A Lexico-Semantic Pattern Language for Learning Ontology Instances from Text

Wouter IJntema, Jordy Sangers, Frederik Hogenboom, Flavius Frasincar*

Erasmus University Rotterdam, P.O. Box 1738, NL-3000 DR, Rotterdam, The Netherlands

Abstract

The Semantic Web aims to extend the World Wide Web with a layer of semantic information, so that it is understandable not only by humans, but also by computers. At its core, the Semantic Web consists of ontologies that describe the meaning of concepts in a certain domain or across domains. The domain ontologies are mostly created and maintained by domain experts using manual, time-intensive processes. In this paper, we propose a rule-based method for learning ontology instances from text that helps domain experts with the ontology population process. In this method we define a lexico-semantic pattern language that, in addition to the lexical and syntactical information present in lexico-syntactic rules, also makes use of semantic information. We show that the lexico-semantic patterns are superior to lexico-syntactic patterns with respect to efficiency and effectivity. When applied to event relation recognition in text-based news items in the domains of finance and politics using Hermes, an ontology-driven news personalization service, our approach has a precision and recall of approximately 80% and 70%, respectively.

Keywords: lexico-semantic patterns, information extraction, ontology learning, Semantic Web

1. Introduction

In today's information-driven world, many individuals try to keep up-to-date with the latest developments by reading news items on the Web. The contents of news items reflect past, current, and future world conditions, and thus news contains information valuable for various purposes. For example, being aware of current market situations is of paramount importance for investors and traders, who need to make informed decisions that could have a significant impact on certain aspects such as profits and market position. However, due to the ever increasing amount of information, it is virtually impossible to keep track of all emerging relevant news in an orderly fashion [1, 2]. Hence, automatically filtering news items by means of computers would alleviate the cumbersome task of manually selecting relevant news messages and extracting information.

In contrast to human beings, machines (e.g., computers) are merely able to read news articles, not to understand them. With the Semantic Web [3], i.e., a collection of technologies that express and reason with content metadata, the World Wide Web Consortium (W3C) provides a framework to add a layer of semantic information to the Web, thereby offering means to help machines understand human-created data (e.g., news messages) on the Web. On the Semantic Web, metadata is defined using semantic information that is usually captured in ontologies, which are defined as shared formal specifications of conceptualizations [4]. Some of the most popular formats to describe ontologies on the Semantic Web are the Resource Description Framework (RDF) and RDF Schema [5, 6], and the Web Ontology Language (OWL) [7]. Ontologies can be used to store domain-specific knowledge in the form of concepts (i.e., classes or instances), together with associated inter-concept relations. These relations are denoted by triples that consist of a subject, a predicate, and an object.

*Corresponding author; tel: +31 (0)10 408 1340; fax: +31 (0)10 408 9162

Email addresses: wouterijntema@gmail.com (Wouter IJntema), jordysangers@hotmail.com (Jordy Sangers), fhogenboom@ese.eur.nl (Frederik Hogenboom), frasincar@ese.eur.nl (Flavius Frasincar)

Most of the current approaches to news filtering, such as, for example, the SeAN [8], YourNews [9], and NewsDude [10] frameworks, are able to retrieve only the news items that contain terms of the user's interest, not taking into account indirect information, which is also deemed relevant, such as competitors of companies of interest, political parties of politicians, etc. Exploiting the semantic contextual information related to concepts of interest enables a more comprehensive overview of relevant news with respect to certain topics. Therefore, in previous work [11, 12], we introduced the Hermes framework, which provides a method for personalizing news items that makes use of semantics. The framework stores lexicalized domain concepts and relations (i.e., properties that relate concepts to each other or concepts to data types) in an ontology. Hence, Hermes stores synonyms or string representations of domain-specific entities (e.g., companies, persons, etc.) and their relations (e.g., subsidiary, competitor, etc.). The ontology is used for retrieving relevant news items in a semantically-enhanced way. In addition to this, we have proposed an ontology-based recommendation method that also benefits from a domain ontology [13]. As adding new information to an arbitrary but sufficiently large knowledge base requires a domain expert to invest a lot of time, in this paper we propose a method that discovers new information automatically.

Automatic information discovery requires the use of information extraction techniques. In the last decades, a vast amount of research has already been conducted in this area. In general, information extraction can be done by means of statistics [14, 15, 16] or pattern-based rules [17, 18, 19], each method having its own benefits and drawbacks. Statistical methods are mainly data-intensive, while pattern-based approaches usually are driven by knowledge more than data. From a user's point of view, large amounts of data are not always readily available, while (general) domain knowledge is usually at hand. As pattern-based approaches often require less training data than statistical methods, and also help users to gain more insight into why a certain relation was found, in this paper we focus on pattern-based information extraction techniques. Pattern discovery, such as the semantic patterns [20, 21] from OntoEdit [22] or the patterns from the Pattern-based Annotation through Knowledge on the Web (PANKOW) framework [23] are outside of the scope of this paper.

The main contribution of this paper is a rule-based language that uses *lexico-semantic patterns* for information extraction. In contrast to *lexico-syntactic patterns* [17, 18, 24], which combine lexical representations (i.e., strings) and syntactical information (e.g., parts-of-speech), lexico-semantic patterns also allow for the usage of semantic information such as concepts that are defined in ontologies. The notion of lexico-semantic patterns has already been introduced in previous work. In [12, 25] we extend the Hermes news processing framework by adding triple-based lexico-semantic event rules that make use of ontological concepts, in order to be able to recognize economic events. After validation, these events are subsequently coupled to the execution of action rules which update the underlying ontology. The use of lexico-semantic patterns for financial events discovery has also been discussed in [26]. There, we present a rule engine that allows for pattern creation based on the triple paradigm (i.e., it makes use of a subject, a predicate, and an optional object), and that relies on triple conversion to the Java Annotations Pattern Engine (JAPE) language [27] and SPARQL [28]. Finally, in [29] we present a semantics-based information extraction pipeline for economic event detection, which makes use of lexico-semantic patterns that are defined in the JAPE language.

In the previous work discussed above, we consider mostly simple lexico-semantic patterns that are merely based on the triple paradigm, which hence makes it impossible to express more complex constructions. In this paper, we present a more expressive language for specifying lexico-semantic patterns that makes use of regular expressions over ontology concepts. Furthermore, in our current endeavors, we aim for a simple, easy to use language for pattern creators. Existing languages like JAPE could easily result in verbose rules, while we aim for more compact ones. In addition, we give the formal specifications of our language and explain its constructs by means of examples, and we give a more extensive evaluation of the proposed pattern language in which we analyze the recognition of different types of events in textual representations.

By using lexico-semantic patterns that employ concepts and relations from a domain ontology, we aim to solve problems caused by ambiguity and specificity that exist in current approaches that employ lexico-syntactic patterns. The design of the lexico-semantic pattern language aims to fulfill the following requirements. First, the language should be developed for a Semantic Web context, where instances and their relations need to be learned from text. Then, the language should be accessible and easy to understand, yet expressive enough to be able to cover the required information extraction needs. By employing Semantic Web technologies, our language should remove some of the ambiguities inherent to lexical approaches, increasing the specifications precision level. In addition, a semantic approach allows to easily specify patterns that have many instances, increasing the recall of the information extraction process.

In this work, we aim to investigate the performance of lexico-semantic patterns compared to lexico-syntactic ones and for that, we evaluate the performance of both pattern languages by creating rules for each one of them that are subsequently applied on two distinct corpora consisting of news messages on financial topics and political topics, respectively. Additionally, we compare the performance of the languages with lexico-semantic patterns written in JAPE. Performance is measured in terms of construction times (i.e., efficiency) and precision, recall, and F_1 scores (i.e., effectivity).

The rest of this paper is organized as follows. Section 2 discusses the related work, followed by Sect. 3, which elaborates on the Hermes Information Extraction Language (HIEL), i.e., the syntax for defining lexico-semantic patterns. Section 4 describes the Hermes Information Extraction Engine (HIEE), after which we evaluate our method in Sect. 5. Section 6 gives our conclusions and identifies future work.

2. Related Work

In the current body of literature, various pattern grammars are described that could be of use in for instance news processing frameworks [30, 31] or general purpose information extraction tools [32, 33, 34, 35, 36]. These patterns are based on linguistic or lexical knowledge, as well as a priori human knowledge regarding the contents or topic of the text that is to be processed. We can make a rough distinction between two types of patterns that can be applied to natural language corpora, i.e., lexico-syntactic patterns and lexico-semantic patterns. The former patterns are a combination of lexical representations and syntactical information, whereas the latter patterns combine lexical representations with syntactic and semantic information.

2.1. Lexico-Syntactic Patterns

Hearst [17, 18] proposes the use of lexico-syntactic patterns for information extraction. This approach aims to find hyponym and hypernym relations by discovering regular expression patterns in free text. An example is the application of the following pattern to the sentence "*... works by such authors as Herrick, Goldsmith, and Shakespeare*":

```
such NP as {NP,}* {(or|and)} NP            (Rule 1)
```

In this pattern, "`NP`" indicates a proper noun. Other text (i.e., "`such`", "`as`", "`or`", and "`and`") is used for lexical matching, while "`(`" and "`)`" contain conjunction and disjunction statements to be evaluated, in this case a disjunction (denoted as "`|`"). Also, "`*`" is a repetition parameter that indicates the sequence between braces ("`{`" and "`}`") is allowed to repeat zero to an infinite number of times. The rule presented above results in the following discovered relationships:

```
hyponym("author", "Herrick")
hyponym("author", "Goldsmith")
hyponym("author", "Shakespeare")
```

These patterns are often easy to comprehend by regular users, yet defining the right patterns to mine corpora to obtain unknown information is not a trivial task. Hearst stresses that, in order to return desired results successfully, patterns should be defined in such a way that they occur frequently and in many text genres. Also, they should often indicate the relation of interest and should be recognizable with little or no pre-encoded knowledge. Furthermore, all existing syntactic variations have to be included into a complex pattern to ensure its proper working.

2.2. Lexico-Semantic Patterns

Lexico-semantic patterns on the other hand are less cumbersome to define, as they make use of concepts instead of merely lexical representations, hereby alleviating the time-consuming process of pattern definition. One of the first works introducing lexico-semantic patterns is [19], where the authors propose a system that processes text prior to normal left-to-right syntactic parsing. The patterns may include terms and operators like lexical features, logical combinations, wildcards, and repetition, which are mostly adopted from the regular expression language. An example of a rule that will classify the verb phrase "*left dead*" as to express death or injury, is as follows:

```
?PIVOT=(or found left shot)                (Rule 2)
?OBJ  =* ?EFFECT=dead
      => (mark-activator
          murder d-vp) ;
```

This sentence would also match "*found dead*" and "*shot dead*". Next to standard elements such as repetition and wildcards, the rule presented here contains features like variable assignment on the left-hand side (LHS) (where words preceded by "`?`" denote variables) and on the right-hand side (RHS) macros such as "`mark-activator`", which uses the results of the pattern match, including variable assignments, along with some other constants, such as "`murder`" and "`d-vp`", to tag and segment the text. The main advantage of such lexico-semantic patterns is that they take into account the domain semantics which help the parser cope with the complexity and flexibility of real text [19].

The Conceptual Annotations for Facts, Events, Terms, Individual Entities, and RElations (CAFETIERE) framework as introduced by Black et al. [33] is a rule-based system for ontology-driven text mining, which makes use of lexico-semantic patterns. CAFETIERE applies several preprocessing techniques to the text, i.e., tokenization, Part-Of-Speech (POS) tagging, and gazetteer lookup. To extract information from text, a rule notation is defined. A rule has the following form:

```
A => B \ C / D                             (Rule 3)
```

where "`A`" represents the phrase that is recognized, "`B`" (optional) represents the text prior to "`C`", "`C`" defines the text elements that are part of the phrase, and "`D`" (optional) is the neighboring text immediately following "`C`". A basic example of a rule that would match an expression like "*40 mg*" is:

```
[syn=NP, sem=QTY] =>                       (Rule 4)
    \[syn=CD], [sem=measure]/;
```

In this pattern, one is able to denote the characteristics of a matching token group, i.e., its syntactic category (i.e., a noun) and its semantic meaning (i.e., a quantity). In order to match an expression, the text should contain a token which is a cardinal digit, followed by a token that represents a measure. CAFETIERE also takes into account the ordering of the rules. When one rule matches the text and annotates the text, the original annotation might no longer be visible to the next rule.

Another information extraction rule language that includes domain semantics is WHISK [37]. This language is based on regular expressions and can be used for extracting information from semi-structured text as well as free text. An example of a rule that extracts the number of bedrooms and the associated price for a rental ad is written as such:

```
Pattern:: * ( Digit ) ‘BR’ * ‘$’           (Rule 5)
          ( Number )
Output::  Rental
          {Bedrooms $1}
          {Price $2}
```

Whenever the pattern of the extraction rule matches a sentence, the syntactical elements that are enclosed by round brackets are being used as variables in the output statement. The first element "`Digit`" is assigned to "`$1`" and the second element "`Number`" is assigned to "`$2`". In WHISK rules, the "`*`" symbol represents a wildcard, i.e., it is used to indicate an arbitrary sequence of characters without limitations to size and contents until the occurrence of the subsequent term in the pattern.

In [38, 39, 40] a MUlti-Source Entity recognition system (MUSE) is proposed. This system employs the General Architecture for Text Engineering (GATE) [41] software, which is a Java-based environment supporting the research and development of language processing software, in order to extract information from text. The main focus is on the extraction of information from multiple sources and retain a certain robustness. The system consists of a number of components, including a tokenizer, gazetteer, sentence splitter, POS tagger, semantic tagger, and an orthographical matcher. The semantic tagging comprises a set of grammar rules based on the Java Annotations Pattern Engine (JAPE) language [27]. An example of such a rule is:

```
Rule: GazLocation                              (Rule 6)
(
    {Lookup.majorType == location}
)
:loc --> :loc.Location =
    {kind = "unknown",
     rule = "GazLocation"}
```

In general, the LHS contains the pattern to be matched, whereas the RHS defines the action that is to be executed once a match has been found. This rule is fired (executed) when the gazetteer lookup results in a location. If this is the case, the pattern will be annotated with the type "*Location*" and two attributes, "*kind*" and "*rule*".

2.3. Our Contribution

The pattern language proposed in this paper differs with respect to several aspects from the languages presented above. The lexico-syntactic patterns proposed by Hearst [17, 18] are often easy to comprehend by regular users, which is also one of our goals when designing our pattern language. However, Hearst's patterns do not capture the semantic context of the text, while our approach aims for a semantic description of the context.

The lexico-semantic pattern language proposed by the authors of [19] is similar to ours, since it also employs patterns for detecting semantics in text. Their framework is implemented in the GE NLToolset [42], which is a set of text interpretation tools. In our framework we benefit from the natural language processing steps performed by GATE [41] and the underlying OWL ontologies. The software allows for easy extension and customization, in contrast to the GE NLToolset. In addition, we propose patterns that are easier to specify and comprehend by the end user than the patterns proposed in [19].

In order to maintain readability, we aim for a notation similar to the one presented in [17, 18]. Even though our patterns add semantic functionalities, they strictly adhere to the standard POS tags [43] (in contrast to the patterns used in [19]). Furthermore, our patterns require less keywords compared to the ones proposed in [19], as they omit mark and pattern activators. Our intent is to explore the possibilities of adding semantics to the patterns by using Semantic Web technologies, and thus to make use of existing ontologies and support tools (e.g., reasoners, editors, readers, writers, etc.).

In our work, we benefit from the research that has been done in the CAFETIERE project, e.g., by reusing parts of the rule notation. A limitation within the CAFETIERE framework is that rules are defined on a specific lexico-semantic level, i.e., semantic concepts are derived from an ontology (knowledge base) described in Narrative Knowledge Representation Language (NKRL) [44]. NKRL is a knowledge representation language which has been defined before the Semantic Web era, and has no formal semantics. Hence, the approach fails to properly describe domain semantics. Both the gazetteer and the lexico-semantic rules could benefit from an ontology-based approach, abstracting from the low-level and sometimes ambiguous lexical representations.

Even though WHISK [37] does properly include domain semantics, the applicability of the language is limited. The support for wildcards creates flexibility in the patterns to be matched, but it is fairly restrained compared to for instance regular expressions. It is not possible to state a specific range of characters or words. Differently than WHISK, our language contains additional repetition operators, so that more expressive extraction rules can be created.

Similar to our approach, MUSE [38, 39, 40] employs GATE. Our work distinguishes itself from this approach by proposing a language with a higher level of abstraction, which is easier to read for regular users. In addition to that, we focus on semantic patterns and aim to determine relations between concepts, rather than solely focusing on recognizing entities.

3. Hermes Information Extraction Language

The Hermes Information Extraction Language (HIEL) employs semantic concepts from an ontology. The language is evaluated in the context of extracting events and relations from news, as an extension to the existing Hermes news personalization framework [11, 12]. This section continues by briefly explaining the characteristics of the Hermes framework as well as the usage of ontologies within the framework in Subsect. 3.1. Subsequently, Subsect. 3.2 introduces HIEL for semi-automatic information extraction from news items, of which the Extended Backus Naur Form (EBNF) grammar is given in Appendix A. Last, Subsect. 3.3 elaborates on the usage of ontology elements within our language.

3.1. Hermes

Hermes [11, 12] is a framework that can be used for building a personalized news service. The framework enables users to select concepts from a knowledge base. Whenever these concepts, which could be instances like *Microsoft* or *Google*, or related concepts, such as competitors, appear in an arbitrary news item, the news item is presented to the user. Hence, the user will only be presented news items that match the user's interest.

Concept selection is done by means of user-defined patterns. Similarly to CAFETIERE, Hermes is based on GATE and employs lexico-semantic patterns. However, these patterns use information from an OWL ontology that contains a schema of concepts and relations of various nature, thus making use of a standard language supported by many reasoners. Knowledge is stored in a separate ontological database that contains instances. Each time a news message is processed, the ontology might be updated with new facts, so that the knowledge base remains up-to-date [12].

The current knowledge base of Hermes is maintained by a manual approach. The domain ontologies are developed by domain experts. The process of developing the ontology is an incremental middle-out approach [11]. Since news events can change the state of the world, each time such a change happens, the knowledge base should be updated. Because updating the ontology manually is a cumbersome process, it is preferred to do this at least semi-automatically. Therefore, we propose an information extraction language that can extract new instances of concepts and relations from news items.

3.2. Language Syntax

The patterns previously proposed by Hearst [17, 18] serve as an inspiration for HIEL, as these lexico-syntactic patterns are easily comprehensible. Furthermore, these patterns provide the user with valuable insights into the reasons behind the extraction of certain information. Therefore, we aim to propose a language that approaches this simplicity, i.e., a language with which one is able to make patterns that are intuitive and easy to understand, but which also addresses the required expressivity. In this regard, it should have at least the expressivity of regular expressions. Our language can be characterized by supporting syntactic features, orthographic features, concepts, relations between concepts, logical operators, repetition, and wildcards. In this subsection, we explore the syntax of the language.

3.2.1. Language Definition

Typically, in HIEL, each pattern is described by a left-hand side (LHS) and a right-hand side (RHS). Once the RHS has been matched in the text to be processed, it is annotated as described by the LHS of the pattern. The LHS describes a relation between a *subject* (`sub`) and an *object* (`obj`) by using a *predicate* (`pred`). For example, *IsCompetitorOf* is a relation between the concepts *Microsoft* and *Google*. We denote the LHS of a pattern as follows:

```
(sub, pred, obj) :- RHS
```
(Rule 7)

The RHS on the other hand describes a pattern that has to be identified in text. We define a pattern as an ordered collection of tokens that are divided by spaces, which indicates the sequence in which the target tokens have to appear in text. The RHS of a HIEL pattern is not limited to one sentence, but is matched against the full news article text. In order to limit a rule to a sentence, one has to specifically define this constraint in the pattern.

3.2.2. Literals

As shown in Sect. 2, pattern grammars typically support literals, i.e., text strings. Literals can be written as a (compound) word surrounded by quotes, e.g., "*John F. Kennedy*". In HIEL, tokens on the RHS of patterns can be of various types, amongst which literals. Whenever literals are used within patterns, the (compound) word between quotes has to match exactly with the text.

3.2.3. Lexical Category

Like many other lexico-syntactic and lexico-semantic pattern languages, our language supports a set of syntactic categories to describe the lexical category of the token, i.e., its part of speech. The possible values of the lexical category are shown in Table 1. In general, we distinguish between various verbs and nouns, prepositions, adjectives, coordinating conjunctions (e.g., "*as well as*"), cardinal numbers, and interjections (e.g., "*well*" as in "*well, that depends*").

Category	Description
CC	Coordinating conjunction
CD	Cardinal number
IN	Preposition
JJ	Adjective
NN	Noun
NNP	Proper Noun
PP	Pronoun
RB	Adverb
UH	Interjection
VB	Verb, base form
VBZ	Verb, 3rd person singular present

Table 1: Common lexical categories

3.2.4. Orthographic Category

In addition to the word lexical category, the language distinguishes four orthographic categories. Note that the field of orthography spans hyphenation, capitalization, word breaks, emphasis, and punctuation. We define orthography as describing (defining) the set of symbols used in tokens. More specifically, we focus on capitalization. The `upperInitial` category is used for tokens that start with an uppercase character. When referring to capitalized words, `allCaps` should be used. In addition, `lowerCase` indicates a token without uppercase characters. Finally, `mixedCaps` is used in words with varying capitalization. Orthographic categories can especially be useful when identifying names or abbreviations.

3.2.5. Labels

The subject, relation, and object described in the LHS need to be identified in the RHS in order to provide a link between text and a new extracted fact. This can be done using labels, which are represented as words preceded by a "`$`" and followed by a colon and an equality sign, as well as a description of the attached token. Whenever the RHS matches with a sentence, the tokens with associated labels are filled in the LHS of the rule. An example rule with labels is:

```
($sub, kb:hasProduct, $obj) :-                (Rule 8)
    $sub:='Google' 'launches'
    $obj:=upperInitial
```

Note that "`kb:`" represents a namespace, which in our case refers to a knowledge base (ontology) in which the predicate "`hasProduct`" has been specified.

3.2.6. Logical Operators

The language supports three of the most common types of logical operators as defined in [45], i.e., and ("`&`"), or ("`|`"), and not ("`!`"). The disjunction and conjunction are used in combination with grouping parentheses in the RHS. An example of such a rule is:

```
($sub, rdf:typeOf, $obj) :-                   (Rule 9)
    $sub:=(NN & upperInitial)
    $obj:=(NN | CD)
```

Here, "`rdf:`" points to the namespace of RDF, which – amongst others – contains the "`typeOf`" property. Negation can be used almost everywhere in the RHS of a rule, except in front of a label, e.g.:

```
($sub, rdf:typeOf, $obj) :-                   (Rule 10)
    $sub:=(!NN)
    $obj:=(!(NN | CD))
```

3.2.7. Repetition

Another feature that is often used in many languages is repetition, which is employed as an indication that a certain pattern can be found a number of times. In HIEL, we distinguish between four types of repetition operators: zero or more ("`*`"), once or more ("`+`"), zero or once ("`?`"), and a range ("`{min[,[max]]}`"). The latter indicates that the foregoing pattern must occur at least `min` times and no more than `max` times. The comma and the maximum are optional. When a maximum has not been defined, the pattern must occur at least `min` times. Leaving out the comma as well indicates that the specified pattern must occur exactly `min` times. An example of a rule utilizing a range operator is:

```
($sub, rdf:typeOf, $obj) :-                   (Rule 11)
    $sub:=NNP (VBZ | NN){1,3}
    $obj:=NNP
```

3.2.8. Wildcards

The patterns defined in the RHS of rules can be very specific. The order of the tokens is fixed and no other words between the tokens are allowed. In order to enable some flexibility in patterns, we allow the user to employ wildcards. These wildcards can be used to state that any word may be found in the text and are inspired by the wildcards of the database query language SQL. Within our language, the user is allowed to specify that zero or more words may be skipped ("%") or exactly one word may be skipped ("_"). An example rule that makes use of wildcards is:

```
($sub, rdf:typeOf, $obj) :-                (Rule 12)
    $sub:=(NN & upperInitial) %
    $obj:=NN
```

3.3. Employing Ontology Elements in the Rules

By employing ontology elements, we are adding semantics to the rules. For instance, if there is a news article about *kb:Google* introducing a new product, e.g., *kb:Chrome*, and *kb:Google* already has an entry in the knowledge base, it is possible to annotate the lexical representation of *kb:Chrome* as a product and add a product-relation between *kb:Chrome* and *kb:Google*. When ontologies are employed in the rules, potentially one rule can be used to describe multiple lexical representations. In this example three features of an ontology occur. First, company is a *class*. Second, *kb:Google* and the product (*kb:Chrome*) are *instances* of classes, and third, the relationship between *kb:Google* and the product represents an *object property*. We now continue by discussing how these three features of the ontology can be employed in information extraction rules.

3.3.1. Concepts

Classes are groups of individuals that share the same properties [7]. For example, *kb:Google* and *kb:Microsoft* both belong to the same class, i.e., *kb:Company*. Other examples of classes are *kb:Product*, *kb:Person*, and *kb:Country*. In information extraction it is useful to look for specific instances in the text. Instances are more specific than classes and are generally used on the RHS of the rule.

In the language we make a distinction between classes and instances. If the rule is to recognize a specific instance of a certain concept it is denoted by the instance itself. The following rule shows an example:

```
($sub, kb:hasProduct, $obj) :-             (Rule 13)
    $sub:=kb:Google kb:Buys
    $obj:=mixedCaps
```

This rule contains two instances, namely *kb:Google* and *kb:Buys*, and are matched to a sentence like "*Google Inc. acquires YouTube*," because "*Google Inc.*" is a lexical representation of the instance *kb:Google* and in a similar manner is "*acquires*" a lexical representation of *kb:Buys*. By employing classes instead of specific instances, the rules become more generic. An example of a rule using classes is:

```
($sub, rdf:typeOf, kb:Company) :-          (Rule 14)
    [kb:Company] (',' | 'and')
    $sub:=(NNP{1,})
```

In the above rule, a list of companies is recognized. The square parentheses denote all the instances of the enclosed type. Each instance has associated lexical representations as we have previously seen. In this example, the proper nouns (NNP) will be annotated as an instance of a company. Assuming that *Google* is already known as a concept, in order to recognize other companies, we can match the rule on the sentence "*A Big-Picture Look at Google, Microsoft Corporation, Apple and Yahoo!*". The first time this is done, "*Microsoft Corporation*" will be annotated as a company, while in order to recognize "*Apple*" and "*Yahoo!*" as well, the rule needs to be run a second and a third time.

3.3.2. Relations between Concepts

As stated earlier, the LHS of the HIEL patterns is used for recognizing concepts, and it is a triple that describes the relationship between a subject and an object. By using labels, we can refer in the LHS to a concept found on the RHS. For instance, a rule such as

```
($sub, kb:hasSubsidiary, $obj) :-          (Rule 15)
    $sub:=[kb:Company] kb:Buys
    $obj:=[kb:Company]
```

can be employed in order to extract the *kb:hasSubsidiary* relation between two companies. The concept *kb:Buys* has various synonyms such as: "*buy*", "*acquire*", and "*take over*". If we apply this rule to the sentence "*Google buys YouTube for $1.65 billion*", it would extract the *kb:Buys* relation between *kb:Google* and *kb:YouTube*. This information can then be used in order to update the ontology, and for instance remove the existing competitor relationship between the companies.

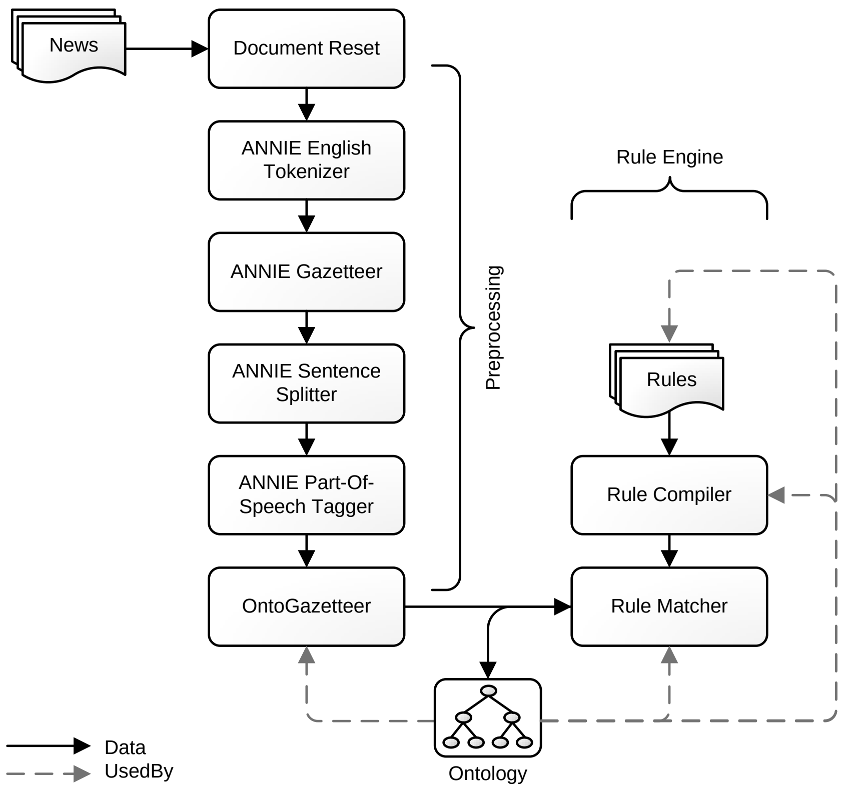


Figure 1: Overview of the Hermes processing pipeline

4. Hermes Information Extraction Engine

Based on the language defined in this paper, we have implemented the Hermes Information Extraction Engine (HIEE). In this section, we first discuss the Hermes News Portal (HNP) in Subsect. 4.1, followed by Subsect. 4.2 that briefly touches upon the general framework that lies underneath the HNP and the HIEE plug-in. Subsequently, Subsect. 4.3 presents the preprocessing of the news items. Subsection 4.4 discusses the rule engine and Subsect. 4.5 illustrates the plug-in for the Hermes News Portal.

4.1. Hermes News Portal

The implementation of the Hermes framework is the Hermes News Portal (HNP), which allows users to formulate queries and execute them on the domain ontology in order to retrieve relevant news items. The HNP application is a stand-alone, Java-based tool which makes use of various Semantic Web technologies.

The internal knowledge base is in fact a domain ontology constructed by domain experts, represented in OWL [7]. While populated ontologies are typically queried by the Semantic Web's standard query language SPARQL [28], querying within HNP is done by means of extended SPARQL queries. Because within the Hermes News Portal time-specific features are exploited, time functionalities were added to SPARQL, which resulted in tSPARQL [11, 12]. Within HNP, the classification of the news articles is done using GATE [41] and the WordNet [46] semantic lexicon. The classification occurs prior to the rules execution that extract information from news.

4.2. General Framework

We developed a general framework that supports our Hermes Information Extraction Engine (HIEE) plug-in. Figure 1 presents the architecture of the processing pipeline that lies underneath our implementation. The framework consists of two main parts, i.e., the preprocessing stage and the rule engine. These parts and their individual components are executed in a specific order, and are discussed in more detail in the following subsections.

In short, preprocessing – which is described in more detail in Subsect. 4.3 – is done using the existing HNP natural language processing pipeline, which classifies news items using the GATE architecture [41]. Most of the components stem from the A Nearly-New Information Extraction (ANNIE) system, which is a selection of standard GATE components. In addition, an ontology-enabled gazetteer is employed.

In contrast to most preprocessing components, the rule engine, which is described in more detail in Subsect. 4.4, makes use of ontologies. The engine consists of two core components, i.e., lexico-semantic pattern (rule) compilation and matching. The compiler and matcher make use of semantic components, i.e., concepts and individuals stored in the main ontology, and syntactic elements, such as Part-Of-Speech (POS) tags that are generated in the preprocessing stage.

4.3. Preprocessing

Before the rules can be employed to match patterns in text, a few processing tasks need to be performed, like tokenization, sentence splitting, and Part-Of-Speech (POS) tagging, which are dealt with by the GATE architecture [41]. GATE provides a pipeline consisting of different components, each of which handles a different aspect of the language processing. The components that are part of the pipeline, and come with GATE by default, are in order of usage: *Document Reset*, *ANNIE English Tokenizer*, *ANNIE Gazetteer*, *ANNIE Sentence Splitter*, *ANNIE Part-Of-Speech Tagger*, and *OntoGazetteer*.

The *Document Reset* component is used for resetting the document, in this case a news item, to its original state. The document is cleared from all its current annotations, enabling the pipeline to re-annotate the text. This is especially useful when running the document through a pipeline several times, as it is undesirable to use a document with previous annotations in an information extraction process. Subsequently, the *ANNIE English Tokenizer* splits the corpus into tokens, such as numbers, punctuation, and words of different types. A distinction is made between words in uppercase and lowercase, and between certain types of punctuation.

After these basic operations, the *ANNIE Gazetteer* looks up words from gazetteer lists (i.e., lists with names of, for example, cities, countries, companies, days of the week, world leaders, etc.) in order to be able to classify them. In our implementation, the latter task is limited to some basic and static lists, such as days of the week, months of the year, etc. After gazetteering, the *ANNIE Sentence Splitter* is employed, which identifies sentences, required for the *ANNIE Part-Of-Speech Tagger*. This tagger is a modified version of the Brill tagger [47], which produces a POS tag as an annotation to each word or symbol. The POS tags, e.g., the ones described in Table 1, can be used in the rules to describe certain patterns.

Finally, the *OntoGazetteer* component is executed, which has similarities with the ANNIE Gazetteer. The biggest difference lies in the fact that the *OntoGazetteer* is an ontology-enabled component, i.e., it utilizes terms stored in an ontology instead of plain gazetteer lists for classification. The component still utilizes lists in order to perform its tasks, but in addition provides a mapping definition between the lists and the ontology classes. The *OntoGazetteer* searches the corpus for occurrences of OWL annotation properties – these are the concept lexical representations – of the classes and instances of the ontology. The found matches are annotated with the name of the OWL instance (or class) against which the piece of text is matched. In order to assure a good performance, one should make sure that the ontology has an extensive list of lexical representations associated to each depicted concept or relation. After annotation using *OntoGazetteer*, tokens have been linked to the ontology, and hence can be used in lexico-semantic patterns. A sentence like "*The conference will be attended by CEOs like Steve Ballmer and Steve Jobs*", gives us the opportunity to recognize "*Steve Ballmer*" and "*Steve Jobs*" as CEOs.

4.4. Rule Engine

After preprocessing a news corpus, the Hermes Information Extraction Rule Engine compiles the rules in the *Rule Compiler* and matches these rules to the text using the *Rule Matcher*. Because we use news items, employing the extracted information it is possible to adapt the underlying ontology based on certain events. For instance, "*Eric Schmidt leaves Google*", informs us that "*Eric Schmidt*" is no longer the CEO of "*Google*" and hence results in an ontology update. Note that in order for the rule engine to be able to run as a stand-alone application, we do not create dependencies with respect to GATE's default JAPE language. Hence, because no conversion is made to JAPE rules, we enable one to employ the rule engine within other information extraction frameworks as a stand-alone component. Also, we have to take into consideration that JAPE might not be suitable to support possible future extensions to HIEL.

The *Rule Compiler* is created using the Java Compiler Compiler [48], developed by Sun Microsystems. The Java Compiler Compiler generates a compiler for the grammar defined in Sect. 3 and Appendix A. During the compilation, Java objects are being created that represent the various parts of a rule. The right-hand side (RHS) of a rule can be represented as a tree, as shown in Fig. 2. Components in this tree are of two main types: internal nodes and leaf nodes. Internal nodes consist of one or more internal nodes or leaf nodes and include sequences, logical operators, and repetitions. Leaf nodes are nodes that do not have any child nodes and include literals, concepts, orthographical categories, Part-of-Speech categories, and wildcards. After the rules are compiled, the matcher tries to match the rules onto the text.

In order to match the compiled rules to the text, each tree node performs its own task. In our recursive algorithm which starts at the tree's root node, child node procedure calls are performed. These children try to match as many tokens as possible. Non-leaf nodes, i.e., nodes that contain child nodes, keep performing calls to their children until a leaf node has been reached. Subsequently, leaf nodes check whether the token at the current position is a match. Each child node reports to its parent the number of tokens it was able to match until the root of the tree is reached. If the root returns a value which is equal or greater than the value of the position it started with, the rule has been matched to the text. This process is repeated until the last token of the text has been reached. After matching a rule, tokens on the right-hand side are bound to labels to be used in the left-hand side of the rule. This allows for determining which tokens belong to the subject, predicate, and object of the computed triple.

In Fig. 3 an example tree of Rule 14 is shown. If we consider the following sentence: *"ASUS and Microsoft Corporation become official partners for Windows Phone 7"*, where *ASUS* is a known instance of *Company* in the ontology and *Microsoft Corporation* is not, the process is as follows. `Sequence` sends a `getNextMatch()` call to `kb:Company`, which returns 1, indicating that one token has been matched. Subsequently, after receiving the response, the `Sequence` sends a `getNextMatch()` call to the `OR` which passes it on to the literal "`and`" which returns 1. Finally, the `Repetition` tries to match the `NNP` as many times as possible, which results in 2, as "*Microsoft Corporation*" has two words. Note that the tokens matched by the `Repetition`, "*Microsoft Corporation*", are assigned to the left-hand side (LHS) entity `sub`.

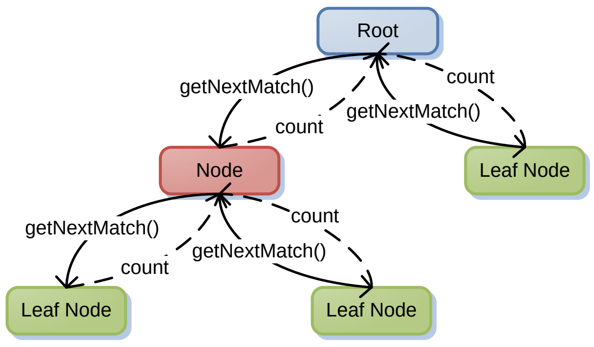


Figure 2: Rule tree template

Regarding speed, the Hermes Information Extraction Rule Engine is able to run on a real-time basis, as both rule compilation and executing rules for one news message have subsecond performance. We did not encounter any speed issues that can be attributed to OWL operations on the underlying ontology, as we only deal simple inferences based on the `typeOf` relations.

4.5. Hermes Plug-in

In order to be able to evaluate the usability and expressivity of the proposed information extraction language, the HIEE plug-in for the Hermes framework was created. This plug-in allows one to create, edit, use, and evaluate extraction rules, and is composed of three different parts, i.e., the rule editor, the annotation validator, and the manual annotator.

The rule editor, as displayed in Fig. 4, allows users to create their own personal information extraction rules. These rules can be divided into so-called rule groups, enabling clustering of different rules of the same type (i.e., they discover the same type of event). After creating a rule, the user is given the option to validate and save the

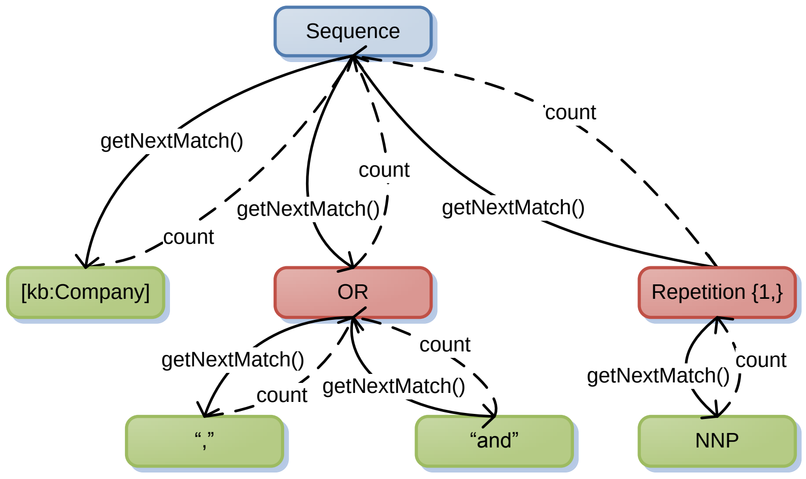


Figure 3: Rule tree example of Rule 14

rule. Whenever syntactical mistakes – such as typographical errors, but in worse cases violations of the grammar as defined in this paper – are made by the user, the built-in compiler will detect them and display informative messages to the user. A rule cannot be saved if it is not valid, ensuring the validity of the rules by construction.

The annotation validator, as depicted in Fig. 5, displays the resulting annotations after applying the user-defined extraction rules to the existing news items. Each result is described by the extracted triple defined in the left-hand side of a rule, a number indicating the number of occurrences of the triple in the evaluated news items, and a checkbox to allow for user validation, resulting in knowledge base updating. After marking the correctly extracted facts as valid, a new annotation run can be executed. The previously extracted facts will be stored in the knowledge base and subsequently, new information can be extracted using these new facts.

The manual annotator is used for evaluating the current rule set. For each news item, the user is able to manually annotate tokens from the news item. When selecting a token, its current classified annotations, such as the Part-Of-Speech (POS) tag, the orthographical category, and the ontology concepts are displayed. If new ontology annotations are preferred, a concept can be selected from a list with existing concepts from the ontology. Events can be described by selecting the subject, predicate, and object in the text and annotate them with the corresponding ontology concepts.

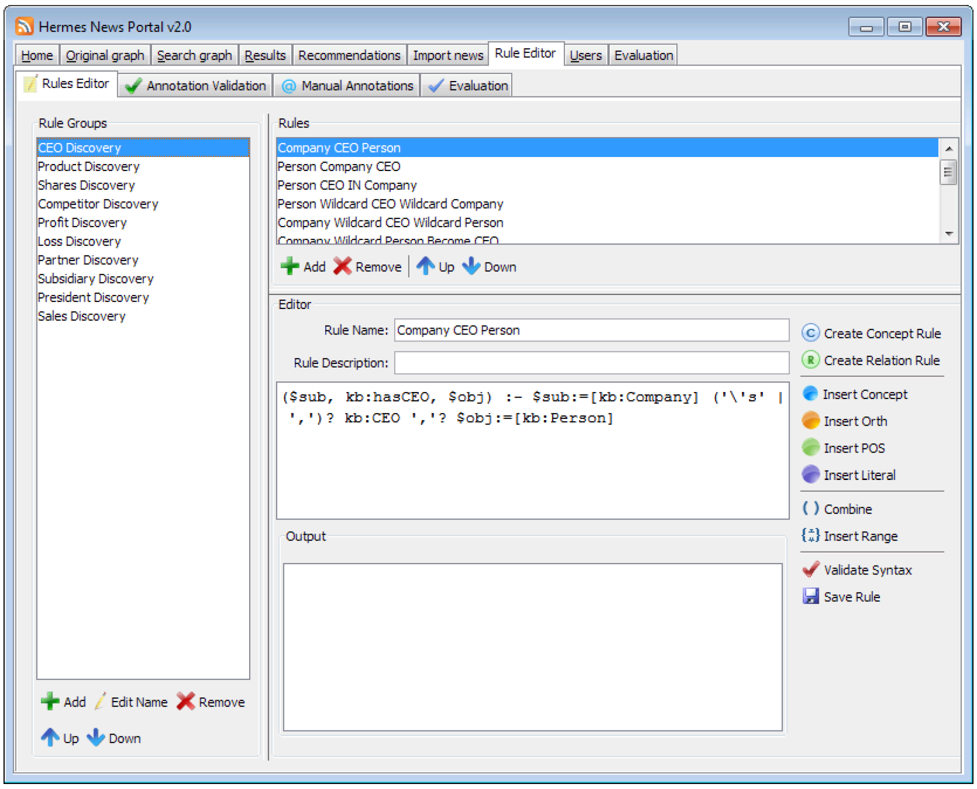
5. Evaluation

In order to evaluate the effectiveness of our approach, we have implemented a test method and built a test environment. First we discuss the evaluation setup in Subsect. 5.1, followed by the results, in Subsect. 5.2.

5.1. Evaluation Setup

For testing the performance of the extraction language, we assembled news messages from financial news feeds, totalling 500 items with an average length of 4,200 words and multiple paragraphs. News messages are written in English using an extensive vocabulary. These news items are divided into two sets, i.e., a training set consisting of 300 news items, and a test set consisting of 200 news items. The gathered news items originate from Reuters Business and Technology News and from The New York Times Business News. Next, an ontology is provided to domain experts (i.e., colleagues with an expertise in finance) that are asked to annotate the news messages and to develop event extraction rules. A similar approach is followed for a second data set containing 100 political news messages, with an average length of 700 words, mainly gathered from Reuters Politics News and Yahoo! Politics News.

The ontologies employed in our experiments contain major domain concepts and their most common representations, and are not overly detailed. It is not within the scope of this paper to develop


Hermes News Portal v2.0
Home | Original graph | Search graph | Results | Recommendations | Import news | Rule Editor | Users | Evaluation
Rules Editor | Annotation Validation | Manual Annotations | Evaluation
Add | Remove | Up | Down
Add | Edit Name | Remove | Up | Down
Create Concept Rule
Create Relation Rule
Insert Concept
Insert Orth
Insert POS
Insert Literal
Combine
Insert Range
Validate Syntax
Save Rule
Output

Rule Groups

- CEO Discovery
- Product Discovery
- Shares Discovery
- Competitor Discovery
- Profit Discovery
- Loss Discovery
- Partner Discovery
- Subsidiary Discovery
- President Discovery
- Sales Discovery

Rules

- Company CEO Person
- Person Company CEO
- Person CEO IN Company
- Person Wildcard CEO Wildcard Company
- Company Wildcard CEO Wildcard Person
- Company Wildcard Person Become CEO

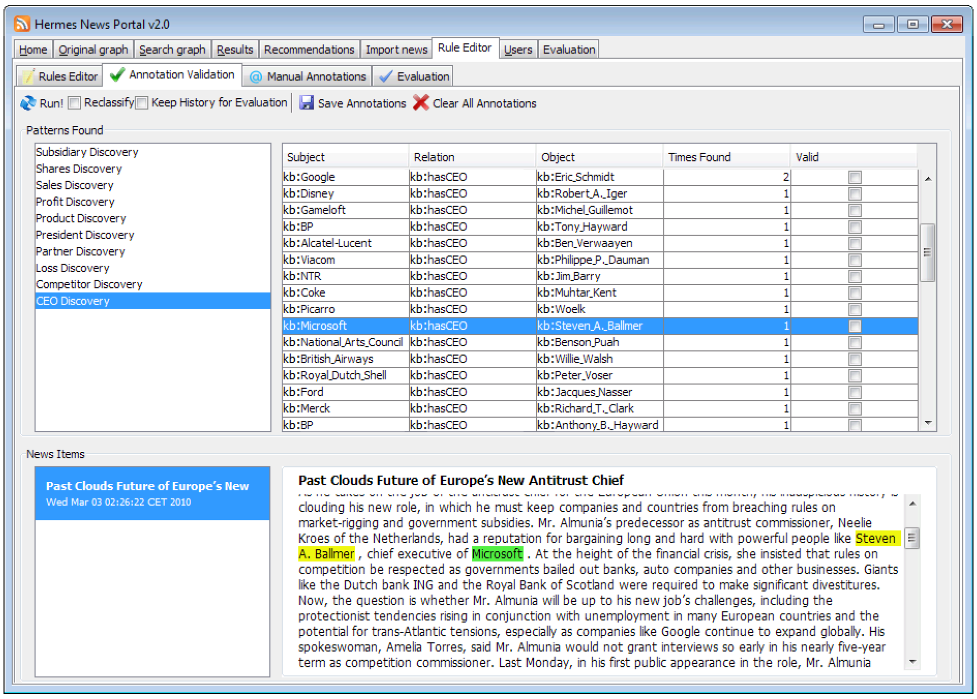
Editor

Rule Name: Company CEO Person

Rule Description:

```
($sub, kb:hasCEO, $obj) :- $sub:=[kb:Company] ('\'s' |
 ',')? kb:CEO ','? $obj:=[kb:Person]
```

Figure 4: Rule editor


Hermes News Portal v2.0
Home | Original graph | Search graph | Results | Recommendations | Import news | Rule Editor | Users | Evaluation
Rules Editor | Annotation Validation | Manual Annotations | Evaluation
Run! | Reclassify | Keep History for Evaluation | Save Annotations | Clear All Annotations

Patterns Found

- Subsidiary Discovery
- Shares Discovery
- Sales Discovery
- Profit Discovery
- Product Discovery
- President Discovery
- Partner Discovery
- Loss Discovery
- Competitor Discovery
- CEO Discovery

Subject	Relation	Object	Times Found	Valid
kb:Google	kb:hasCEO	kb:Eric_Schmidt	2	☐
kb:Disney	kb:hasCEO	kb:Robert_A._Iger	1	☐
kb:Gameloft	kb:hasCEO	kb:Michel_Guillemot	1	☐
kb:BP	kb:hasCEO	kb:Tony_Hayward	1	☐
kb:Alcatel-Lucent	kb:hasCEO	kb:Ben_Verwaayen	1	☐
kb:Viacom	kb:hasCEO	kb:Philippe_P._Dauman	1	☐
kb:NTR	kb:hasCEO	kb:Jim_Barry	1	☐
kb:Coke	kb:hasCEO	kb:Muhtar_Kent	1	☐
kb:Picarro	kb:hasCEO	kb:Woelk	1	☐
kb:Microsoft	kb:hasCEO	kb:Steven_A._Ballmer	1	☐
kb:National_Arts_Council	kb:hasCEO	kb:Benson_Puah	1	☐
kb:British_Airways	kb:hasCEO	kb:Willie_Walsh	1	☐
kb:Royal_Dutch_Shell	kb:hasCEO	kb:Peter_Voser	1	☐
kb:Ford	kb:hasCEO	kb:Jacques_Nasser	1	☐
kb:Merck	kb:hasCEO	kb:Richard_T._Clark	1	☐
kb:BP	kb:hasCEO	kb:Anthony_B._Hayward	1	☐

News Items

Past Clouds Future of Europe's New
Wed Mar 03 02:26:22 CET 2010

Past Clouds Future of Europe's New Antitrust Chief

clouding his new role, in which he must keep companies and countries from breaching rules on market-rigging and government subsidies. Mr. Almunia's predecessor as antitrust commissioner, Neelie Kroes of the Netherlands, had a reputation for bargaining long and hard with powerful people like Steven A. Ballmer , chief executive of Microsoft . At the height of the financial crisis, she insisted that rules on competition be respected as governments bailed out banks, auto companies and other businesses. Giants like the Dutch bank ING and the Royal Bank of Scotland were required to make significant divestitures. Now, the question is whether Mr. Almunia will be up to his new job's challenges, including the protectionist tendencies rising in conjunction with unemployment in many European countries and the potential for trans-Atlantic tensions, especially as companies like Google continue to expand globally. His spokeswoman, Amelia Torres, said Mr. Almunia would not grant interviews so early in his nearly five-year term as competition commissioner. Last Monday, in his first public appearance in the role, Mr. Almunia

Figure 5: Annotation validation

Name	Subject	Relation	Object
CEO	Company	hasCEO	Person
Product	Company	hasProduct	Product
Shares	Company	hasShareValue	Literal
Competitor	Company	hasCompetitor	Company
Profit	Company	hasProfit	Literal
Loss	Company	hasLoss	Literal
Partner	Company	hasPartner	Company
Subsidiary	Company	hasSubsidiary	Company
President	Company	hasPresident	Person
Revenue	Company	hasRevenue	Literal

Table 2: Relations and events for the financial domain, used for evaluation purposes

Name	Subject	Relation	Object
Election	Person	isElectedAs	Function
Visit	Person	visits	Country
Sanction	Country	sanctions	Country
Join	Country	joins	Union
Resignation	Person	resignsFrom	Function
Investment	Country	investsIn	Country
Riots	Country	hasRiots	N/A
Collaboration	Country	collaboratesWith	Country
Provocation	Country	provokes	Country
Help	Country	helps	Country

Table 3: Relations and events for the political domain, used for evaluation purposes

large, complete, and exhaustive ontologies for the specific domains as we merely explore the functionalities of our language by means of concepts within a particular financial or political context. The developed ontologies allow domain experts to annotate texts with common concepts from finance and politics, and to recognize frequently occurring financial and political events.

Our financial ontology contains a small subset of commonly used, well-known, financial entities. Examples of ontology concepts are: companies, products, persons, currencies, CEOs, etc. These concepts have associated lexical representations, e.g., the CEO concept has associated "*CEO*", "*Chief Executive Officer*", "*Chief Executive*", etc. The ontology consists of 65 classes, 18 object properties, 11 data properties, and 1,167 individuals, which can be used for annotation and event detection.

The ontology that is used for event discovery in political news items is also a high-level ontology, yet considerably smaller than the financial ontology. Our political ontology contains 14 classes, 12 object properties, 5 data properties, and 391 individuals. Most individuals are associated with countries. Also, we included many lexical representations of politics-related nouns and verbs, e.g., those linked to elections, provocations, meetings, etc.

For each data set, three domain experts manually annotate the events and relations that we take into account in our evaluation, based on an inter-annotator agreement of at least 66% (i.e., at least two out of three annotators should agree). During the evaluation we focus on the extraction of ten events and relations from the financial domain and ten events and relations from the political domain. Each of these events are described in Tables 2 and 3, by a name, subject, relation, and an optional object. Based on the events and relations that exist in the news items in the training sets, we let three domain experts construct a set of information extraction rules, where we take the conjunction of the three constructed rule sets. The constructed rules are subsequently matched to the news items in the test sets, in order to measure the performance.

In our experiments, for each rule group we compare the performance of lexico-syntactic patterns (our baseline) to the performance of lexico-semantic patterns written in HIEL and in JAPE in terms of construction time (i.e., efficiency) and in terms of precision and recall (i.e., expressivity). The latter two measures are often employed in the information extraction field, i.e., *precision* P and *recall* R. These measurements are defined as follows:

$$P = \frac{|\{Relevant\} \cap \{Found\}|}{|Found|} , \quad (1)$$

$$R = \frac{|\{Relevant\} \cap \{Found\}|}{|Relevant|} , \quad (2)$$

where *Relevant* is the set of relevant annotations (events) and *Found* is the set of found annotations. There is a trade-off between precision and recall, and hence we compute the F_1 measure. The F_1 measure is applied to compute an even combination, i.e., the harmonic mean of precision and recall:

$$F_1 = \frac{2 \times Precision \times Recall}{Precision + Recall} . \quad (3)$$

We measure the rule creation times by averaging the individual rule set creation times of our domain experts. We evaluate the average time it takes for the F_1 measures to become equal to or higher than 0.5. Such a value would be large enough to rule out randomness, as the F_1 measure for a random classifier (based on prior occurrence probability) is a lot less than 0.5 due to the fact that events are seldomly occurring in a news item (when comparing the likelihood of a specific event occurrence with the absence of a specific event with respect to a possible event word sequence in a news item). In theory, creating patterns with an F_1 performance of 0.5 should be manageable within a reasonable amount of time. Additionally, with F_1 scores of 0.5, one avoids the risk of overfitting patterns to a specific data set.

We hypothesize that the creation of well-performing lexico-syntactic rule groups requires more time than the creation of the equivalent lexico-semantic ones. In a second experiment, rule quality, indicated by the precision, recall, and F_1 measures is evaluated for lexico-syntactic, HIEL, and JAPE rule groups given a fixed time in which our domain experts are allowed to create and improve the individual rules. We allow the domain experts to improve their (HIEL and JAPE) lexico-semantic rule groups up until the time it took for creating the equally performing lexico-syntactic rule groups.

5.2. Evaluation Results

The construction times presented in Tables 4 and 5 confirm our hypothesis that the creation of lexico-syntactic rules requires more time than the creation of equally performing lexico-semantic rules, both in HIEL and in JAPE. The tables display rule group creation times in seconds for the lexico-syntactic and lexico-semantic variants, which are obtained on our test sets while aiming for an F_1 score of at least 0.5. For our financial data set, on average, equally well-performing lexico-semantic rule groups are created up to 5 to 70 times faster than their lexico-syntactic counterparts. For JAPE patterns, creation times are considerably lower than for lexico-syntactic rules, yet they are higher than those for HIEL lexico-semantic patterns. Addition-

	HIEL		JAPE
Name	Lex-Syn	Lex-Sem	Lex-Sem
CEO	8,424	281	738
Product	9,428	132	312
Shares	2,403	648	703
Competitor	9,116	133	850
Profit	1,923	416	1,027
Loss	5,991	313	589
Partner	4,924	185	474
Subsidiary	6,620	776	1,851
President	4,239	179	722
Revenue	5,317	498	798
Overall	5,839	356	806

Table 4: Creation times (in seconds) of lexico-syntactic and lexico-semantic rule groups in HIEL, and lexico-semantic rule groups in JAPE, using the financial test set ($F_1 \geq 0.5$)

	HIEL		JAPE
Name	Lex-Syn	Lex-Sem	Lex-Sem
Election	1,517	232	689
Visit	4,238	543	913
Sanction	4,013	419	1,247
Join	3,986	297	405
Resignation	1,259	366	540
Investment	5,162	781	2,304
Riots	1,734	306	451
Collaboration	1,103	137	719
Provocation	1,428	530	828
Help	1,987	211	362
Overall	2,643	382	846

Table 5: Creation times (in seconds) of lexico-syntactic and lexico-semantic rule groups in HIEL, and lexico-semantic rule groups in JAPE, using the political test set ($F_1 \geq 0.5$)

ally, for our political data set we observe similar results, although the measured differences are regularly smaller.

The major cause of the construction time reduction that is measured when switching from lexico-syntactic to lexico-semantic patterns lies within the fact that concepts used in HIEL and JAPE lexico-semantic rules, e.g., persons and companies, are conveniently described in an ontology (containing classes, instances, and their associated lexical representations), thus enabling easy reuse. For lexico-syntactic rules however, it is difficult and cumbersome to create rules that distinguish names of persons from companies, products, months, days, etc. Additionally, the verbosity of lexico-syntactic rules and the use of literals to exclude common words (e.g., months) contribute to a considerable amount of extra creation time.

Let us consider a rule that extracts provocation events, where one country provokes another. When solely utilizing lexico-syntactic elements within the pattern, one would need to intelligently combine lexicographic and orthographic categories. For instance, a country could be defined as a series of nouns and adjectives that contain capitals, i.e., `((JJ | NNS | NNP | NNPS | NN) & (upperInitial | allCaps | mixedCaps))+`, matching phrases like "*Spain*", "*United States*", etc. Additionally, this could be extended so that it would also match strings like "*U.S.*" by adding an extra condition, resulting in `(((JJ | NNS | NNP | NNPS | NN) & (upperInitial | allCaps | mixedCaps)) ('.' NNP '.'?)?)+`. However, finding the right combination of nouns and conditions in order to match countries and not persons, companies, etc., is a tedious task. An example of a lexico-syntactic provocation discovery rule is:

```
($sub, kb:provokes, $obj) :-                 (Rule 16)
   $sub:=(
          (
           (JJ | NNS | NNP | NNPS | NN) &
           (upperInitial | allCaps | mixedCaps)
          )
          ('.' NNP '.'?)?
         )+
   (!'.' & !'(' & !')' & !'-'){0,3}
   ('angers' | 'angered' | 'accuses' |
    'accused' | 'insult' | 'insulted' |
    'provokes' | 'provoked' | 'threatens' |
    'threatened')
   (!'.' & !'(' & !')' & !'-'){0,3}
   $obj:=(
          (
           (JJ | NNS | NNP | NNPS | NN) &
           (upperInitial | allCaps | mixedCaps)
          )
          ('.' NNP '.'?)?
         )+
```

Here, the subject and object are defined as series of capitalized nouns, possibly representing countries. Additionally, verbs related to provocation are required. These are enumerated as literals. Finally, the pattern allows up to three non-punctuation tokens in between the countries and the verb.

When replacing lexical categories and literals with concepts stemming from our political ontology, we obtain the following lexico-semantic rule in HIEL:

```
($sub, kb:provokes, $obj) :-           (Rule 17)
    $sub:=([kb:Country] | [kb:Continent] |
           [kb:Union])
    (!'.' & !'(' & !')' & !'-'){0,3}
    (kb:toAnger | kb:toAccuse | kb:toInsult |
     kb:toProvoke | kb:toThreaten)
    (!'.' & !'(' & !')' & !'-'){0,3}
    $obj:=([kb:Country] | [kb:Continent] |
           [kb:Union])
```

The rule is much cleaner and takes considerably less effort to write. As concepts like countries, continents, and unions are conveniently described in the ontology, the user merely needs to refer to them and avoids the hassle of trying to find optimal combinations of lexicographic and orthographic categories, keywords, etc. Moreover, lexico-semantic rules exploit the `typeOf` hierarchy, i.e., because of the inference that can be applied to ontological concepts, the user can suffice with using concepts like `Country`, instead of their subclasses `US`, `UK`, etc., that have associated lexical representations.

Even though JAPE is more expressive than HIEL as it supports templates (macros) as well as the usage of any Java code – which is useful for removing temporary annotations, percolating and manipulating features from previous annotations, etc. – HIEL rules offer more accessibility to the user. Let us consider the following rule, which is an exact JAPE copy of our previously introduced HIEL rule:

```
Rule: Geo_provokes_Geo                          (Rule 18)
(
    (
        {Lookup.classURI == "Country"} |
        {Lookup.classURI == "Continent"} |
        {Lookup.classURI == "Union"}
    ):sub
    ({!Token.string ==~ "[.()-]"})[0,3]
    (
        {Lookup.URI == "toAnger"} |
        {Lookup.URI == "toAccuse"} |
        {Lookup.URI == "toInsult"} |
        {Lookup.URI == "toProvoke"} |
        {Lookup.URI == "toThreaten"}
    )
    ({!Token.string ==~ "[.()-]"})[0,3]
    (
        {Lookup.classURI == "Country"} |
        {Lookup.classURI == "Continent"} |
        {Lookup.classURI == "Union"}
    ):obj
)
:match --> :match.provokes =
    {sub = :sub.Lookup.propertyValue,
     obj = :obj.Lookup.propertyValue}
```

Even without employing the extra features that JAPE rules offer, we already obtain a rule that is more verbose. Therefore, this rule takes considerably longer to write than lexico-semantic HIEL rules. On the other hand, due to the availability of ontology concepts also the creation of lexico-semantic JAPE rules requires less effort than constructing plain lexico-syntactic rules.

In Table 6, the experimental results of lexico-semantic rules on the test set are displayed for the financial data set. After allowing the domain experts to improve the lexico-semantic rules written in HIEL up until the time it took for creating the equally performing lexico-syntactic rules (e.g., $[2,403 - 648 =]$ $1,755$ extra seconds for shares discovery), the overall precision and recall are 84% and 74%, respectively, resulting in an F_1 score of approximately 79%. With measured precision, recall, and F_1 scores of 85%, 58%, and 69%, respectively, the lexico-semantic rules that are written in JAPE perform notably better than the lexico-syntactic rules, which have a precision, recall, and F_1 measure of 55%, 49%, and 52%, respectively, yet their performance is consistently worse than the performance of lexico-semantic HIEL rules.

For both lexico-semantic pattern languages, the highest recalls are obtained for CEO, Shares, and Partner relations. This is mainly due to the homogeneous sentence structures related to these relations. Judging from the low recalls, the subsidiary and president relations were harder to discover in the text. This could be caused by overfitted rules, which means that it was difficult to create generic rules on the training set that would match many different instances of these relations. The same can be said for the precision and recall (and hence the F_1 value) of the discovery of a company's loss. Another notable observation is the high number of product relations that are discovered in our data set, which can be explained by the fact that many news items discuss companies and their products.

For our political data set, we observe similar overall performances, as depicted in Table 7. Generally, lexico-semantic patterns written in HIEL perform better than those written in JAPE. While we observe a precision and recall of 76% and 72%, respectively, for lexico-semantic HIEL rule sets, JAPE rules measure respective scores of 70% and 63%. With F_1 scores of 74% and 66%, this is still considerably better than the 51% accomplished by the lexico-syntactic rules. High precisions and recalls are observed in rules covering elections, resignations, and riots, as these events can usually be found in non-complex sentences where key terms are closely located near one another. Political visits and provocations suffer from low recall values, caused by the wide structural variety and complexity of sentences denoting these events.

On a side note, within our framework it is relatively straightforward to obtain high recall scores. For instance, it would be likely for a rule such as

```
($sub, kb:hasProduct, $obj) :-          (Rule 19)
    $sub:=[kb:Company] %
    $obj:=[kb:Product]
```

to discover each and every existing product relation. However, there is a tradeoff between high recall and high precision. In order to obtain high scores for both measures (expressed in a high F_1 score), rules need to be far more sophisticated. In texts that contain product relations, often several different companies are mentioned, which makes it difficult to match only the right product with the right company.

Based on the evaluation results, we validated the requirements set for our approach in Sect. 1. Our proposed language, HIEL, is more easy to use for

	Lex-Syn HIEL					Lex-Sem HIEL					Lex-Sem JAPE				
Name	P	R	F_1	+	–	P	R	F_1	+	–	P	R	F_1	+	–
CEO	0.5217	0.6000	0.5581	69	24	0.8966	0.8667	0.8814	58	8	0.9412	0.5333	0.6808	34	28
Product	0.6667	0.4118	0.5091	84	80	0.8607	0.7721	0.8140	122	31	0.8558	0.6544	0.7417	104	47
Shares	0.4286	0.6667	0.5218	70	15	0.9000	0.8000	0.8471	40	9	0.8889	0.7111	0.7901	36	13
Competitor	0.5333	0.4800	0.5052	45	26	0.7600	0.7600	0.7600	50	12	0.7568	0.5600	0.6437	37	22
Profit	0.7500	0.4545	0.5660	20	18	0.8800	0.6667	0.7586	25	11	0.9375	0.4545	0.6122	16	18
Loss	0.6471	0.4074	0.5000	17	16	0.8125	0.4815	0.6047	16	14	0.9231	0.4444	0.6000	13	15
Partner	0.3864	0.7391	0.5075	44	6	0.8000	0.8696	0.8333	25	3	0.7619	0.6957	0.7273	21	7
Subsidiary	0.7500	0.3913	0.5143	24	28	0.9063	0.6304	0.7436	32	17	0.9091	0.4348	0.5883	22	26
President	0.4333	0.5909	0.5000	30	9	0.6667	0.6364	0.6512	21	8	0.8462	0.5000	0.6286	13	11
Revenue	0.6429	0.4091	0.5000	14	13	0.7143	0.6818	0.6977	21	7	0.7059	0.5455	0.6154	17	10
Overall	0.5492	0.4935	0.5199	417	235	0.8390	0.7414	0.7872	410	120	0.8530	0.5754	0.6872	313	197

Table 6: Results of lexico-syntactic and lexico-semantic rule groups on the financial test set (within fixed time), displaying precision (P), recall (R), and F_1 scores, as well as the number of items found (+) and the number of items missed (–)

	Lex-Syn HIEL					Lex-Sem HIEL					Lex-Sem JAPE				
Name	P	R	F_1	+	–	P	R	F_1	+	–	P	R	F_1	+	–
Election	0.4615	0.5455	0.5000	13	5	1.0000	0.8182	0.9000	9	2	0.8182	0.8182	0.8182	11	2
Visit	0.6774	0.4038	0.5060	31	31	0.7027	0.5000	0.5843	37	26	0.6944	0.4808	0.5682	36	27
Sanction	0.5263	0.4918	0.5085	57	31	0.7857	0.7213	0.7521	56	17	0.6774	0.6885	0.6829	62	19
Join	0.7222	0.4063	0.5200	18	19	0.8125	0.8125	0.8125	32	6	0.7917	0.5938	0.6786	24	13
Resignation	0.9091	0.3846	0.5405	11	16	0.8000	0.9231	0.8572	30	2	0.7333	0.8462	0.7857	30	4
Investment	0.5208	0.4808	0.5000	48	27	0.7778	0.6731	0.7217	45	17	0.7500	0.5192	0.6136	36	25
Riots	0.5200	0.5200	0.5200	50	24	0.7069	0.8200	0.7593	58	9	0.6863	0.7000	0.6931	51	15
Collaboration	0.4250	0.6538	0.5151	40	9	0.7143	0.7692	0.7407	28	6	0.5000	0.6923	0.5806	36	8
Provocation	0.7727	0.4250	0.5484	22	23	0.7857	0.5714	0.6616	28	18	0.7778	0.5250	0.6269	27	19
Help	0.5510	0.4737	0.5094	49	30	0.7541	0.8070	0.7797	61	11	0.7273	0.7018	0.7143	55	17
Overall	0.5664	0.4694	0.5134	339	215	0.7630	0.7164	0.7390	384	114	0.7011	0.6308	0.6641	368	149

Table 7: Results of lexico-syntactic and lexico-semantic rule groups on the political test set (within fixed time), displaying precision (P), recall (R), and F_1 scores, as well as the number of items found (+) and the number of items missed (–)

expressing lexico-semantic patterns than the current state-of-the-art JAPE language. Also, we have shown the superiority of lexico-semantic approaches over lexico-syntactic ones with respect to both precision and recall.

6. Conclusion

As structuring data on the Web is a tedious and time consuming process, in this paper, we proposed a method to extract relations and events in news articles. The contribution to the existing body of knowledge is twofold.

Firstly, our proposed method relies on the Hermes Information Extraction Language (HIEL), i.e., a lexico-semantic pattern language that not only makes use of lexical and syntactical elements, but also employs ontology concepts and relations. These patterns are based on regular expressions, which enhance the expressivity of the rules. In this paper, we have provided a formal syntax for the lexico-semantic rules.

Secondly, in order to show how the proposed rule-based extraction method can be applied in practice, we have implemented the approach in the Hermes News Portal (HNP) as the Hermes Information Extraction Engine (HIEE) plug-in. Combined with standard text preprocessing tasks performed by the GATE framework, as well as a central knowledge base expressed in an OWL ontology, events and relations that occur in news items are extracted.

In order to assess the performance of our proposed method, we have evaluated the implementation by building rules and measuring the performance of the extraction of events and relations by using these rules. On two separate data sets and corresponding ontologies from the financial and political domains, this resulted in a precision of approximately 80% and a recall of 70%, as the lexico-semantic patterns are superior to lexico-syntactic patterns with respect to expressivity. Additional experiments show that, when compared to lexico-semantic rules in JAPE, lexico-semantic HIEL rules obtain higher precision and recall scores than their JAPE equivalents.

Furthermore, our experiments showed that creating lexico-semantic rules requires significantly less time than creating equally performing lexico-syntactic rules, as lexico-semantic rule group creation times were in general one degree of magnitude smaller than lexico-syntactic rule group creation times. We argue that lexico-syntactic rules require more development time because of the larger amount of effort needed for entering the individual literals, resulting in low precision. Also, lexico-semantic rules exploit the inference capabilities of ontologies. This underlines the advantage of using lexico-semantic rules. Moreover, we have demonstrated that lexico-semantic HIEL rules are less verbose than their JAPE equivalents, resulting in less construction time and contributing to higher precision and recall values.

While we have focused on finding new information and identifying events and relations in news articles, as future research we suggest to focus on automatically processing the information that was found and updating the ontology. Additionally, in our approach, we can only extract one triple per rule (the left-hand side of the rule), while events often consist of more than a subject, predicate, and an object. For instance, time can play a role in the event. Also we want to increase the expressivity of our lexico-semantic patterns by making use of the relationships stored in the ontology, or going one step further by employing the expressivity of one-dimensional SPARQL queries. Finally, we would like to investigate how to automatically discover and construct new rules [49] by searching for certain patterns in the text, trained by using relations and events we have previously identified.

Acknowledgements

The authors are partially sponsored by the Dutch Organization for Scientific Research (NWO) Physical Sciences Free Competition project 612.001.009: Financial Events Recognition in News for Algorithmic Trading (FERNAT) and the FES COMMIT Infiniti project Information Retrieval for Information Services.

References

[1] B. M. Gross, The Managing of Organizations: The Administrative Struggle, Vol. 1, Free Press of Glencoe, 1964.

[2] K. R. Rampal, Global Journalism: Survey of International Communication, Longman, 1995, Ch. The Collection and Flow of World News, pp. 35–52.

[3] T. Berners-Lee, J. Hendler, O. Lassila, The Semantic Web, Scientific American 284 (5) (2001) 34–43.

[4] T. R. Gruber, A Translation Approach to Portable Ontologies, Knowledge Acquisition 5 (2) (1993) 199–220.

[5] G. Klyne, J. J. Carroll, Resource Description Framework (RDF): Concepts and Abstract Syntax - W3C Recommendation 10 February 2004, From: `http://www.w3.org/TR/rdf-concepts/` (2004).

[6] D. Brickley, R. Guha, RDF Vocabulary Description Language 1.0: RDF Schema: W3C Recommendation 10 February 2004, From: `http://www.w3.org/TR/rdf-schema/` (2004).

[7] S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, L. A. Stein, OWL Web Ontology Language Reference - W3C Recommendation 10 February 2004, From: `http://www.w3.org/TR/owl-ref/` (2004).

[8] L. Ardissono, L. Console, I. Torre, An Adaptive System for the Personalized Access to News, AI Communications 14 (3) (2001) 129–147.

[9] J. Ahn, P. Brusilovsky, J. Grady, D. He, S. Y. Syn, Open User Profiles for Adaptive News Systems: Help or Harm?, in: 16th International Conference on World Wide Web (WWW 2007), ACM, 2007, pp. 11–20.

[10] D. Billsus, M. J. Pazzani, A Personal News Agent that Talks, Learns and Explains, in: 3rd International Conference on Autonomous Agents (Agents 1999), ACM, 1999, pp. 268–275.

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

[12] K. Schouten, P. Ruijgrok, J. Borsje, F. Frasincar, L. Levering, F. Hogenboom, A Semantic Web-Based Approach for Personalizing News, in: Twenty-Fifth Symposium On Applied Computing (SAC 2010), Web Technologies Track, ACM, 2010, pp. 854–861.

[13] W. IJntema, F. Goossen, F. Frasincar, F. Hogenboom, Ontology-Based News Recommendation, in: International Workshop on Business intelligencE and the WEB (BEWEB 2010) at 13th International Conference on Extending Database Technology and 13th International Conference on Database Theory (EDBT/ICDT 2010), Vol. 426 of ACM International Conference Proceeding Series, ACM, 2010.

[14] C. D. Manning, H. Schütze, Foundations of Statistical Natural Language Processing, 1st Edition, MIT Press, 1999.

[15] A. L. Berger, S. A. D. Pietra, V. J. D. Pietra, A Maximum Entropy Approach to Natural Language Processing, Computational Linguistics 22 (1) (1996) 39–71.

[16] R. K. Taira, S. G. Soderland, A Statistical Natural Language Processor for Medical Reports, in: Annual Fall Symposium of the American Medical Informatics Association (AMIA 1999), American Medical Informatics Association, 1999, pp. 970–974.

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

[18] M. A. Hearst, WordNet: An Electronic Lexical Database and Some of its Applications, MIT Press, 1998, Ch. Automated Discovery of WordNet Relations, pp. 131–151.

[19] P. S. Jacobs, G. R. Krupka, L. F. Rau, Lexico-Semantic Pattern Matching as a Companion to Parsing in Text Understanding, in: Workshop on Speech and Natural Language colocated with the 6th Human Language Technology Conference (HLT 1991), Morgan Kaufmann, 1991, pp. 337–341.

[20] S. Staab, M. Erdmann, A. Maedche, Semantic Patterns, Tech. rep., AIFB, University of Karlsruhe (2001).

[21] S. Staab, M. Erdmann, A. Maedche, Engineering Ontologies Using Semantic Patterns, in: Workshop: E-Business & the Intelligent Web (WEB-1) collocated with the 17th International Joint Conference on Artificial Intelligence (IJCAI 2001), 2001.

[22] Y. Sure, J. Angele, S. Staab, OntoEdit: Multifaceted Inferencing for Ontology Engineering, in: Journal on Data Semantics, Vol. 2800 of Lecture Notes in Computer Science, Springer, 2003, pp. 128–152.

[23] P. Cimiano, S. Staab, Learning by Googling, SIGKDD Explorations Newsletter 6 (2) (2004) 24–33.

[24] S.-H. Hung, C.-H. Lin, J.-S. Hong, Web Mining for Event-Based Commonsense Knowledge Using Lexico-Syntactic Pattern Matching and Semantic Role Labeling, Expert Systems with Applications 37 (1) (2010) 341–347.

[25] F. Frasincar, J. Borsje, F. Hogenboom, E-Business Applications for Product Development and Competitive Growth: Emerging Technologies, IGI Global, 2011, Ch. Personalizing News Services Using Semantic Web Technologies, pp. 261–289.

[26] J. Borsje, F. Hogenboom, F. Frasincar, Semi-Automatic Financial Events Discovery Based on Lexico-Semantic Patterns, International Journal of Web Engineering and Technology 6 (2) (2010) 115–140.

[27] H. Cunningham, D. Maynard, V. Tablan, JAPE: a Java Annotation Patterns Engine, Technical Report CS–00–10, University of Sheffield, Department of Computer Science (2000).

[28] E. Prud'hommeaux, A. Seaborne, SPARQL Query Language for RDF - W3C Recommendation 15 January 2008, From: `http://www.w3.org/TR/rdf-sparql-query/` (2008).

[29] F. Hogenboom, A. Hogenboom, F. Frasincar, U. Kaymak, O. van der Meer, K. Schouten, D. Vandic, SPEED: A Semantics-Based Pipeline for Economic Event Detection, in: Twenty-Ninth International Conference on Conceptual Modeling (ER 2010), Vol. 6412 of Lecture Notes in Computer Science, Springer, 2010, pp. 452–457.

[30] J. Domingue, E. Motta, PlanetOnto: From News Publishing to Integrated Knowledge Management Support, IEEE Intelligent Systems 15 (3) (2000) 26–32.

[31] A. Java, T. Finin, S. Nirenburg, Text Understanding Agents and the Semantic Web, in: 39th Hawaii International Conference on Systems Science (HICSS 2006), Vol. 3, IEEE Computer Society, 2006, p. 62b.

[32] H. Cunningham, GATE, a General Architecture for Text Engineering, Computers and the Humanities 36 (2) (2002) 223–254.

[33] W. J. Black, J. M^{c}Naught, A. Vasilakopoulos, K. Zervanou, B. Theodoulidis, F. Rinaldi, CAFETIERE: Conceptual Annotations for Facts, Events, Terms, Individual Entities, and RElations, Technical Report TR–U4.3.1, Department of Computation, UMIST, Manchester, from: `http://www.nactem.ac.uk/files/phatfile/cafetiere-report.pdf` (2005).

[34] D. Manov, A. Kiryakov, B. Popov, K. Bontcheva, D. Maynard, H. Cunningham, Experiments with Geographic Knowledge for Information Extraction, in: Workshop on Analysis of Geographic References collocated with the 1st Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL 2003), Association for Computational Linguistics, 2003, pp. 1–9.

[35] B. Popov, A. Kiryakov, A. Kirilov, D. Manov, D. Ognyanoff, M. Goranov, KIM - Semantic Annotation Platform, in: 2nd International Semantic Web Conference (ISWC 2003), Vol. 2870 of Lecture Notes in Computer Science, Springer, 2003, pp. 834–849.

[36] B. Popov, A. Kiryakov, D. Ognyanoff, D. Manov, A. Kirilov, KIM - A Semantic Platform for Information Extraction and Retrieval, Journal of Natural Language Engineering 10 (3–4) (2004) 375–392.

[37] S. Soderland, Learning Information Extraction Rules for Semi-Structured and Free Text, Machine Learning 34 (1–3) (1999) 233–272.

[38] D. Maynard, V. Tablan, H. Cunningham, C. Ursu, H. Saggion, K. Bontcheva, Y. Wilks, Architectural Elements of Language Engineering Robustness, Journal of Natural Language Engineering – Special Issue on Robust Methods in Analysis of Natural Language Data 8 (1) (2002) 257–274.

[39] D. Maynard, H. Saggion, M. Yankova, K. Bontcheva, W. Peters, Natural Language Technology for Information Integration in Business Intelligence, in: 10th International Conference on Business Information Systems (BIZ 2007), Vol. 4439 of Lecture Notes in Computer Science, Springer, 2007, pp. 366–380.

[40] H. Saggion, A. Funk, D. Maynard, K. Bontcheva, Ontology-Based Information Extraction for Business Intelligence, in: The Semantic Web, 6th International Semantic Web Conference (ISWC 2007), 2nd Asian Semantic Web Conference (ASWC 2007), Vol. 4825 of Lecture Notes in Computer Science, Springer, 2007, pp. 843–856.

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

[42] G. Krupka, P. Jacobs, L. Rau, L. Childs, I. Sider, GE NLToolset: Description of the System as Used for MUC-4, in: 4th conference on Message Understanding (MUC 1992), Association for Computational Linguistics, 1992, pp. 177–185.

[43] R. H. Robins, General Linguistics, 4th Edition, Longman, 1989.

[44] G. P. Zarri, NKRL, a Knowledge Representation Tool for Encoding the ‘Meaning’ of Complex Narrative Texts, Natural Language Engineering 3 (2) (1997) 231–253.

[45] L. T. F. Gamut, Introduction to Logic, Vol. 1 of Logic, Language, and Meaning, The University of Chicago Press, 1991.

[46] C. Fellbaum, WordNet: An Electronic Lexical Database, MIT Press, 1998.

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

[48] Sun Microsystems, Java Compiler Compiler, From: `https://javacc.dev.java.net/` (2010).

[49] R. Snow, D. Jurafsky, A. Y. Ng, Learning Syntactic Patterns for Automatic Hypernym Discovery, in: 18th Annual Conference on Neural Information Processing Systems (NIPS 2004), Vol. 17 of Advances in Neural Information Processing Systems, MIT Press, 2004, pp. 1297–1304.

Appendix A. Hermes Information Extraction Language Grammar

This section contains a formal grammar description in Extended Backus Naur Form (EBNF) of the Hermes Information Extraction Language (HIEL) that is presented in this paper. Appendix A.1 presents an overview of the non-terminals used in our language, whereas Appendix A.2 summarizes all terminals used in HIEL.

Appendix A.1. Non-Terminals

```
Start        ::=  Lhs SETS Rhs
Lhs          ::=  PL SPACE* LhsP SPACE* COMMA
                  SPACE* LhsP SPACE* (COMMA
                  SPACE* LhsP)? SPACE* PR
Rhs          ::=  Label?  (RhsP | RhsCP) (SPACE+
                  Label?  (RhsP | RhsCP))*
RhsCP        ::=  NOT? PL SPACE* (RhsP | RhsCP)
                  SPACE* ((OR | AND)
                  SPACE*)?  (RhsP | RhsCP)
                  SPACE* PR RepOp?
RhsP         ::=  (NOT? (Element RepOp?))  |
                  WILDCARD
LhsP         ::=  (DOLLAR? Name) | Element
Label        ::=  DOLLAR Name COL_EQ
Element      ::=  String_Lit | SYN | ORTH |
                  Class | Inst
Class        ::=  BL Name BR
Inst         ::=  Name
Name         ::=  Ns?  CHAR (CHAR | NUMBER)*
Ns           ::=  CHAR (CHAR | NUMBER)* COLON
RepOp        ::=  REP | (AL NUMBER (COMMA
                  NUMBER?)?  AR)
String_Lit   ::=  String_Lsq | String_Ldq
String_Lsq   ::=  ‘'’ Seq ‘'’
String_Ldq   ::=  ‘"’ Seq ‘"’
Seq          ::=  (NUMBER | CHAR | HEX | ESC)*
```

Appendix A.2. Terminals

```
NUMBER       ::=  [0-9]+
CHAR         ::=  [A-Z] | [a-z]
HEX          ::=  ‘\x’ ([0-9] | [A-F] | [a-f])
                  ([0-9] | [A-F] | [a-f])
                  (([0-9] | [A-F] | [a-f])
                  ([0-9] | [A-F] | [a-f]))?
ESC          ::=  ‘\'’ | ‘\"’ | ‘\\’
SYN          ::=  ‘CC’ | ‘CD’ | ‘IN’ | ‘JJ’ |
                  ‘NN’ | ‘NNP’ | ‘PP’ | ‘RB’|
                  ‘UH’ | ‘VB’ | ‘VBZ’
ORTH         ::=  ‘upperInitial’ | ‘allCaps’ |
                  ‘lowerCase’ | ‘mixedCaps’
```

```
PL          ::=  ‘(’
PR          ::=  ‘)’
AL          ::=  ‘{’
BL          ::=  ‘[’
BR          ::=  ‘]’
AR          ::=  ‘}’
COMMA       ::=  ‘,’
COLON       ::=  ‘:’
COL_EQ      ::=  ‘:=’
SETS        ::=  ‘:-’
OR          ::=  ‘|’
AND         ::=  ‘&’
NOT         ::=  ‘!’
DOLLAR      ::=  ‘$’
SPACE       ::=  ‘ ’
REP         ::=  ‘+’ | ‘*’ | ‘?’
WILDCARD    ::=  ‘%’ | ‘_’
```