

Semantics-Based Information Extraction for Detecting Economic Events

Alexander Hogenboom, Frederik Hogenboom,
Flavius Frasinca, Kim Schouten, and
Otto van der Meer

Abstract As today's financial markets are sensitive to breaking news on economic events, accurate and timely automatic identification of events in news items is crucial. Unstructured news items originating from many heterogeneous sources have to be mined in order to extract knowledge useful for guiding decision making processes. Hence, we propose the Semantics-Based Pipeline for Economic Event Detection (SPEED), focusing on extracting financial events from news articles and annotating these with meta-data at a speed that enables real-time use. In our implementation, we use some components of an existing framework as well as new components, e.g., a high-performance Ontology Gazetteer, a Word Group Look-Up component, a Word Sense Disambiguator, and components for detecting economic events. Through their interaction with a domain-specific ontology, our novel, semantically enabled components constitute a feedback loop which fosters future reuse of acquired knowledge in the event detection process.

Keywords Event detection · Semantics · Natural language processing · Information extraction

1 Introduction

Communication plays an important role in today's society, as it provides ways to convey messages, typically with a specific goal in mind. Communication can thus facilitate effective, well-informed decision making. Recent decades have shown a tendency of human communication to expand – driven by the increasing popularity of automating processes – such that it also includes human-machine interaction besides purely human interaction. So far, communication between humans and machines has been thwarted by the disability of machines to fully understand complex natural language. Humans have hence adapted their communication with machines by using clearly defined, fixed, and unambiguous morphology, syntax, and semantics. Yet, this only provides limited means of communication. It is the

Econometric Institute, Erasmus University Rotterdam
PO Box 1738, NL-3000 DR, Rotterdam, The Netherlands
{hogenboom, fhogenboom, frasinca}@ese.eur.nl
{288054ks, 276933rm}@student.eur.nl

flexibility and complexity of human language that makes it so expressive. Hence, in order to enable more effective human-machine communication, machines should be able to understand common human language. This is one of the promises of the ongoing research on automated Natural Language Processing (NLP).

In today’s information-driven society, machines that can process natural language can be of invaluable importance. Decision makers are expected to process a continuous flow of (news) messages or any kind of raw data through various input channels, by extracting information and understanding its meaning. Knowledge can then be acquired by applying reasoning to the gathered information. However, the amount of available data is overwhelming, whereas decision makers need a complete overview of their environment in order to enable effective, well-informed decision making. In today’s global economy, this is of paramount importance. Decision makers need an intuition on the state of their market, which is often extremely sensitive to breaking news on economic events like acquisitions, stock splits, or dividend announcements. In this context, the identification of events can guide decision making processes, as these events provide means of structuring information using concepts, with which knowledge can be generated by applying inference. Automating information extraction and knowledge acquisition processes can facilitate or support decision makers in fulfilling their cumbersome tasks, as faster processing of more data enables one to make better informed decisions.

Therefore, we aim to have a fully automated application for processing financial news messages – fetched from Really Simple Syndication (RSS) [43] feeds – in such a way that the essence of the messages is extracted and captured in events that are represented in a machine-understandable way. Thus, in line with the philosophy of the Semantic Web [3], the extracted events can be made accessible for other applications as well, e.g., in order to enable knowledge acquisition. Furthermore, the application should be able to handle news messages at a speed that is sufficient for real-time use, because new events can occur any time and require decision makers to respond in a timely and adequate manner.

We propose a framework (pipeline) that identifies the concepts of interest (i.e., concepts related to economic events), which are defined in a domain ontology and are associated to synsets from a semantic lexicon (WordNet [13]). A preliminary version of this Semantics-based Pipeline for Economic Event Detection (SPEED) has been proposed in [17]. In our current endeavors, we elaborate on this framework by providing a more extensive discussion of the specifics of our framework (e.g., its individual components and algorithms), as well as a more detailed (component-wise) evaluation of its performance. For concept identification, we match lexical representations of concepts retrieved from the text with event-related concepts that are available in WordNet, and thus aim to maximize recall. Here, we use lexico-semantic patterns based on concepts from the ontology. The identified lexical representations of relevant concepts are subject to a procedure for identifying word groups rather than individual words as well as a word sense disambiguation procedure for determining the corresponding sense, in order to maximize precision. In order for our pipeline to be real-time applicable, we also aim to minimize the latency, i.e., the time it takes for a news message to be processed by the pipeline.

Our contributions are two-fold. The first contribution relates to our proposed combination of a number of existing techniques and a number of new components into a novel pipeline for event extraction. As our pipeline is semantically enabled, it is designed to generalize well to other domains, which would typically require

the existing ontology to be replaced by other domain-specific ones. Through their interaction with a domain-specific ontology, our novel, semantically enabled components constitute a feedback loop which fosters future reuse of acquired knowledge in the event detection process. An additional contribution lies in the efficiency and effectiveness of our newly proposed components for identifying relevant ontology concepts, word group look-up, and word sense disambiguation. Our framework, which also builds on previous work on news personalization [6,38], distinguishes itself by means of its fast ontology gazetteer, precise discovery of events using word sense disambiguation, and event decoration with related information using lexico-semantic patterns [5].

This paper is structured as follows. First, Sect. 2 discusses related work. Subsequently, Sect. 3 elaborates on the proposed framework and its implementation. The approach is evaluated in Sect. 4. Last, Sect. 5 concludes the paper and provides directions for future research.

2 Related Work

This section discusses tools that can be used for Information Extraction (IE) purposes. First, we elaborate on SemNews, which is an application that aims at accurately extracting information from heterogeneous news sources. Then, we continue by focusing on IE pipelines.

2.1 SemNews

SemNews [20] is a Semantic Web-based application that aims to discover the meaning of news items. These items are retrieved from RSS feeds and are processed by the NLP engine OntoSem [32]. The engine retrieves Text Meaning Representations (TMR), which are subsequently stored in an ontology (fact repository) that holds as a representation of the world. Results are then published in Ontology Web Language (OWL) [2] format, so that they can be used in Semantic Web applications. This approach is very much related to the work of Vargas-Vera and Celjuska [42], as they present an approach to recognize events in news stories and to populate an ontology semi-automatically.

The information extraction process of OntoSem can be divided into several stages that the application goes through for each news article that is to be analyzed. First, the *Preprocessor* ensures that sentence and word boundaries are identified, as well as named entities, acronyms, numbers, dates, etcetera. Then, the *Syntactic Parser* is invoked to analyze the syntax of the corpus and to resolve syntactic ambiguity. The parsed text is passed through the *Basic Semantic Analyzer*, which produces a basic TMR using various concepts defined in the ontology and copes with resolving semantic ambiguity. Subsequently, there is a phase that is associated with extended analysis, such as resolving referential ambiguity and temporal ordering. Finally, the fact repository is updated by the *Fact Extractor*, using the knowledge stored within the extended TMR.

SemNews seems to suit the approach we aim for well. However, OntoSem employs a frame-based language for representing the ontology and an onomasticon for storing proper names, whereas we envisage an approach in which both the input

ontology and the facts extracted from news items are represented in OWL, as this fosters application interoperability and the reuse of existing reasoning tools. Also, the use of an onomasticon is not sufficient when disambiguating word senses, and hence a general semantic lexicon like WordNet is desired.

2.2 ANNIE

Most IE-focused tools utilize their own framework for information extraction. However, over the last few years, GATE [8,9], a freely available general purpose framework for IE purposes, has become increasingly popular as a basis for IE tools. GATE is highly flexible in that the user can construct natural language processing pipelines from components that perform specific tasks. One can distinguish between various linguistic analysis applications such as tokenization (e.g., distinguishing words), syntactic analysis jobs like Part-Of-Speech (POS) tagging, and semantic analysis tasks such as understanding. By default, GATE loads the A Nearly-New Information Extraction (ANNIE) system, consisting of several key components which can be useful components for many custom natural language processing pipelines.

The first component in the ANNIE pipeline is the *English Tokenizer*, which splits text into separate chunks, such as words and numbers, and takes into account punctuation. The tokenizer is a vital component and other components rely upon its output. The next component is the *Sentence Splitter*, which splits text into sentences. Subsequently, the *POS Tagger* determines the part-of-speech (e.g., noun, verb, etcetera) of words within a scanned corpus. The fourth component in the ANNIE pipeline is the *Gazetteer*, which identifies named entities in the corpus that is processed, such as people, organizations, percentages, etcetera. After defining named entities and after annotating words with their proper POS tags, there could be a need to combine and disambiguate discovered annotations. The fifth component in ANNIE, i.e., the *NE (Named Entity) Transducer*, employs JAPE rules, which only offer limited support to express in a generic way rules geared towards for example combining and disambiguating entities. Finally the last component, the *OrthoMatcher*, adds identity relations between named entities found earlier in the pipeline. Its output can for instance be used for orthographic co-referencing, which is not part of ANNIE.

There are several tools or frameworks that utilize the ANNIE pipeline, or use (modified) ANNIE components together with newly developed components. For instance, Artequakt [23] aims to generate tailored narrative artist biographies using automatically annotated articles from the Web. In their semantic analysis, they employ GATE components for gazetteering and named entity recognition. Another example of a tool that uses ANNIE components is Hermes [6], which extracts a set of news items related to specific concepts of interest. For this purpose, semantically enhanced ANNIE GATE components are used, i.e., they make use of concepts and relations stored in ontologies.

Although the ANNIE pipeline has proven to be useful in various information extraction jobs, its functionality does not suffice when applied to discovering economic events in news messages. For instance, ANNIE lacks important features such as a component that focuses on performing Word Sense Disambiguation (WSD), although some disambiguation can be done using JAPE rules in the *NE Transducer*.

This is however a cumbersome and ineffective approach where rules have to be created manually for each term, which is prone to errors. Furthermore, ANNIE lacks the ability to individually look up concepts from a large ontology within a limited amount of time. Nevertheless, GATE is highly flexible and customizable, and therefore ANNIE's components are either usable, or extendible and replaceable in order to suit our needs.

2.3 CAFETIERE

Besides the Artequakt and Hermes frameworks, another example of an adapted ANNIE pipeline is the Conceptual Annotations for Facts, Events, Terms, Individual Entities, and RElations (CAFETIERE) relation extraction pipeline [4], developed in the Parmenides project [28,36]. The pipeline contains an ontology lookup process and a rule engine. Within CAFETIERE, the Common Annotation Scheme (CAS) DTD is applied, allowing for three annotation layers, i.e., structural, lexical, and semantic annotation. CAFETIERE employs extraction rules defined at lexico-semantic level which are similar to JAPE rules. Nevertheless, the syntax is at a higher level than is the case with JAPE, resulting in easier to express, but less flexible rules.

Because CAFETIERE stores knowledge in an ontology by means of the Narrative Knowledge Representation Language (NKRL), Semantic Web ontologies are not employed. NKRL has no formal semantics and lacks reasoning support, which is desired when identifying for instance financial events. Furthermore, gazetteering is a slow process when going through large ontologies. Finally, the pipeline also misses a WSD component.

2.4 KIM

The Knowledge and Information Management (KIM) platform [33] provides another infrastructure for IE purposes, by combining the GATE architecture with semantic annotation techniques. The back-end and middle layer of the KIM platform focus on automatic annotation of news articles, where named entities, inter-entity relations, and attributes are discovered. For this, it is employed a pre-populated OWL upper ontology, i.e., a minimal but sufficient ontology that is suitable for open domain and general purpose annotation tasks. The semantic annotations in articles allow for applications such as semantic querying and exploring the semantic repository.

KIM's architecture is a conglomeration of three layers. In the back-end, a standard GATE pipeline is invoked for named entity recognition with respect to the KIM ontology. The GATE pipeline is altered in such a way that its components are semantically enabled, and is extended with semantic gazetteers and pattern-matching grammars. Furthermore, GATE is used for managing the content and annotations within the back-end of KIM's architecture. The middle layer of the KIM architecture provides services that can be used by the topmost layer, e.g., semantic repository navigation, semantic indexing and retrieval, etcetera. The topmost layer of KIM embodies front-end applications, such as the *Annotation Server* and the *News Collector*.

The differences between KIM and our envisaged approach are in that we aim for a financial event-focused information extraction pipeline, which is in contrast to KIM’s general purpose framework. Hence, we employ a domain-specific ontology instead of an upper ontology. Also, we specifically focus on extracting events from corpora, and not on (semantic) annotation. Furthermore, no mention has been made regarding WSD within the KIM platform, whereas we consider WSD to be an essential component in an IE pipeline.

2.5 Discussion

Although the approaches to information extraction we discussed so far each have their advantages, they also fail to address some of the issues we aim to alleviate. From a technical point of view, the frameworks incorporate semantics only to a limited extent, which is also demonstrated by Table 1. For instance, they make use of gazetteers or knowledge bases that either do not use ontologies or employ ontologies that are not based on OWL and thus do not make use of existing standards to represent ontologies. Being able to use a standard language as OWL fosters application interoperability and the reuse of existing reasoning tools. Also, to the best of our knowledge, existing applications typically lack a feedback loop, i.e., the acquired knowledge is not used for future information extraction. Furthermore, WSD is absent and the focus often is on annotation, instead of event recognition. Therefore, we aim for a framework that combines the insights gained from the approaches that are previously discussed, targeted at the discovery of financial events in news articles.

Table 1 Comparison of existing approaches and the characteristics required for our current endeavors, based on purpose (*Purpose*), input (*Input*), output (*Output*), knowledge base utilization (*KB utilization*), presence of knowledge base updates (*KB Δ*), and usage of word sense disambiguation (*WSD*)

Approach	Purpose	Input	Output	KB utilization	KB Δ	WSD
SemNews	Fact extraction	RSS	OWL ontology	Frame-based language and an onomasticon for proper names	No	No
ANNIE	Entity detection	Text	Annotations, XML	Looping through gazetteering lists	No	No
CAFETIERE	Entity and relation detection	Text	Annotations, XML	Gazetteering NKRL ontology	No	No
KIM	Entity detection	Text	Annotations, RDF(s) ontology	Gazetteering RDF(s) ontology	Yes	No
Desired	Economic event detection	RSS	Annotations, OWL ontology	Reasoning with OWL ontology and a general semantic lexicon	Yes	Yes

3 Economic Event Detection based on Semantics

The analysis presented in Sect. 2 demonstrates several approaches to automated information extraction from news messages. However, the state-of-the-art in text processing does not enable us to perform the specific task we aim to perform. Current approaches are more focused on annotation of documents, whereas we strive to actually extract information – specific economic events and their related concepts – from documents, with which, e.g., a knowledge base can be updated.

In order to be able to discover economic events in written text, the analysis of texts needs to be driven by semantics, as the domain-specific information captured in these semantics facilitates detection of relevant concepts. Therefore, we propose the Semantics-Based Pipeline for Economic Event Detection (SPEED), consisting of several components which sequentially process an arbitrary document, as visualized in Fig. 1. These components are supported by a semantic lexicon (i.e., WordNet) and a domain-specific ontology.

Due to the potential of the General Architecture for Text Engineering (GATE), we use this IE framework for its modularity. However, none of the existing applications of the general GATE architecture can support the tasks we seek to perform. Even more, no implementation exists of several specialized envisioned components. Therefore, the Java-based implementation of our proposed pipeline requires the development of techniques that support our needs. The default GATE implementations of the *English Tokenizer*, *Sentence Splitter*, *Part-Of-Speech Tagger*, and the *Morphological Analyzer* suit our needs to a limited yet for now sufficient extent.

This section continues by explaining the domain ontology that supports our pipeline in Sect. 3.1. Subsequently, Sects. 3.2 through 3.11 discuss the pipeline’s individual components. We run through the processing steps of the SPEED framework by means of a typical example news item, displayed in Fig. 2. This short news item was extracted at October 9th, 2006 at 20:15:33 hours from the Yahoo! Business and Technology newsfeed and discusses Google’s acquisition of YouTube. In our pipeline, each individual component adds its own annotations to the example news item above. These annotations can be considered as multiple layers on top of the corpus. This means that one word can have multiple annotations, and can also be part of a larger annotation spanning multiple words at the same time.

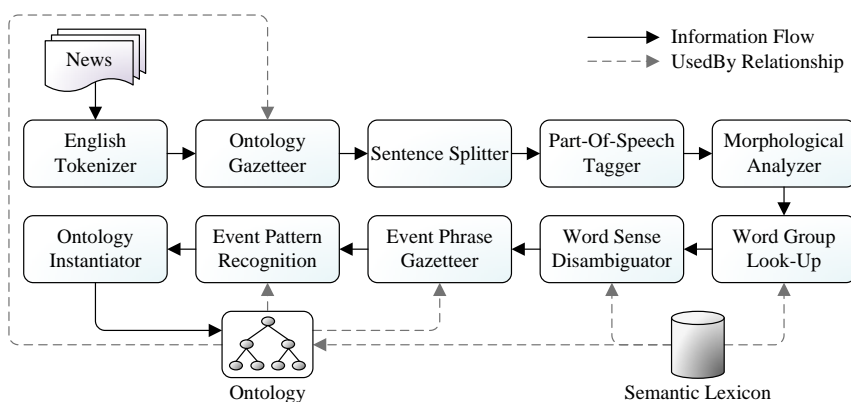


Fig. 1 SPEED design

SAN FRANCISCO (Reuters) - Web search leader Google Inc. on Monday said it agreed to acquire top video entertainment site YouTube Inc. for \$1.65 billion in stock, putting a lofty new value on consumer-generated media sites.

Fig. 2 A typical news example

3.1 Domain Ontology

Our envisaged approach is driven by an ontology containing information on the NASDAQ-100 companies, extracted from Yahoo! Finance. This domain ontology has been developed by domain experts through an incremental middle-out approach. The ontology captures concepts and events concerning the financial domain, e.g., companies, competitors, products, CEO's, etcetera. Many concepts in this ontology stem from a semantic lexicon (i.e., WordNet) and are linked to their semantic lexicon counterparts, but a significant part of the ontology consists of concepts representing named entities (i.e., proper names). In our ontology, we distinguish between ten different financial events, i.e., announcements regarding CEOs, presidents, products, competitors, partners, subsidiaries, share values, revenues, profits, and losses, which are supported by appropriate classes and properties.

We validated our domain ontology using OntoClean [15], a methodology for analyzing ontologies that uses notions for philosophical ontological analysis. OntoClean is based on formal, domain-independent class properties (meta-properties and their modifiers), i.e., identity, unity, rigidity, and dependence. Once annotated with these meta-properties, the ontology can be considered to be valid (or "clean") whenever no constraints are violated that are based on these properties.

3.2 English Tokenizer

SPEED is designed to identify relevant concepts and their relations in a document. To this end, first, individual text components are identified as such using the *English Tokenizer*, which splits text into tokens (e.g., words, numbers, or punctuation) and subsequently applies rules specific to the English language in order to split or merge identified tokens. For example, the token combination `|' |60| |s|` would be merged into one token `|'60s|`. Note that spaces are considered as special tokens and are annotated as a `'SpaceToken'` rather than a `'Token'`. For our running example, this translates to the annotations shown in Fig. 3, where tokens are shaded in medium and light tones (for the sake of clarity) and spaces have a dark shading.

SAN FRANCISCO (Reuters) - Web search leader
 Google Inc. on Monday said it agreed to acquire top
 video entertainment site YouTube Inc. for \$1.65
 billion in stock, putting a lofty new value on
 consumer-generated media sites.

Fig. 3 *English Tokenizer* annotations (tokens)

3.3 Ontology Gazetteer

A first step towards understanding the text is subsequently taken by the *Ontology Gazetteer*, which links concepts in the text to concepts defined in an ontology with relevant concepts (which tend to refer to proper names rather than common words from the semantic lexicon). A normal gazetteer uses lists of words as input, whereas our ontology gazetteer is ontology-driven and scans the text for lexical representations of concepts from the ontology. Matching tokens in the text are annotated with a reference to their associated concepts defined in the ontology. For example, suppose our ontology contains a concept ‘Google’ of type ‘Company’, with a lexical representation ‘Google Inc.’. Any matching ‘Google Inc.’ occurrence in the text is then annotated with the concept ‘Google’.

The default GATE *OntoGazetteer* uses a linear search algorithm to match lexical representations in a text with a list of ontology concepts and their associated lexical representations. However, in our novel *OntoLookup* approach, we use a look-up tree of approximately 5,000 nodes (based on the Yahoo! Finance news messages represented in the ontology), in which possible lexical representations of all relevant concepts in the ontology are mapped to their associated concepts. Each concept can have multiple lexical representations (groups of 1 or more words). These word groups are all represented in the look-up tree. Nodes in the tree represent individual tokens and a path from the root node to an arbitrary leaf node represents a word group.

Fig. 4 depicts a sample tree structure. In this sample, the root node contains – among other things – references to ‘Cisco’, ‘Google’, and ‘Yahoo!’. The ‘Cisco’ token contains a reference to ‘Systems’, which in turn contains a reference to a resource in the ontology, as well as to another token, ‘Inc’. The latter token also contains a reference to a resource in the ontology, but does not contain a reference to another token. Thus, ‘Cisco Systems’, and ‘Cisco Systems Inc’ refer to a concept in the ontology. The paths for ‘Google’ and ‘Yahoo!’ are not fully depicted in Fig. 4, but could exhibit similar characteristics.

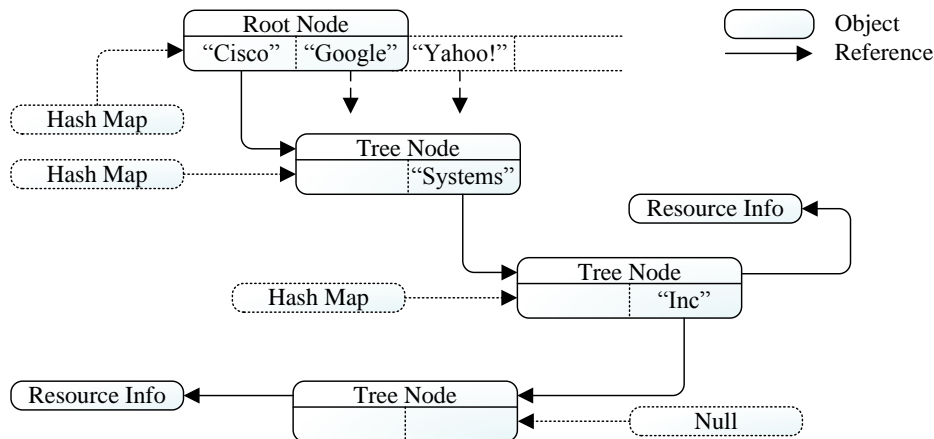


Fig. 4 Sample *OntoLookup* tree structure

SAN FRANCISCO (Reuters) - Web search leader **Google Inc.** on Monday said it agreed to acquire top video entertainment site **YouTube Inc.** for \$1.65 billion in stock, putting a lofty new value on consumer-generated media sites.

Fig. 5 *Ontology Gazetteer* annotations (concepts)

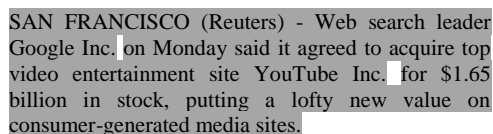
For a given series of tokens, the *OntoLookup* process iterates over the tokens. For each token, it checks whether the look-up tree contains the token. This look-up process starts at the root node of the tree. If the token is not found, the next token in the text is looked up in the root node of the full look-up tree. However, if the token is found, the next token in the text is looked up in the root node of the subtree belonging to the former token. This process is iterated until either a leaf node is reached (i.e., the word group cannot be further expanded), or the root node of the considered subtree does not contain a reference to the next token in the text. The word group associated with the followed path is then annotated with the associated concept from the ontology. The tree is implemented using hash maps, in order to reduce the time needed to traverse the tree. The tree structure representing lexical representations of the concepts in our ontology, indexed using hash maps, is of benefit because matching a token with a child node by using, e.g., a linear search algorithm assessing every child node for a possible match with the token is typically less efficient than determining the index of a child node associated with a token by means of hashing.

When run through the discussed component, several concepts are recognized in our running example. As the text is about two companies, i.e., Google and YouTube, the strings referring to these companies are annotated. These lexical representations are stored within the ontology and are linked to the ontology concepts of the type ‘Company’, which causes the strings to be annotated with ontology concepts ‘Google’ and ‘YouTube’. Figure 5 demonstrates this annotation process, where the highlighted text is annotated with the appropriate ontology concepts.

3.4 Sentence Splitter

Then, the *Sentence Splitter* groups the tokens in the text into sentences, based on tokens indicating a separation between sentences, which can be, for instance, (a combination of) punctuation symbols or new line characters. The grammatical structure of the text is then uncovered in order to facilitate an initial model of the text’s meaning.

As shown in Fig. 6, grouping tokens into sentences is anything but a straightforward task, as periods do not always denote the end of a sentence, but can also be used as for example decimal separators (or in some languages as thousands separators), in abbreviations, etcetera. In the case of our leading example, the *Sentence Splitter* fails to find the correct sentences because of the usage of full stops after ‘Inc’. Later on, this is fixed due to the fact that ‘.’ is part of the lexical representation of a concept. Note that the period inside the value of ‘1.65’ is correctly ignored as a full stop.



SAN FRANCISCO (Reuters) - Web search leader Google Inc. on Monday said it agreed to acquire top video entertainment site YouTube Inc. for \$1.65 billion in stock, putting a lofty new value on consumer-generated media sites.

Fig. 6 *Sentence Splitter* annotations (sentences)

3.5 Part-Of-Speech Tagger

For each sentence, the type of each word token is subsequently determined by the *Part-Of-Speech Tagger*, which tags each word with its part-of-speech. When employing the *Part-Of-Speech Tagger*, no new annotations are added to the document. Instead, features of tokens are added. Tokens already contain information added by the *English Tokenizer* on start and end character number, kind (e.g., word, symbol, etcetera.), length, orthographic category (e.g., lowercase), and the string of characters belonging to the tag. The *Part-Of-Speech Tagger* determines the syntactic category of each token and stores this in a POS feature, which is encoded in capitalized abbreviations. For instance, syntactic categories with suffix ‘VB’ are verbs, e.g., ‘VBZ’ denotes a verb in third person singular present. Categories beginning with ‘NN’ are nouns, such as a single proper noun (‘NNP’). Common syntactic categories are displayed in Table 2.

3.6 Morphological Analyzer

Different forms of a word have a similar meaning; they relate to the same concept, albeit from possibly different perspectives. Therefore, the *Morphological Analyzer* component subsequently reduces the tagged words to their lemma (i.e., canonical form) and when needed a suffix and/or affix denoting the deviation from this lemma. For instance, for the verb ‘walk’, the ‘walks’ morph is annotated as ‘root=walk, suffix=s’. Similar to the *Part-Of-Speech Tagger*, the *Morphological Analyzer* does not add new annotations to the document, but token features. When applicable, the *Morphological Analyzer* adds features related to morphology (such as affixes) to the tokens. At any rate, for each token, the root (lemma) is added.

Table 2 Common syntactic categories

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, third person singular present

3.7 Word Group Look-Up

Words and meanings, denoted often as synsets (set of synonyms) have a many-to-many relationship. A word can have multiple meanings and a meaning can be represented by multiple words. Hence, the next step in interpreting a text is disambiguation of the meaning of the words, given their POS tags, lemmas, etcetera. To this end, first of all, the *Word Group Look-Up* component combines words into maximal word groups, i.e., it aims at assigning as many words as possible to a group representing some concept in a semantic lexicon such as WordNet. We use the complete list of approximately 65,000 existing word groups extracted from WordNet. These word groups are represented in a tree structure, where nodes represent individual tokens and a path from the root node to an arbitrary leaf node represents a word group.

Similarly to the *OntoLookup* process, the word group tree can then be used for matching word groups in the text with word groups extracted from the semantic lexicon. For each set of tokens, the tree is traversed until either a leaf node is reached, or the next token in the text is not in the considered subtree. Again, indexing of the tree is implemented using hash maps, in order to optimize the time needed for traversing the tree in the look-up process.

In our running example, where the feature set of the tokens has been previously extended by the *Part-Of-Speech Tagger* and the *Morphological Analyzer*, the *Word Group Look-Up* module of our pipeline employs the WordNet semantic lexicon in order to identify word groups, such as ‘*SAN FRANCISCO*’. In Fig. 7, the tokens in the text that form a word group are merged into a single token.

3.8 Word Sense Disambiguator

After identifying word groups, the *Word Sense Disambiguator* determines the word sense of each word group by exploring the mutual relations between senses (as defined in the semantic lexicon and the ontology) of word groups; the stronger the relation with surrounding senses, the more likely a sense matches the context. Grouping words is important, because combinations of words may have very specific meanings compared to the individual words. For instance, ‘*Gross Domestic Product*’ is a combination with a unique meaning that is not associated with any of the individual words in this group. The accuracy of WSD may hence be improved when considering word groups rather than individual words.

We propose an adaptation of the Structural Semantic Interconnections (SSI) [31] algorithm for word sense disambiguation. The SSI approach uses graphs to describe word groups and their context (word senses), as derived from a seman-

SAN FRANCISCO (Reuters) - Web search leader Google Inc. on Monday said it agreed to acquire top video entertainment site YouTube Inc. for \$1.65 billion in stock, putting a lofty new value on consumer-generated media sites.

Fig. 7 *Word Group Look-Up* annotations (tokens)

tic lexicon (e.g., WordNet). The senses are determined based on the number and type of detected semantic interconnections in a labeled directed graph representation of all senses of the considered word groups. Similarities are calculated based on an arbitrary distance measure.

More than other common approaches, the SSI approach enables us to incorporate a notion of semantics into the word sense disambiguation process by exploiting a vast semantic lexical database. Other common approaches are typically restricted to a relatively small collection of representations of ontological concepts [41] or barely use any notion of semantics at all, but rather use collocation-based statistical techniques [44] or machine learning techniques [11,27]. Furthermore, SSI is an unsupervised approach, which makes it easy to add new terms as neologisms and jargon for disambiguation (i.e., there is no need of training). Moreover, in recent years, the SSI algorithm has turned out to be a promising and performing word sense disambiguation technique, as it performs better than other state-of-the-art unsupervised WSD methods in the Senseval-3 all-words and the Semeval-2007 coarse-grained all-words competition [30].

Semantic similarity evaluation can be performed on numerous ways using distance measures [21,24,26,35]. Similar to Navigli and Velardi [31], we make use of a simple, transparent, and intuitive distance measure which takes into account the length of paths between words in our semantic lexicon. The shorter a path between two arbitrary words in our semantic lexicon, the more similar we consider them to be.

The word sense disambiguation algorithm we propose in our current endeavors differs from the original SSI algorithm in a number of ways. First, we consider the two most likely senses for each word group and iteratively disambiguate the word group with the greatest weighted difference between the similarity of both senses to the context, rather than the word group with the greatest similarity for its best sense. Intuitively, this should yield better results than the original SSI, as it allows to consider the best separation of the senses of the to-be-disambiguated terms – picking the most similar sense might not be the best option if the similarity difference with respect to the next best sense is small. Furthermore, in case an arbitrary word cannot be disambiguated, we default to the first sense in our semantic lexicon (which in WordNet is statistically the most likely sense), whereas the original SSI algorithm fails to provide any word sense at all in such cases.

For an arbitrary news item, our algorithm (described in Algorithm 1) considers two lists of word groups. The first list d contains all word groups associated with only one sense, according to the semantic lexicon (WordNet), the ontology, and the already disambiguated word groups. The second list a contains all word groups with multiple possibilities for senses, i.e., the word groups to be disambiguated. The algorithm iteratively computes the similarity l of senses c of word groups in the second list to the senses s of word groups in the first list. The higher the similarity of a sense to already disambiguated senses, the more likely this sense is assumed to be correct. The algorithm is initialized in lines 1 through 24. Then, each iteration, each word group in a is assessed by updating the similarity of its senses to s (lines 33 through 36) and identifying its best and second best senses (lines 37 through 45). Additionally, the word group with the greatest difference between the similarity of the best and second best sense (i.e., with the highest confidence) – weighted with respect to the similarity of the best sense – is identified (lines 47 through 52). When all word groups in a have been assessed, the best pick thus

Algorithm 1: Word Sense Disambiguation

```

1   $a, d, s, c, l = \emptyset$ ;
2   $w = \text{getWordGroups}()$ ;
3  foreach  $g$  in  $w$  do
4       $senses = \text{getSenses}(g)$ ;
5      if  $|senses| == 1$  then
6           $\text{add}(d, g)$ ;  $\text{add}(s, senses)$ ;
7      else
8           $\text{add}(a, g)$ ;
9          foreach  $sense$  in  $senses$  do
10             if  $sense \notin c$  then
11                  $\text{add}(c, sense)$ ;
12             end
13         end
14     end
15 end
16 foreach  $sense$  in  $c$  do
17      $simToS = 0$ ;
18     foreach  $knownSense$  in  $s$  do
19          $simToS = simToS + 1/\text{shortestPathLength}(sense, knownSense)$ ;
20     end
21      $\text{add}(l, simToS)$ ;
22 end
23  $lastAddedSense = \emptyset$ ;
24  $disambiguate = \text{true}$ ;
25 while  $disambiguate$  and  $a \neq \emptyset$  do
26      $bestPick, bestPickSense = \emptyset$ ;
27      $bestPickConf = -\infty$ ;
28     foreach  $g$  in  $a$  do
29          $bestSense1, bestSense2 = \emptyset$ ;
30          $bestSim1, bestSim2 = -\infty$ ;
31          $senses = \text{getSenses}(g)$ ;
32         foreach  $sense$  in  $senses$  do
33              $indexSense = \text{indexOf}(c, sense)$ ;
34              $simToS = \text{get}(l, indexSense)$ ;
35              $simToS = simToS + 1/\text{shortestPathLength}(sense, lastAddedSense)$ ;
36              $\text{set}(l, indexSense, simToS)$ ;
37             if  $simToS > bestSim2$  then
38                 if  $simToS > bestSim1$  then
39                      $bestSense2 = bestSense1$ ;  $bestSense1 = sense$ ;
40                      $bestSim2 = bestSim1$ ;  $bestSim1 = simToS$ ;
41                 else
42                      $bestSense2 = sense$ ;
43                      $bestSim2 = simToS$ ;
44                 end
45             end
46         end
47          $confidence = ((bestSim1 - bestSim2) \times bestSim1)$ ;
48         if  $confidence > bestPickConf$  then
49              $bestPick = g$ ;
50              $bestPickSense = bestSense1$ ;
51              $bestPickConf = confidence$ ;
52         end
53     end
54     if  $bestPickConf > 0$  then
55          $\text{rem}(a, \text{indexOf}(a, bestPick))$ ;  $\text{add}(d, bestPick)$ ;  $\text{add}(s, bestPickSense)$ ;
56          $lastAddedSense = bestPickSense$ ;
57     else
58          $disambiguate = \text{false}$ ;
59     end
60 end
61 foreach  $g$  in  $a$  do
62      $\text{rem}(a, \text{indexOf}(a, g))$ ;  $\text{add}(d, g)$ ;  $\text{add}(s, \text{getSense}(g, 1))$ ;
63 end

```

identified is disambiguated by taking the sense with the highest similarity to all disambiguated senses and moving the word group to the list of disambiguated word groups (lines 55 and 56), provided that this similarity is a positive number. In all other cases, the disambiguation process is terminated. If some ambiguous words remain by the time the disambiguation process finishes, our algorithm defaults to selecting their first WordNet sense (lines 61 through 63).

The similarity of a sense to already disambiguated senses is computed as the sum of the inverse of the shortest path length between this sense and the disambiguated senses in the WordNet graph. In our labeled directed graph representation of all senses of the considered word groups, we determine the shortest path between two concepts in a way which is similar to Prim’s algorithm [34] for finding a minimum spanning tree for a connected weighted graph, an algorithm on which Dijkstra’s algorithm [12] is also based. Instead of computing a minimum spanning tree for the entire WordNet graph of the source and target concept, we compute two smaller spanning trees, having the source concept and the target concept as their root. We do this – for both collections – by iteratively walking to all direct neighbors of concepts considered in the collection, until a concept encountered in a walk in one collection has previously been encountered in the other collection.

In our running example, the *Word Sense Disambiguation* component adds the determination of noun and verb senses to the tokens’ feature sets subsequently. These features contain numbers referring to the corresponding WordNet senses. Hence, no new annotations are added.

3.9 Event Phrase Gazetteer

When the meaning of word groups has been disambiguated, the text can be interpreted using semantics introduced by linking word groups to an ontology, thus capturing their essence in a meaningful and machine-understandable way. As we are interested in specific economic events, the *Event Phrase Gazetteer* scans the text for those events. It uses a list of phrases or concepts that are likely to represent some part of a relevant event. For example, when we are looking for stock splits, we can search for ‘`stock split`’. Since the *Word Group Look-Up* component has already combined ‘`stock`’ and ‘`split`’ and the *Word Sense Disambiguator* has already assigned a concept value to this group of words, we can easily match this concept with events in our ontology.

The *Event Phrase Gazetteer* has some similarities with the *Ontology Gazetteer* since both of them try to find data from an ontology in a news message. In contrast to the *Ontology Gazetteer*, the *Event Phrase Gazetteer* takes annotated texts as input. Furthermore, the *Event Phrase Gazetteer* does not process the text lexically, but it looks for concepts, using the sense numbers that are assigned to the words in the text.

The look-up process takes place in two stages. First, the gazetteer is initialized by extracting all events from the ontology and linking them to the proper WordNet senses. This mapping is made accessible through a hash map, where a word sense can be used as a key to retrieve a reference to an event defined in the ontology. Second, at run time, the gazetteer iterates over the words in the text and uses the sense key (if any) to test whether a mapping to a corresponding event exists.

SAN FRANCISCO (Reuters) - Web search leader Google Inc. on Monday **said** it agreed to **acquire** top video entertainment site YouTube Inc. for \$1.65 billion in stock, putting a lofty new value on consumer-generated media sites.

Fig. 8 *Event Phrase Gazetteer* annotations (phrases)

When processing our running example through the *Event Phrase Gazetteer*, we obtain the highlighted annotations – representing key concepts for possible events – shown in Fig. 8. Since there are multiple types of events, in the features of these annotations a specification is given. Both the type of event is added, as well as the URI that points to the specific event in the ontology.

3.10 Event Pattern Recognition

Events thus identified by the *Event Phrase Gazetteer* are supplied with available additional information by the *Event Pattern Recognition* component, which checks whether identified events match certain lexico-semantic patterns (which are then used for extracting additional information related to discovered events). For instance, in case of a stock split, a concept indicating a company should precede the stock split keyword, and either before or just after the stock split keyword, a split-rate concept should be mentioned.

The *Event Pattern Recognition* component is based on the GATE *Rule Transducer* component, which uses JAPE [10] for manually defining patterns. JAPE provides a layer between the user and the regular expressions that are used internally. A typical JAPE rule consists of a pattern that has to be matched, followed by the commands that will be executed when that pattern is matched. These commands most of the time are comprised of a simple annotation command, but more powerful Java code is allowed too in the right hand side of the rule.

The following example of a JAPE rule extracts the proportions associated with a stock split event, e.g., ‘3-for-1’ (three new shares for one old share):

```

1 Rule: Props (({Token.category == CD}) :new
2             ({Token.string == "-"})?
3             ({Token.string == "for"})
4             ({Token.string == "-"})?
5             ({Token.category == CD}) :old)
6 :prop --> :prop.Prop = {rule = "Props",
7                 new = :new.Token.root,
8                 old = :old.Token.root}

```

Lines 1 through 5 define the pattern to be searched for in the text. This pattern should consist of a cardinal number token (representing the number of new shares), followed by an optional ‘-’ token, a ‘for’ token, another optional ‘-’ token, and a cardinal number token (representing the number of old shares). Lines 6 through 8 specify the commands to be executed when the pattern is matched. The results from the pattern (i.e., the number of new shares, ‘new’, and the number of old shares, ‘old’) are stored into an annotation property.

SAN FRANCISCO (Reuters) - Web search leader Google Inc. on Monday said it agreed to acquire top video entertainment site YouTube Inc. for \$1.65 billion in stock, putting a lofty new value on consumer-generated media sites.

Fig. 9 *Event Pattern Recognition* annotations (subject, predicate, and object)

By default, the GATE *Rule Transducer* only allows for simultaneous execution of JAPE rule files. If layering of rules (i.e., using one rule's output as another rule's input) is desired, an extra transducer has to be employed. In our implementation, we tackle this problem by feeding a JAPE rule file to the transducer that is nothing but a table of contents containing an ordered list of the different rule files that have to be executed. In this way, layering is possible, without being obliged to have multiple transducers in the pipeline. In addition to this, it enables easy recycling of useful blocks of rules.

In our running example, the *Ontology Gazetteer* already identified a subject and an object, namely 'Google Inc.' and 'YouTube Inc.', but those are not the subject and object of the sentence in a linguistic sense. To find the linguistic subject and object, the company names are merged with the surrounding nouns, adjectives, and determiners. This is also done for verbs. For instance, 'acquire' indicates a buy event, but in order to have a better understanding of the sentence, the *Event Pattern Recognition* component annotates the predicate of the sentence by merging the 'VerbEvent' annotation with the surrounding verbs, resulting in the annotations depicted in Fig. 9.

Subsequently, after merging subjects, objects, and predicates, JAPE rules are matched to the annotated text. Whenever there is a match, the event pattern is executed, resulting in event annotations, e.g., 'BuyEvent', 'DeclarationEvent', etcetera. The annotation holds URIs to all important features of this event, including event type, event actors, and time stamp (derived from the news message). Figure 10 shows the final event annotation.

3.11 Ontology Instantiator

Finally, the knowledge base can be updated by inserting the identified events and their extracted associated information into the ontology using the *Ontology Instantiator*. At this phase, event instances are fully annotated in the text, which implies that no additional corrections need to be made. The module first retrieves a reference to the ontology by using the Jena [19] library and then iterates over

SAN FRANCISCO (Reuters) - Web search leader Google Inc. on Monday said it agreed to acquire top video entertainment site YouTube Inc. for \$1.65 billion in stock, putting a lofty new value on consumer-generated media sites.

Fig. 10 *Event Pattern Recognition* annotations (events)

the available event annotations. Each time an event annotation is processed, an event instance is created in the ontology which belongs to a specific event class. Annotation features that are available are stored as properties of the individual. Furthermore, (relations between) concepts affected by the event are updated in the ontology. When the plug-in finished execution, the ontology is again updated as it is enriched with new events originating from the processed text.

In the running example used throughout this section, we do not have to deal with a buy event, as an upcoming acquisition has only been announced. Therefore, a ‘`DeclarationEvent`’ individual with its associated properties is created within the ontology. The relations between ‘`Google`’ and ‘`YouTube`’ can remain unchanged within the ontology. However, some of their properties are updated so that the ontology reflects Google’s upcoming acquisition of YouTube.

4 Evaluation

In order to evaluate the performance of the implementation, we assess the quality of the individual pipeline components, each of which contributes to the output of the pipeline – i.e., annotations and events – and the pipeline as a whole. We measure the performance by means of statistics that describe, where applicable, latency and the cumulative error in terms of precision and recall. We define precision as the part of the identified elements (e.g., word senses or events) that have been identified correctly, and recall represents the number of identified elements as a fraction of the number of elements that should have been identified. When we compare the performance of different approaches, we assess the statistical relevance of differences in performance by means of a paired, two-sided Wilcoxon signed-rank test [14, 18], which is a non-parametric test evaluating the null hypothesis that the differences between paired observations are symmetrically distributed around a median equal to 0. If this null hypothesis is rejected, the compared samples are significantly different. This test would be suitable in this experimental setup, as the distribution of the values to be compared is unknown.

In our evaluation, we mainly focus on a data set consisting of 200 news messages extracted from the Yahoo! Business and Technology newsfeeds. In order to arrive at a golden standard, we have let three domain experts manually annotate the economic events and relations that we take into account in our evaluation, while ensuring an inter-annotator agreement of at least 66% (i.e., at least two out of three annotators agree). We distinguish between ten different financial events, i.e., announcements regarding CEOs, presidents, products, competitors, partners, subsidiaries, share values, revenues, profits, and losses. Our data set contains 60 CEO and 22 president discoveries, 232 statements linking companies with their products, partners, and subsidiaries, i.e., 136, 50, and 46, respectively, and 127 announcements of share values (45), revenues (22), profits (33), and losses (27).

Some components in our pipeline are existing, well-tested components, the performance of which has already been demonstrated in an extensive body of literature. However, one of the contributions of our current endeavors is that we propose several novel components that require a more detailed evaluation in terms of performance. The first component we evaluate in this respect is our *Ontology Gazetteer* component with our *OntoLookup* method, the performance of which we compare to the performance of the default GATE *OntoGazetteer* it replaces. The

goal of both components is to identify lexical representations of concepts defined in an ontology. Precision and recall are not particularly useful here, as exact lexical representations known a priori (as is the case here) can always be identified in our corpus. Conversely, the latency is a more important issue in this component. On average, the *OntoGazetteer* needs 1.137 milliseconds (with a standard deviation of 0.265 milliseconds) per document to identify ontology concepts, whereas our *OntoLookup* method completes the same task in approximately 0.213 milliseconds (with a standard deviation of 0.039 milliseconds) per document. This significant 81% decrease (Wilcoxon p -value equals 0.000) in execution time needed can be attributed to the employed hash map trees.

Another newly proposed component utilizing hash map trees is our *Word Group Look-Up* component, which aims to identify compound words (i.e., word groups) in each document. If we do not use hash map trees in this component, but instead attempt to maximize our word groups by making numerous calls to our semantic lexicon in a linear search procedure, we need on average 68 milliseconds (with a standard deviation of 25 milliseconds) per document in our Yahoo! Business and Technology corpus for our task. Conversely, when we implement our proposed approach utilizing hash map trees, execution time needed decreases significantly with 46% (Wilcoxon p -value equals 0.000) to, on average, 37 milliseconds, with a standard deviation of 16 milliseconds.

Our *Word Sense Disambiguator* can be evaluated on a large, publicly available corpus designed specifically for this purpose – SemCor [29]. We consider all 186 syntactically and semantically tagged SemCor documents containing 192,639 nouns, verbs, adjectives, and adverbs, which have been annotated with their associated POS, lemma, and WordNet sense. On this corpus, the original SSI word sense disambiguation algorithm exhibits an average precision of 53% with a standard deviation of 5 percentage points, a recall of 31% with a standard deviation of 9 percentage points, and an average execution time of 1,966 milliseconds, with a standard deviation of 755 milliseconds. Conversely, our proposed adaptation of SSI exhibits an average precision and recall of 59% with a standard deviation of 5 percentage points, as well as an average execution time of 2,050 milliseconds, with a standard deviation of 796 milliseconds. This implies an overall improvement in precision with 12% and an improvement in recall with 90% in terms of the performance of the original SSI algorithm, while experiencing a mere 4% increase in execution time, which is just a matter of milliseconds. All observed differences are statistically significant, as they are all associated with a Wilcoxon p -value of 0.000, yielding a rejection of the null hypothesis of no difference between performance measures at a significance level of 0.001.

On our data set, our pipeline exhibits a latency of 632 milliseconds per document, with a standard deviation of 398 milliseconds. As for the output of the pipeline as a whole, we observe a precision for the concept identification in news items of 86% and a recall of 81%, which is comparable with existing systems. Table 3 shows the reported precision and recall for entity recognition for several existing information extraction tools, together with SPEED’s scores. Scores for other approaches are extracted from existing literature, as the individual tools are optimized for different purposes and therefore employ different data sets. As the evaluated data sets are different for each analyzed approach, the results presented in the table can merely be used as an indication of comparable performance, yet the table still underlines that in terms of precision and recall, SPEED’s perfor-

Table 3 Overview of the reported entity recognition precision and recall scores for several existing algorithms and information extraction pipelines

Pipeline	Precision	Recall
SemNews [20,39]	68.00%	68.00%
ANNIE [8]	85.00%	85.00%
CAFETIERE [4,37]	84.03%	84.13%
KIM [33]	86.00%	82.00%
SPEED	86.00%	81.00%

mance is similar to existing (related) approaches. It should be noted that precision and recall of pipeline outputs, i.e., fully decorated events, result in lower values of approximately 62% and 53%, as they rely on multiple concepts that have to be identified correctly. To our knowledge, none of the existing approaches decorates identified events with their related information. As such, we cannot compare the final outputs of the considered approaches, as each approach in Table 3 has been designed for a distinct task.

Errors in concept identification result from missing lexical representations of the knowledge base concepts, and missing concepts in general. The disambiguator is supported by the *Word Group Look-Up* module, which identifies groups of nouns and verb phrases using WordNet. As a result of storing all data in a data base to keep it ready to use for future look-up, the more often the disambiguator is invoked, the faster the execution times will be (as concept similarities have been previously computed), thus eliminating a potential bottleneck. Despite using only WordNet as a semantic lexicon, we obtain high precision as many of our concepts’ lexical representations are named entities, which often are monosemous. High recall can be explained by SPEED’s focus on detecting concepts from the ontology in the text, rather than on identifying all concepts in the text. The senses of word groups that are not present in the ontology are only used to help in the disambiguation of existing (already identified) concept lexical representations. The senses of the word groups not present in the ontology are not reflected in the precision and recall measures, as these measures only relate to identified ontological concepts (and their disambiguated senses).

5 Conclusions and Future Work

We have proposed the Semantics-Based Pipeline for Economic Event Detection (SPEED), which aims to extract financial events from news articles (announced through RSS feeds) and to annotate these with meta-data, while maintaining a speed that is high enough to enable real-time use. For implementing the SPEED pipeline we have reused some of the ANNIE GATE components and developed new ones such as a high-performance gazetteer, word group look-up component, and word sense disambiguator. Although we focus on the financial domain, SPEED is generalizable to other domains, as we separate the domain-specific aspects from the domain-independent ones.

We have introduced a couple of novelties into the pipeline. Our pipeline components are semantically enabled, i.e., they make use of semantic lexicons and ontologies. Also, our WSD component employs a semantic lexicon (WordNet). Furthermore, the pipeline outputs results with semantics, which introduces a feedback

loop; the knowledge base used within the pipeline can be updated when events are discovered, so that it represents the current state of the world. We thus incorporate learning behavior, making event identification more adaptive. Hence, the merit of our pipeline is in the use of semantics, enabling broader application interoperability. Other contributions lie within the speed of gazetteering and the improvements made to an existing word sense disambiguation algorithm (SSI). These novelties contribute to improved precision and recall.

However, since our framework is designed to deal with natural language, it may encounter noisy linguistic information. Our current framework is able to parse standard terms (which can be found in WordNet), as well as compound terms (which we identify by means of our novel word group look-up component). As future work, we aim to implement jargon terms by exploiting, e.g., Wikipedia redirects. Additionally, we plan to account for nonsense terms (i.e., misspellings) by using a similarity measure such as the Levenshtein distance. Alternatively, more extensive experiments regarding semantic similarity evaluation [21, 24, 26, 35] are subject to future research, e.g., experiments with other similarity measures such as concept neighborhood, which is also applied in related domains [40], show promising results that could also be beneficial for our work.

Furthermore, research into the development of event trigger-based update languages [25] for domain ontologies would be a fruitful direction. Another suggestion for future research is to investigate event extraction rules learning from text using intelligent techniques (such as genetic algorithms). More interesting avenues for future work lie in investigating further possibilities for implementation in algorithmic trading environments [1, 7, 16, 22]. We aim to find a principal way of utilizing discovered events in this field. To this end, we also envision another addition, i.e., a way of linking sentiment (trends, moods, and opinions) to discovered events in order to assign more meaning to these events that can be exploited in an algorithmic trading setup. Sentiment of actors with respect to events may be the driving force behind their reactions to these events.

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

References

1. Allen, F., Karjalainen, R.: Using Genetic Algorithms to Find Technical Trading Rules. *Journal of Financial Economics* **51**(2), 245–271 (1999)
2. Bechhofer, S., van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D., Patel-Schneider, P., Stein, L.: OWL Web Ontology Language Reference (2004). From: <http://www.w3.org/TR/owl-ref/>
3. Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web. *Scientific American* **284**(5), 34–43 (2001)
4. Black, W.J., McNaught, J., Vasilakopoulos, A., Zervanou, K., Theodoulidis, B., Rinaldi, F.: CAFETIERE: Conceptual Annotations for Facts, Events, Terms, Individual Entities, and Relations. Technical Report TR-U4.3.1, Department of Computation, UMIST, Manchester (2005)
5. Borsje, J., Hogenboom, F., Frasinca, F.: Semi-Automatic Financial Events Discovery Based on Lexico-Semantic Patterns. *International Journal of Web Engineering and Technology* **6**(2), 115–140 (2010)

6. Borsje, J., Levering, L., Frasinca, F.: Hermes: A Semantic Web-Based News Decision Support System. In: Proceedings of the 23rd Annual ACM Symposium on Applied Computing, pp. 2415–2420. ACM (2008)
7. Brock, W.A., Lakonishok, J., LeBaron, B.: Simple Technical Trading Rules and the Stochastic Properties of Stock Returns. *Journal of Finance* **47**(5), 1731–1764 (1992)
8. Cunningham, H.: GATE, a General Architecture for Text Engineering. *Computers and the Humanities* **36**(2), 223–254 (2002)
9. Cunningham, H., Maynard, D., Bontcheva, K., Tablan, V.: GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. In: Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL 2002), pp. 168–175. Association for Computational Linguistics (2002)
10. Cunningham, H., Maynard, D., Tablan, V.: JAPE: a Java Annotation Patterns Engine (Second Edition). Research Memorandum CS-00-10, Department of Computer Science, University of Sheffield (2000)
11. Decadt, B., Decadt, B., Hoste, V., Daelemans, W., van den Bosch, A.: GAMBL, Genetic Algorithm Optimization of Memory-Based WSD. In: R. Mihalcea, P. Edmonds (eds.) Proceedings of the 3rd ACL/SIGLEX International Workshop on the Evaluation of Systems for the Semantic Analysis of Text (senseval-3), pp. 108–112. Association for Computational Linguistics (2004)
12. Dijkstra, E.W.: A Note on Two Problems in Connexion with Graphs. *Numerische Mathematik* **1**, 269–271 (1959)
13. Fellbaum, C.: WordNet an Electronic Lexical Database. *Computational Linguistics* **25**(2), 292–296 (1998)
14. Gibbons, J.: Nonparametric Statistical Inference. *Technometrics* **28**(3), 275 (1986)
15. Guarino, N., Welty, C.A.: Evaluating Ontological Decisions with OntoClean. *Communications of the ACM* **45**(2), 61–65 (2002)
16. Hellstrom, T., Holmstrom, K.: Parameter Tuning in Trading Algorithms using ASTA. In: Proceedings of the 6th International Conference Computational Finance (CF 1999), pp. 343–357. MIT Press (1999)
17. Hogenboom, F., Hogenboom, A., Frasinca, F., Kaymak, U., van der Meer, O., Schouten, K., Vandic, D.: SPEED: A Semantics-Based Pipeline for Economic Event Detection. In: J. Parsons, M. Saeki, P. Shoval, C.C. Woo, Y. Wand (eds.) Proceedings of the 29th International Conference on Conceptual Modeling (ER 2010), *Lecture Notes in Computer Science*, vol. 6412, pp. 452–457. Springer (2010)
18. Hollander, M., Wolfe, D.: Nonparametric Statistical Methods. *Journal of the American Statistical Association* **95**(449), 333 (2000)
19. HP Labs: Jena - A Semantic Web Framework for Java (2009). From: <http://jena.sourceforge.net/>
20. Java, A., Finin, T., Nirenburg, S.: SemNews: A Semantic News Framework. In: Proceedings of the 21st National Conference on Artificial Intelligence (AAAI 2006), pp. 1939–1940 (2006)
21. Jiang, J.J., Conrath, D.W.: Semantic Similarity Based on Corpus Statistics and Lexical Taxonomy. In: Proceedings of the International Conference on Research in Computational Linguistics (ROCLING X), pp. 19–33 (1997)
22. Kearns, M.J., Ortiz, L.E.: The Penn-Lehman Automated Trading Project. *IEEE Intelligent Systems* **18**(6), 22–31 (2003)
23. Kim, S., Alani, H., Hall, W., Lewis, P.H., Millard, D.E., Shadbolt, N.R., Weal, M.J.: Artequakt: Generating Tailored Biographies from Automatically Annotated Fragments from the Web. In: Proceedings of the Workshop on Semantic Authoring, Annotation & Knowledge Markup (SAAKM 2002), pp. 1–6 (2002)
24. Lin, D.: An Information-Theoretic Definition of Similarity. In: Proceedings of the 15th International Conference on Machine Learning (ICML 1998), pp. 296–304. Morgan Kaufmann (1998)
25. Lösch, U., Rudolph, S., Vrandečić, D., Studer, R.: Tempus Fugit. In: L. Aroyo, P. Traverso, F. Ciravegna, P. Cimiano, T. Heath, E. Hyvönen, R. Mizoguchi, E. Oren, M. Sabou, E.P.B. Simperl (eds.) Proceedings of the 6th European Semantic Web Conference (ESWC 2009), *Lecture Notes in Computer Science*, vol. 5554, pp. 278–292. Springer (2009)
26. Maguitman, A.G., Menczer, F., Roinestad, H., Vespignani, A.: Algorithmic Detection of Semantic Similarity. In: Proceedings of the 14th International Conference on the World Wide Web (WWW 2005), pp. 107–116. ACM (2005)

27. Mihalcea, R., Csomai, A.: SenseLearner: Word Sense Disambiguation for All Words in Unrestricted Text. In: Proceedings of the 43th Annual Meeting of the Association for Computational Linguistics (ACL 2005), pp. 53–56. Association for Computational Linguistics (2005)
28. Mikroyannidis, A., Mantes, A., Tsalidis, C.: Information Management: The Parmenides Approach. In: B. Theodoulidis, C. Tsalidis (eds.) Proceedings of the International Workshop on Text Mining Research, Practice and Opportunities collocated with the 2nd International Conference on Recent Advances in Natural Language Processing (RANLP 2005), pp. 23–31 (2005)
29. Miller, G., Chodorow, M., Landes, S., Leacock, C., Thomas, R.: Using a Semantic Concordance for Sense Identification. In: Proceedings of the Human Language Technology Workshop (HLT 1994), pp. 240–243. Association for Computational Linguistics (1994)
30. Navigli, R.: Word Sense Disambiguation: A Survey. *ACM Computing Surveys* **41**(2) (2009)
31. Navigli, R., Velardi, P.: Structural Semantic Interconnections: A Knowledge-Based Approach to Word Sense Disambiguation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **27**(7), 1075–1086 (2005)
32. Nirenburg, S., Raskin, V.: Ontological Semantics, Formal Ontology, and Ambiguity. In: Proceedings of the International Conference on Formal Ontology in Information Systems (FOIS 2001), pp. 151–161. ACM (2001)
33. Popov, B., Kiryakov, A., Ognyanoff, D., Manov, D., Kirilov, A.: KIM - A Semantic Platform For Information Extraction and Retrieval. *Journal of Natural Language Engineering* **10**(3–4), 375–392 (2004)
34. Prim, R.C.: Shortest Connection Networks and Some Generalisations. *Bell System Technical Journal* **36**, 1389–1401 (1957)
35. Resnik, P.: Using Information Content to Evaluate Semantic Similarity in a Taxonomy. In: Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI 1995), pp. 448–453 (1995)
36. Rinaldi, F., Dowdall, J., Hess, M., Ellman, J., Zarri, G.P., Persidis, A., Bernard, L., Karanikas, H.: Multilayer annotations in Parmenides. In: Proceedings of the Workshop on Knowledge Markup and Semantic Annotation (SemAnnot 2003), pp. 33–40 (2003)
37. Rinaldi, F., Schneider, G., Kaljurand, K., Dowdall, J., Andronis, C., Persidis, A., Konstanti, O.: Mining Relations in the GENIA Corpus. In: T. Scheffer (ed.) Proceedings of the 2nd European Workshop on Data Mining and Text Mining for Bioinformatics collocated with the 15th European Conference on Machine Learning (ECML 2004) and the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD 2004), pp. 61–68 (2004)
38. Schouten, K., Ruijgrok, P., Borsje, J., Frasincar, F., Levering, L., Hogenboom, F.: A Semantic Web-Based Approach for Personalizing News. In: M.J. Palakal, C. Hung (eds.) Twenty-Fifth Symposium On Applied Computing (SAC 2010), Web Technologies Track, pp. 854–861. ACM (2010)
39. Sergei Nirenburg and Stephen Beale and Marjorie McShane: Baseline Evaluation of WSD and Semantic Dependency in OntoSem. In: J. Bos, R. Delmonte (eds.) Proceedings of the 1st STEP workshop on Semantics in Text Processing (STEP 2008), *Research in Computational Semantics*, vol. 1, pp. 179–192. College Publications (2008)
40. Taddesse, F.G., Tekli, J., Chbeir, R., Viviani, M., Yetongnon, K.: Semantic-based Merging of RSS Items. *World Wide Web Journal, Internet and Web Information Systems*, Special Issue on Human-Centered Web Science **13**(1–2), 169–207 (2009)
41. Theobald, M., Schenkel, R., Weikum, G.: Exploiting Structure, Annotation, and Ontological Knowledge for Automatic Classification of XML Data. In: V. Christophides, J. Freire (eds.) Proceedings of the 6th International Workshop on the Web and Databases (WebDB 2003), pp. 1–6. IEEE Computer Society (2003)
42. Vargas-Vera, M., Celjuska, D.: Event Recognition on News Stories and Semi-Automatic Population of an Ontology. In: Proceedings of the 3rd IEEE/WIC/ACM International Conference on Web Intelligence (WI 2004), pp. 615–618 (2004)
43. Winer, D.: RSS 2.0 Specification (2003). From: <http://cyber.law.harvard.edu/rss/rss.html>
44. Yuret, D.: Some experiments with a Naive Bayes WSD System. In: R. Mihalcea, P. Edmonds (eds.) Proceedings of the 3rd ACL/SIGLEX International Workshop on the Evaluation of Systems for the Semantic Analysis of Text (senseval-3), pp. 265–268. Association for Computational Linguistics (2004)