrete domains. It is thus possible to represent concrete traits of entities based on quantifiable attribute values and datatype properties. The datatype properties can be created at the design stage and linked to XML Schema Datatypes. In this fashion, simple concepts such as age, height, and even temporal durations may be represented.

However, for the current purpose, we deem this to be insufficient. Truly meaningful representations in a temporal context relate, for a a large part, to complex temporal restrictions enabled by the language. Such constructions are usually built using function compositions, equivalent to functional role chains in OWL-DL. Complex class definitions involving a temporal aspect are represented based on such chains, with the important characteristic that this definition involves the concrete domain. Translating this to OWL-DL involves the representation of restrictions through concrete domain predicates imposing some restriction of functional role chains. This type of representations are not only relevant in a temporal context. Even in a static environment one can envision some concrete relation restricting a class definition based on concrete feature chains. The general form of a functional role chain in DL is:

$$f_1 \circ f_2 \circ ... \circ f_n \circ g$$

In TOWL we make this type of constructs available by means of syntactic sugaring. The representation of a concrete feature chain in TOWL abstract syntax is as follows:

$$\texttt{ConcreteFeatureChain}(f_1\ \ f_2\ \ ...\ \ f_n\ \ g)$$

where f_1,...,f_n are abstract features and g is a concrete feature pointing to some value in the concrete domain.

In what follows, we denote by u_i concrete feature chains of the form introduced above. Developing hereon, we additionally introduce restrictions imposed through the use of concrete domain predicates over such chains. The general form of such restrictions in DL is:

$$\exists u_1, u_2.p_d,$$

where p_d is a concrete domain predicate and $u_1,\ u_2$ are concrete feature chains. In TOWL, such restrictions are represented as:

$$\texttt{dataSomeValuesFrom}\,(u_1\ u_2\ p_d),$$

where $u_1,\ u_2,\ p_d$ preserve their meaning as previously presented. Additionally, we can also represent universal quantification over such restrictions, what in DL would be represented as:

$$\forall u_1, u_2.p_d,$$

which translates, in TOWL abstract syntax, to:

$$\texttt{dataAllValuesFrom}\,(u_1\ u_2\ p_d),$$

For example, if we base our approach on a concrete domain describing dates and times (which in turn is based on the set $\mathbb{Q}$ of rational numbers with $<$ and $=$ as its two binary predicates), we can express restrictions of the following form in DL syntax:

$$\exists u_1, u_2. <$$
$$\forall u_1, u_2. =$$

which are completely equivalent to the following representations in TOWL abstract syntax:

$$\texttt{dataSomeValuesFrom}\,(u_1\ u_2\ \texttt{numeric-smaller-than})$$
$$\texttt{dataAllValuesFrom}\,(u_1\ u_2\ \texttt{numeric-equal})$$

To summarize, the concrete domains layer in TOWL builds upon the expressiveness offered by concrete domains added to OWL-DL. This increased expressiveness comes from allowing complex class definitions where a concrete domain predicate that is used to restrict paths in the form of concrete feature chains.

A special type of concrete domain is represented by *constraint systems* [7]. The main discriminating characteristic of such systems is the type of predicates they are based on. Unlike traditional concrete domains, constraint systems are based on a set of jointly-exhaustive and pairwise disjoint binary relations. This makes constraint systems suitable for temporal representations based on intervals and Allen's interval relations [1], as we discuss in the next section. Additionally, such an extensions maintains the decidability of the language, as shown in [7].

3.2. Time Representation

The concrete domain in the TOWL context, as presented in the previous section, enables the representation of time in the language. Under this TOWL layer we include basic representations of time in the form of temporal intervals, as well as a number of temporal relations between these intervals in the form of Allen's 13 interval relations. This forms the basis for our approach, as it allows the definition of complex restrictions, such as the one described in the previous section, but this time having a temporal character. The type of concrete domain considered is a constraint system based on intervals and Allen's relations that may hold between intervals [8].

By employing this particular concrete domain, we can express Allen's 13 [1] interval relations in the language and enable the representation of temporal dependencies between events. It should be noted that all relations between intervals can be translated to formulas in terms of the intervals' endpoints.

3.3. Timeslices & Fluents

The concrete domain approach towards representing quantitative time in ontologies as presented in the previous sections forms the basis for our approach. Building further upon these blocks, we seek to represent temporal aspects of entities other than timespan. In this context, the final level of expressiveness that we enable in TOWL regards change and state transitions. The perdurantist approach outlined in Section 2.3 forms the foundation of this type of features. The current approach builds upon the concrete domain layer for the representation of time as described in this document. Up to a certain level, it can be argued that the fluents and timeslices employed for the representation of temporal information do not go beyond the expressiveness of OWL-DL. Rather, fluents and timeslices represent a kind of vocabulary employed for the representation of temporal parts of individuals that change some property in time. However, the semantics of fluents as envisioned for TOWL enforces a number of restrictions on TOWL specific concepts, and most importantly on fluents and timeslices. Some interesting features fuel the interdependence between the concrete domain and the timeslices/fluents approach and relate

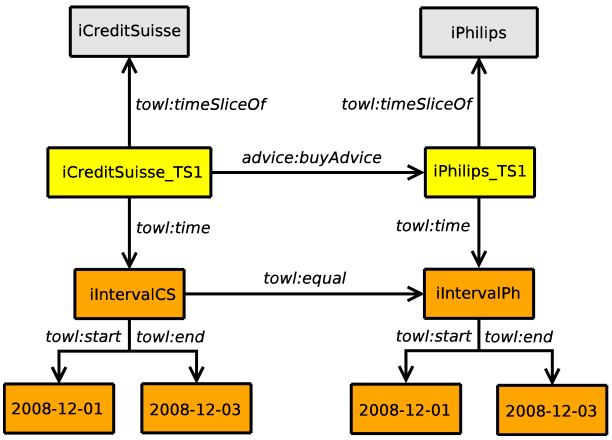

Figure 3. Temporal restrictions on timeslices connected by fluents.

mostly to the restrictions this approach imposes on the very concepts it introduces. A concrete example of this interdependence consists of employing the definition of the interval as introduced in the previous sections in defining the time associated with timeslices (the period over which timeslices hold true).

One such restriction relates to the fact that fluents only relate timeslices that hold over the same time interval. Representing such a restriction involves the concept of equality of (feature chains ending with) intervals, a representation that is enabled through the use of the constraint system based on intervals and Allen's relations. We illustrate this idea through an example that we graphically depict in Figure 3.

For illustrating this point, we represent a buy advice issued by the company Credit Suisse for Philips. We assume the existence of a *Company* class in the ontology. Two instances of this class are created, *iCreditSuisse* and *iPhilips*, denoting the two companies involved. For each of these static individuals we create a timeslice, resulting in *iCreditSuisse_TS1* and *iPhilips_TS1*, respectively. The timeslices are associated to the concept they describe through the *towl:timeSliceOf* property. The *towl:time* property is employed to associate a time interval to each of the timeslices, in this case *iIntervalCS* and *iIntervalPh*, respectively. Both intervals are described in terms of their starting and ending points through the *towl:start* and *towl:end* properties. Finally, we connect the two timeslices through the *advice:buyAdvice* fluent, thus representing the buy advice issued by Credit Suisse for Philips.

Returning to the point of restrictions enabled by the TOWL language - fluents may only connect timeslices that hold over the same interval - through the use of the constraint system we enforce the *towl:equal* Allen relation to hold between the intervals that describe the two timeslices. In the context of TOWL, the Allen relations are defined as binary predicates over the concrete domain, and are employed to specify such restrictions as follows (in DL notation):

$$\exists(time, fluentObjectProperty \circ time).\texttt{equal}$$

Enforcing such an axiom may have two consequences:

- If the intervals do not satisfy the *equal* relation, an inconsistency is discovered;
- If one of the intervals is not defined, its endpoints may be inferred from the other interval.

However, as argued later in this section, the second point does not apply, as we enforce timeslices to always be associated with a timeinterval for which both endpoints are defined.

At this point, is should be mentioned that, building upon the approach in [14], TOWL enables differentiations between fluents that take values from the *TimeSlice* class and fluents that indicate changing values (datatypes). This is achieved through the use of the *FluentObjectProperty* and *FluentDatatypeProperty* properties, and comes to reduce the proliferation of objects in TOWL ontologies due to the fact that, in the case of datatypes, the number of timeslices that need to be created is reduced to half.

A final issue that needs to be addressed relates to the *Frame Problem*. In the current case, this resumes to being able to determine, at any point in time, what holds true and what not. What we are trying to cope with also relates to being able to determine what has not been affected by an action. In Figure 4 we present two concepts, *Concept1* and *Concept2*, each with timeslices across different time intervals. If timepoint *T* is chosen, it becomes obvious that determining what holds true at that time is a simple matter. This is however only possible if we enforce timeslices to always be associated with a time interval that indicates the period across which these timeslices hold true. The definition of a TOWL timeslice, as formalized in the TOWL semantics, satisfies this condition.

4. Stock Recommendations Aggregation System

Stock recommendations, although taking on different denominations, can always be reduced to advices of the form buy/hold/sell. They are issued by large brokerage firms, and mirror the expectations regarding the development of the stock price of the envisioned company. The collection of such recommendations that are true at a given point in

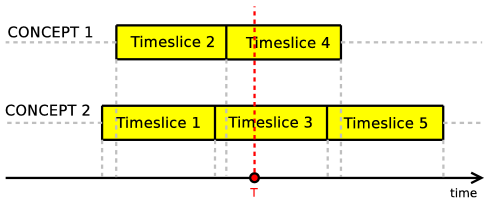

Figure 4. Determining what holds true at a certain point in time.

time is denoted as *market consensus*, and can often be a good indicator of the average expectation regarding the future (within 1 year) value of a company. Roughly, a stock recommendation thus consists of the issuer (the brokerage firm), the targeted company and the type of the advice (buy/hold/sell). Although other characteristics may be included in such a recommendation, such as 12-months price target, 12-months earnings per share, etc., they fall out of scope for the current purpose.

Going beyond pure market consensus, an interesting research question is whether these recommendations can be aggregated into a single advice such that the aggregated advice is a better indicator of how the price of a certain stock is likely to develop. The most intuitive variable to consider in this context is the historical performance of the individual brokerage firms present in the market consensus. This is exactly the goal of the Stock Recommendations Aggregation System (SRAS). This section focuses on outlining the system and, additionally, presenting preliminary results obtained through the aggregation of advices.

4.1. System Architecture

In this section we provide an overview of SRAS in terms of the system's architecture. The individual components are described in more detail in the following sections. A graphical depiction of SRAS is presented in Figure 5. Here, the different modules are grouped according to the system component they belong to.

A distinction may be made between the following components:

- HTML wrappers;
- Information extraction;
- Knowledge base update;
- Advice generation.

A walkthrough the system, at a general level, involves fetching data from the HTML data sources, extracting the

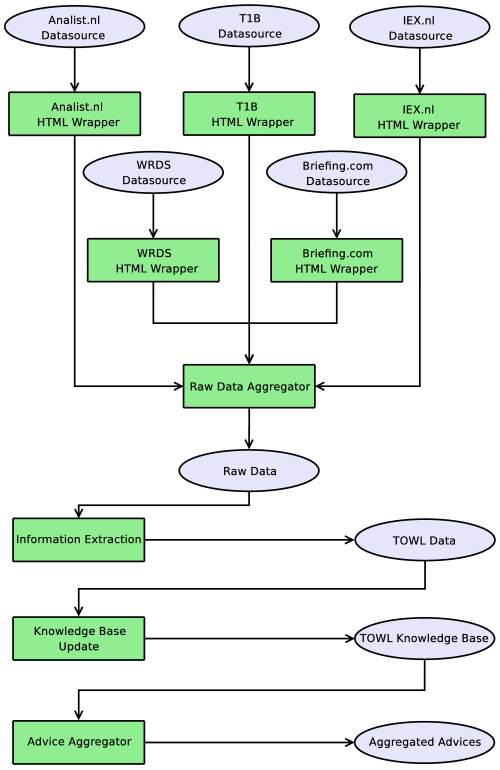

Figure 5. System architecture.

relevant knowledge and updating the TOWL knowledge base. Based hereon, the SRAS advice generation component aggregates the available information and presents a single advice to the end-user (i.e., trader).

4.2. HTML Wrappers

One of the main features of the system consists of the various raw data sources employed for the generation of advices. Additionally, different types of data are employed for the purpose of the system. In this section we describe both the sources as well as the type of data we extract for the current purpose.

The main data ingredient consists of the raw advices, in the form of buy/hold/sell recommendations. Emerging advices are grabbed from the `http://www.analist.nl`, `http://www.iex.nl` and `http://www.briefing.com` websites, which are updated on a daily basis. We focus on a total of 8 markets: Euronext Amsterdam, Euronext Brussels, Euronext Paris, DAX 30, DowJones 30, FTSE 100, NASDAQ 100 and SMI. His-

torical advices are fetched for the purpose of determining the past performance of brokerage companies. A comprehensive resource for this type of data is the Wharton Research Data Center (WRDS), available at `http://wrds.wharton.upenn.edu/`.

Next to data that directly concerns stock recommendations, we collect different characteristics of companies, such as the company codes, and the markets and sectors in which they are active. Thomson One Banker (T1B) provides extensive information for this purpose, together with historical daily closing stock price data for the companies considered by SRAS. The set of company features that we consider consists of attributes such as *company name* and *quote symbol*, but also a series of different codes that uniquely identify companies: such as *ISIN*, *SEDOL*, *CUSIP*a and *IBES*. Additionally, we include the *GICS* code, the is used to identify the industry classification to which a certain company belongs.

The need for being able to identify companies by different codes (ISIN, SEDOL, etc.) emerges from the fact that, in different contexts, the company may be identified by any of these codes (think of the data extracted from HTML advices). Additionally, as all these codes are generally used just as often, we provide different means of identifying a company at application level.

The data described in this section is grabbed from the different data sources through the use of individual HTML wrappers, and fed to the Information Extraction component, which we describe next.

4.3. Information Extraction

The first module of the Information Extraction component is the Stanford Part-of-Speech (POS) tagger [13]. Some of the commonly used tags that we found in the description of stock recommendations include: /NNP (proper noun, singular), /IN (preposition or subordinating conjunction), /CD (cardinal number), /NN (noun, singular or mass), /DT (determiner), /JJ (adjective) and/VBZ (verb, 3rd person singular present). The individual recommendations, as provided by the HTML wrapper, are annotated word by word based on the part of speech they represent.

After having annotated the textual data by means of the Stanford POS tagger, the tagged information is further processed. The relevant knowledge is extracted by means of pattern-based extraction. In the case of stock recommendations, the goal of this phase is to identify the following features:

- Issuer: the brokerage firm that issued the recommendation;
- Company: the company for which the recommendation was issued;
- Advice type: the type of the advice, one of buy/hold/sell;
- Identifier: unique code of the company for which the advice was issued;
- Issue date: the date when the advice was issued.

Regarding the time associated with the validity of the advice, a sidenote is in place. At the time an advice is issued, it is not known for how long this recommendation will hold true. In other words, the recommendation might expire with no special events occurring, or a new recommendation may be issued. For this goals, at the time the advice is created by SRAS, a default validity period of 12 months is assumed. In the case that during this period a new advice is issued by the same brokerage firm for the same company the length of the old advice is adjusted so that it ends right before the new one was issued. This temporal feature of the system will further be detailed when discussing the Knowledge Base Update component.

After having tagged and extracted the relevant knowledge from the raw input, this data is passed on to the Knowledge Base Update module. The main purpose of this module is the creation of TOWL instances that reflect the relevant knowledge extracted from the various data sources. In the case of recommendations, timeslices of the brokerage firm and the company are created with the corresponding time intervals. The timeslices are connected with the appropriate fluent (e.g., *buyAdvice* in the case of a buy advice). The thus created data is passed on to the information to the Knowledge Base Update component.

4.4. Knowledge Base Update

The knowledge base (KB) employed by SRAS is populated with static information once before the fetching of advices is initiated. All static information regarding companies, codes and exchanges is present, although the process of populating the KB with this knowledge may be repeated on user demand. The high quantity and very low rate of change of this data make it unfeasible to initiate this process each time new advices are grabbed.

The main task of the component discussed in this section thus consists of updating the KB with the newly issued stock recommendations, as received from the Information Extraction component. This task is achieved by employing Jena [9] - a Semantic Web framework designed for the manipulation of ontologies. In combination with the rule-based inference engine provided by Jena, it becomes possible to check for temporal constraints and inconsistencies, in addition to the static ones. This can be achieved through customizing the Jena engine by an extension that includes temporal rules.

Figure 6 shows an example of a TOWL Jena rule, i.e., the rule used to check if an instance is inside a certain interval. This rule uses the `before` predicate to verify whether the starting point of an interval is before a given instance and whether the instance is before the ending point of the interval. The `before` predicate is defined in another TOWL Jena rule which uses the `lessThan` function (available in Jena for comparing *xsd:dateTime* expressions).

```
# inside
[inside:
(?ins rdf:type time:InstantThing)
(?int rdf:type time:IntervalThing)
(?int time:start ?intstart)
(?int time:end ?intend)
(?ins time:before ?intend)
(?intstart time:before ?ins)
->
(?ins time:inside ?int)]
```

Figure 6. TOWL Jena rule example.

An interesting feature at this stage is the adjustment of the length of the advices based on the current context. As already mentioned, the duration of advices is not known at the time the advice is issued. However, TOWL enforces all timeslices to be described by a defined interval, and thus an ending point for the interval (i.e., the recommendation) is necessary. The temporal rule we employ for this purpose is initially based on adding a default duration of 12 months to each newly issued advice. If, during the update of the KB, an advice is discovered that is identical to the advice that is being added in terms of issuer and target company, the endpoint of the old advice is adjusted. We assume of course that the advice, though old, is not older than 12 months. The new ending date of the interval associated with the old advice is set to one day before the new advice was issued, and the new advice may now be added to the KB without affecting the consistency of the old advice.

After each update of the KB, the system generates new aggregated advices for the company where changes are detected following the process described in this section. More light is shed on the generation of aggregated advices in the next section.

4.5. Aggregated Advice Generation

The focus of this section is on the core module of the Stock Recommendations Aggregation System, namely the Advice Aggregator. The computation of the actual aggregated advices is based, as previously mentioned, on the individual performance of the brokerage firms present in the market consensus for which the system generates a recommendation, at the time this recommendation is calculated. We base this calculation on the Performance Index (PI) of each broker, in turn based on the profitability (generated return) of the broker. This is calculated based on the performance of the individual advices issued by the latter. However, due to the varying duration of advices, absolute return proves to be a subjective measure of performance. For this reason, we rely on the average daily return.

We first introduce the daily return of an advice, calculated according to the following formula:

$$R_t^{daily} = \frac{price_t - price_{t-1}}{price_{t-1}}$$

Based on the daily return, the average return of an advice is calculated according to the following formula, where n is the total number of days across which the advice was true:

$$R_{advice} = \frac{\sum_{t=1}^{n} R_t^{daily}}{n}$$

Finally, the performance index of a brokerage firm is calculated as the average return that all its advices generated:

$$PI_{broker} = \frac{\sum_{advice=1}^{m} R_{advice}}{m},$$

where m is the total number of advices issued by a single broker.

It should be noted that the range of advices considered when calculating the performance index may vary, according to user preferences. One obvious option is selecting all recommendations that the brokerage firm issued. Another option however, is to narrow the set of considered advices to those issued for companies active in the same industry as the company for which an advice is generated by SRAS. By relying on the GICS code for this task, the field of advices may be varied based on the selected length of this code.

The next step in the generation of an advice is determining the confidence factors for each type of possible recommendations (buy/hold/sell), based on the performance indexes. For each of type of recommendation, the confidence factor $CF^{[buy,hold,sell]}$ is calculated as follows, provided at least one performance index is positive:

$$\frac{\sum_{i=1}^{p} PI_i^{[buy,hold,sell]}}{\sum_{j=1}^{q} PI_j^{buy} + \sum_{k=1}^{r} PI_k^{hold} + \sum_{l=1}^{s} PI_l^{sell}}$$

where $PI^{[buy,hold,sell]}$ represents the performance index of the broker that issued a $[buy, hold, sell]$ advice, p represents the total number of brokerage firms in the market consensus, and q, r, s the number of brokerage firms in the market consensus that issue a buy, hold or sell advice, respectively, such that $p = q + r + s$.

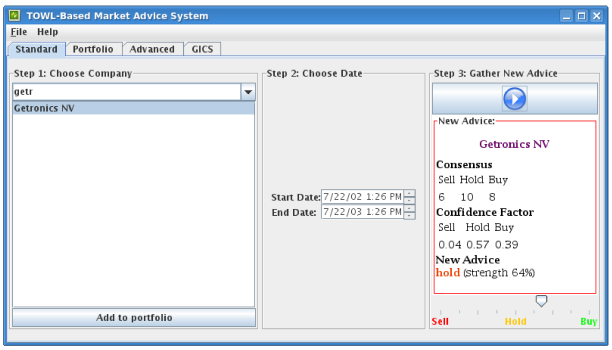

Figure 7. The standard tab.

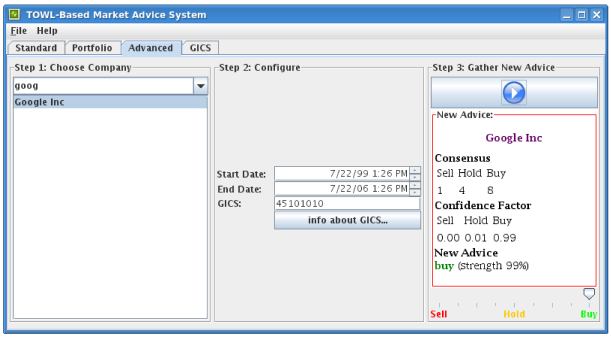

Figure 8. The advanced tab.

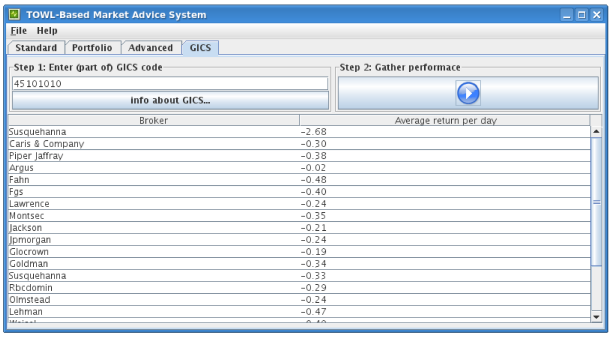

Figure 9. The GICS tab.

In case all advices are negative, the confidence factors are calculated as follows:

$$1 - \frac{|\sum_{i=1}^{p} PI_i^{[buy,hold,sell]}|}{|\sum_{j=1}^{q} PI_j^{buy} + \sum_{k=1}^{r} PI_k^{hold} + \sum_{l=1}^{s} PI_l^{sell}|}$$

This is done so that, despite the negative values, a proper scaling may still be achieved. In this fashion, confidence factors may be determined regardless of the nature (positive/negative) of the performance indexes, while maintaining the ability to compare results.

Having calculated a confidence factor for each type of recommendation based on market consensus and historical information, the type of advice issued by the application is the one with the highest confidence factor after performing the calculations presented in this section.

Additionally, we calculate a strength, S_{adv}, for the advice issued by the application, based on the individual confidence factors. This is calculated as:

$$\begin{aligned} S_{adv} \quad = \quad & (max(CF^{buy}, CF^{hold}, CF^{sell}) - \\ & -min(CF^{buy}, CF^{hold}, CF^{sell}) + \\ & +max(CF^{buy}, CF^{hold}, CF^{sell}) - \\ & -median(CF^{buy}, CF^{hold}, CF^{sell}))/2 \end{aligned}$$

The intuition behind this formula relates to regarding the strength of advices in terms of whether there is a clear discrimination between the confidence factors of the different advice types. Additionally, this 'distance' is quantified, thus providing the strength of the advice.

4.6. User Interface & Preliminary Results

The user interface is divided into 4 tabs, as presented in Figure 7, where the default tab - Standard - is selected.

In this tab the user can select a company by typing anything from the company name to one of the available company identifiers (codes). Once selected, a period to be considered when calculating the aggregated advice may be selected. Finally, upon clicking the *Gather New Advice* button, an aggregated advice is generated following the methodology as outlined in the previous section.

The second tab, *Portfolio*, is identical to the *Standard* tab, with the exception that here the user may select a whole range of companies rather than just one, and receive advices for each selected company.

The *Advanced* tab allows the user to select what length of the GICS code of the selected company is going to be used when calculating the aggregated recommendation. As shown in Figure 8, the GICS code of the selected company is already shown, and can be edited by the user. Additionally, information on the GICS code is available, at different lengths of this code.

Finally, the *GICS* tab presents the average performance of brokerage firms within a specific GICS industry. As shown in Figure 9, the GICS code is editable, and any length (maximum 8) may be entered when performing this benchmark.

The preliminary results present a number of interesting features. Perhaps the most striking one relates to the fact that the aggregated recommendation generated by SRAS

does not always follow the market consensus. In other words, a recommendations distribution (across buy, hold and sell) that has a unique maximum (say *buy*), does not always agree with the advice generated by the application. This occurs especially when taking into account the performance of the brokerage firms within the specific industry where the company for which the advice is generated is active. This could be an indicator that taking into account historical performance of brokerage firms leads to the creation of new knowledge regarding the most likely development of a company's share price.

5. Conclusion and Future Work

The TOWL language is an extension of OWL-DL that enables the representation of time and temporal aspects such as change. It comes to meet shortcomings of previous approaches, such as [5, 14] that only address this issue to a limited extent and don't seek to enable automated reasoning in a temporal context.

The approach presented in [5] for example only deals with the representation of time in the form of intervals and instants. However, ensuring that intervals are properly defined (starting point is always strictly smaller than the ending point) is not possible in this approach. The approach taken in [14] builds upon [5] by addressing one of its limitations, namely: the representation of temporal aspects such as change. However, the semantics underlying this approach are still OWL-DL semantics, and thus no reasoning support is provided for representations based on timeslices and fluents other than the standard (and insufficient) OWL-DL reasoning.

The TOWL language has been developed as an extension of OWL-DL to address a need that initiated at a practical (application) level. Developing financial applications in a Semantic Web context requires an increased level of context awareness and thus support for the representation of highly dynamic data. One such application is the Stock Recommendations Aggregation System that we present in this paper. Here, the temporal data consists of emerging advices that must be represented consistently in the knowledge base. Enabling the aggregation of recommendation data and other financial information results in a powerful application able to issue single advices, of different strengths. The preliminary results show that the informational content of this aggregated knowledge is increased when compared to the individual advices.

Acknowledgement

The authors are partially supported by the EU funded IST STREP Project FP6 - 26896: Time-determined ontology-based information system for realtime stock market analysis. More information is available on the official website[1] of the project.

References

[1] J. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, 1983.

[2] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, 284(5):28–37, 2001.

[3] Economist. Algorithmic trading: ahead of the tape. *The Economist*, (383):85, 2007.

[4] C. Gutierrez, C. Hurtado, and A. Vaisman. Introducing time into RDF. *IEEE Transactions on Knowledge and Data Engineering*, 19(2):207–218, 2007.

[5] J. Hobbs and F. Pan. An ontology of time for the semantic web. *ACM Transactions on Asian Language Information Processing*, 3(1):66–85, 2004.

[6] D. Lewis. Counterparts of Persons and Their Bodies. *The Journal of Philosophy*, 68(7):203–211, 1971.

[7] C. Lutz and M. Milicic. A tableau algorithm for description logics with concrete domains and general tboxes. *Journal of Automated Reasoning*, 38(1–3):227–259, 2007.

[8] C. Lutz and M. Milicic. A tableau algorithm for description logics with concrete domains and general tboxes. *Journal of Automated Reasoning*, 38(1–3):227–259, 2007.

[9] B. McBride. Jena: Implementing the RDF model and syntax specification. In *Proceedings of the Second International Workshop on the Semantic Web (SemWeb 2001)*, 2001.

[10] A. Micu, L. Mast, V. Milea, F. Frasincar, and U. Kaymak. *Financial news analysis using a semantic web approach*. Semantic Knowledge Management: an Ontology-based Framework. Idea Group, 2008. To appear.

[11] V. Milea, F. Frasincar, U. Kaymak, and T. di Noia. An OWL-based approach towards representing time in web information systems. In *The 4th International Workshop of Web Information Systems Modeling Workshop (WISM 2007)*, pages 791–802. Tapir Academic Press, 2007.

[12] T. O'Reilly. What is web 2.0: design patterns and business models for the next generation of software. `http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html`.

[13] K. Toutanova and C. Manning. Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC 2000)*, pages 63–70, 2000.

[14] C. Welty and R. Fikes. A reusable ontology for fluents in OWL. In *Formal Ontology in Information Systems: Proceedings of the Fourth International Conference (FOIS 2006)*, pages 226–336. IOS Press, 2006.

[15] H. Zhu, S. E. Madnick, and M. Siegel. Reasoning about temporal context using ontology and abductive constraint logic programming. In *Principles and Practice of Semantic Web Reasoning, Second International Workshop (PPSWR 2004)*, volume 3208 of *Lecture Notes in Computer Science*, pages 90–101. Springer, 2004.

[1] http://www.towl.org