A Genetic Programming Approach for Learning Semantic Information Extraction Rules from News

Wouter IJntema, Frederik Hogenboom, Flavius Frasincar, and Damir Vandic

Erasmus University Rotterdam
PO Box 1738, NL-3000 DR
Rotterdam, the Netherlands

wouterijntema@gmail.com, {fhogenboom, frasincar, vandic}@ese.eur.nl

Abstract. Due to the increasing amount of data provided by news sources and the user specific information needs, recently, many news personalization systems have been proposed. Often, these systems process news data automatically into information, while relying on underlying knowledge bases, containing concepts and their relations for specific domains. For this, information extraction rules are frequently used, yet they are usually manually constructed. As it is difficult to efficiently maintain a balance between precision and recall, while using a manual approach, we present a genetic programming-based approach for automatically learning semantic information extraction rules from (financial) news that extract events. Our evaluation results show that compared to information extraction rules constructed by expert users, our rules yield a 27% higher F_1-measure after the same amount of rules construction time.

1 Introduction

The immense growth of the World Wide Web in the past decades resulted in enormous amounts of data that are readily available to the average user. Many researchers have hence developed ways to convert these vast amounts of data into valuable information. The Semantic Web aims to organize the currently unstructured Web. While to a human reader, the meaning of text is easily interpretable, machines interpret an average document currently found on the Web as a random collection of characters, without properly associating meaning to it. The Semantic Web is organized by using numerous languages and technologies to convey meaning. A structural means for representing Web information is provided by ontologies.

An ontology is formally defined as a specification of a conceptualization and can be used to store domain-specific knowledge in the form of concepts with various relations between them. Utilizing ontologies within Web information systems allows one to perform searches based on these concepts and relations. Often,

ontologies are used as a knowledge base to support the information-intensive operations. An example of a finance-oriented Web information system is the Hermes framework [10]. Its ontology consists of lexicalized concepts that exist in the financial domain and is used for the classification of Web news items, as well as querying.

A major problem in such news processing frameworks is that in order to turn news item data into information, a knowledge engineer has to keep up with the incoming data and process it in order to determine the value of the data. The ontology needs to be kept up-to-date in an efficient and timely manner. For this, user-defined patterns can be used that extract information that is needed for ontology updating. Usually, lexico-syntactic patterns are employed [11], yet the problem with these type of patterns is that they are based on syntactical elements, such as Part-of-Speech (POS) tags, and thus do not make use of the available domain semantics. Their application is hence often limited to hypernym (generalization), hyponym (specialization), meronym (part of), and holonym (the whole) relations.

In order to overcome the aforementioned problems with common information extraction patterns, in earlier work, we have proposed the lexico-semantic Hermes Information Extraction Language (HIEL) [12], which makes use of lexical and syntactical elements as well as semantic elements. While the use of information extraction rules allows for semi-automatic information extraction, the construction of the patterns remains a non-trivial, tedious, and time-consuming process, because a trade-off needs to be made between the rules' precision and recall. Therefore, in this paper, we propose a method that assists the construction of information extraction rules. Additionally, the method is implemented and evaluated on a data set containing news, while employing the learned rules for fact extraction (relations between concepts) within the financial domain. In this research, we consider facts to represent (financial) events like acquisitions, profit announcements, CEO changes, etc., which are captured by triples consisting of a subject, a predicate, and an object.

The remainder of this paper is organized as follows. First, we discuss related work in Sect. 2. Next, we introduce our information extraction pattern language and our rule learning framework in Sects. 3 and 4, respectively. Subsequently, our implementation is discussed in Sect. 5 and Sect. 6 presents the performance evaluation of our algorithm. Last, we draw conclusions and discuss some directions for future work in Sect. 7.

2 Related Work

In the mature field of pattern learning, there is a lot of work previously done. Hence, we discuss only work that is closely related to ours, i.e., learning patterns for information extraction. This section elaborates on a small, yet representative, part of the current body of knowledge, ranging from work as old as 1992 to more recent work from 2010.

In the early 1990's, Hearst [11] showed that simple lexico-syntactic patterns can be used to extract hyponyms from text. While the patterns of Hearst generated high precision, they did not perform well recall-wise, hence, driving the development of new pattern languages that were (loosely) based on the patterns of Hearst. For example, in 2002, the authors of [7] proposed JAPE rules for information extraction which are nowadays widely employed, e.g., in the work of Maynard et al. [14]. Other applications that use patterns for general purpose information extraction are presented in for example [2], while [8] uses patterns specifically for news processing, analogous to Hermes, our news processing framework [10]. Even though some of the aforementioned pattern languages incorporate some semantics, they could easily result in verbose rules and they could become rather complex [12]. Hence, we defined our own information extraction language which we incorporate in the Hermes news processing framework, i.e., the Hermes Information Extraction Language (HIEL) [12].

Since the composition of information extraction rules is a tedious process which requires a domain expert to invest a lot of time, a vast amount of effort has been put into automation of this process. We distinguish between supervised and unsupervised learning, where in the former method a model is learned from data of which the correct outcomes (classifications) are known, while the latter method does not rely on any prior knowledge. Due to the fact that on free text supervised methods generally perform better compared to unsupervised methods [6], we aim to employ a supervised learning technique. A problem many supervised approaches have to deal with, is the sparse amount of training examples, for which bootstrapping has proven to be an effective solution [4].

In [16] hypernym relations are learned from text using a supervised learning technique. The authors collect noun pairs from a corpus in order to identify new hypernym pairs, and for each of these pairs sentences are gathered in which both nouns occur. New hypernym classifiers are trained based on patterns extracted from the gathered sentences, using classifiers like Naïve Bayes, genetic algorithms, and logistic regression. When such methods are applied in rule learning processes, rules are generated randomly during initialization and are altered in such a way that the built rules perform better in terms of a predefined metric, which is often a combination of precision and recall. With respect to rule generalization and specialization, the authors of [16] distinguish between top-down and bottom-up approaches. The first type starts with a general rule and then aims to specialize it, while the latter starts with a specialized rule which is then generalized. Our approach goes beyond the one from [16] by allowing domain concepts and relationships to be extracted from text.

WHISK [17] employs a supervised top-down rule learning method. The rules learned in this system are based on regular expressions, which is similar to our approach. In addition to simple literals, syntactic and semantic tags are used to generalize the rules. In the learning process, these tags are determined by means of heuristics. While classes are allowed, no is-a hierarchy or other relationships are employed, while at the heart of our system is a domain ontology with both concepts and relations, which is used to create generic lexico-semantic patterns.

KNOWITALL [9] uses an unsupervised bottom-up approach in order to extract named-entities from the Web. The system employs patterns that incorporate POS tags to extract new information. Pattern learning is based on Web searches, where for each occurrence of an instance, a prefix of a specific amount of words and a suffix of a number of words is added to the pattern. The learned patterns consist only of an entity surrounded by words, unlike our approach, which employs a larger amount of linguistic information like orthographical categories and ontology elements, and not only POS tags. Furthermore, the expressiveness of the learned patterns appears to be limited, since repetition and logical operators are not allowed. KNOWITALL focuses on learning named entity extraction patterns rather than on the extraction of new relationships between entities, which is something we pursue.

While many of the above methods have proven to be effective when using lexico-syntactic rules, Genetic Algorithms (GA) are suitable for rule learning as well, since the input is often a bit string. One can encode a pattern such that every bit represents a token or its corresponding features. By employing different genetic operators, such as inheritance, selection, mutation, and cross-over, the optimal information extraction rule can be determined. A similar method is applied in [5] for learning lexico-syntactic patterns, that only incorporate POS tags, in order to extract entities, which is different from our approach, since we aim to generate lexico-semantic patterns to extract concepts, relationships, and events (complex concepts) from text.

A branch of Genetic Algorithms is Genetic Programming (GP), where generally each problem is represented as a tree instead of a bit string. This makes it easier to encode the problem. Each node either represents a sequence, a logical operator – e.g., conjunction, disjunction, and negation – or a repetition operator. Similarly, terminal tree nodes are suitable to represent a literal, syntactic category, orthographical category, or a concept. Genetic algorithms often converge fast to a good solution when compared to other meta-heuristics, such as simulated annealing [18]. By performing the default genetic operators, trees can evolve until the desired performance is achieved. In a similar manner [3] employs trees to represent rules, containing POS tags, that are used in genetic programming to discriminate between definitions and non-definitions in text. Because of the identified advantages of Genetic Programming approaches over other approaches, in our research, we use a Genetic Programming approach to pattern learning.

3 HIEL: the Hermes Information Extraction Language

The Hermes Information Extraction Language (HIEL) has been extensively described and evaluated in earlier work [12]. This section provides an overview of the basic constructs of HIEL, which are captured by Fig. 1. The latter figure shows an example rule that links CEOs to their companies. Lexical and syntactic elements are indicated by white labels, whereas semantic elements (which make use of a domain ontology) are indicated by shaded labels.

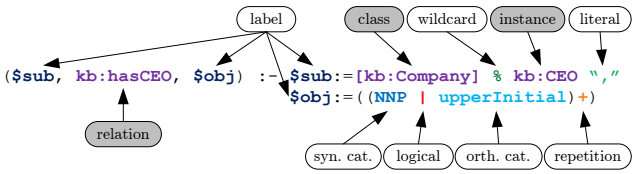

Fig. 1. Example HIEL rule

In HIEL, a rule typically consists of a left-hand side (LHS) and a right-hand side (RHS). Once the pattern on the RHS has been matched, it is used in the LHS, consisting of three components, i.e., a subject, predicate, and an object, where the predicate describes the relation between the subject and the object (in this case `kb:hasCEO`). The RHS supports sequences of many different features, as explained below.

First, labels (preceded by `$`) on the RHS associate sequences using assignment (`:=`) to the correct entities specified on the LHS. Second, syntactic categories (e.g., nouns, verbs, etc.) and orthographical categories (i.e., token capitalization) can be employed. Next, HIEL supports the basic logical operators *and* (`&`), *or* (`|`), and *not* (`!`), and additionally allows for repetition (regular expression operators, i.e., `*`, `+`, `?`, and {...}). Moreover, wildcards are also supported, allowing for ≥ 0 tokens (`%`) or exactly 1 token (`_`) to be skipped.

Of paramount importance is the support for semantic elements through the use of ontological classes, which are defined as groups of individuals that share the same properties, i.e., the instances of a class. A concept (class or instance) or relationship may consist of several lexical representations that are stored using the synonym property in the (lexicalized) domain ontology. The hierarchical structure of the ontology allows the user to make rules either more specific or generic, depending on the needs at hand.

4 Rule Learning

In order to assist domain experts with rule creation, we propose to employ a genetic programming approach to rule learning. Our information extraction language, HIEL, which can intuitively be implemented using tree structures, fits the required tree structure of the genetic programming operators. Additionally, a genetic programming approach offers transparency in the sense that it gives the user insight into how information extraction rules are learned. Also, a genetic approach – as opposed to other meta-heuristics such as simulated annealing – often converges to a good solution in a relatively small amount of time [18].

4.1 Rule Learning Process

Figure 2 depicts the basic steps of our genetic programming approach to rule learning. First rules are initialized, followed by the evaluation of the fitness of each of these rules. Rule evolution is done by applying a genetic operator

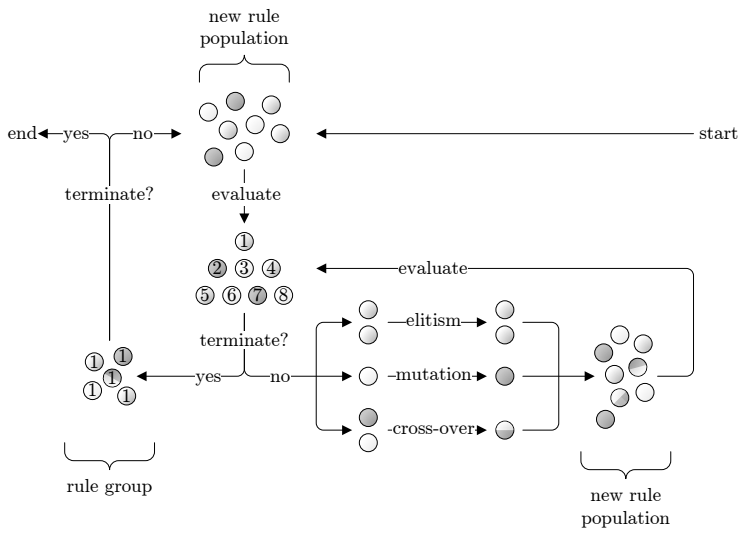

Fig. 2. Rule learning process

on the rules, such as elitism, cross-over, and mutation. Based on a selection procedure which takes into account the fitness of individuals we determine the rules on which these operators are applied. This process continues until one of the termination criteria (see Subsect. 4.7) is fulfilled, after which the rule with the highest fitness is collected in a rule group. The latter is a group of rules, in which each rule aims to extract the same type of information, albeit covering different situations. Since a single rule is not likely to achieve a high recall, because of the many different sentence structures, a collection of rules could achieve this goal. After the best rule, i.e., the rule with the highest fitness, has been selected and the rule group does not yet meet its termination requirements, a new population is initialized and another iteration is performed.

4.2 Representation

While in genetic algorithms, individuals are generally encoded in the form of an array of bits, in genetic programming individuals are specified as trees. In our representation, a tree consists of functions and terminals. Functions have functions and terminals as children, whereas terminals cannot have child nodes. We differentiate between five functions, i.e., a sequence, a conjunction operator, a disjunction operator, a negation operator, and a repetition. Also, we distinguish four terminals, i.e., a syntactic category, an orthographic category, a concept, and a wildcard.

Each information extraction rule can be represented by a tree. As HIEL requires labels to be placed on separate elements on the first level of the tree and each label should be bound to different tokens in the text, the root of each tree is a sequence, which can have one or more function or terminal child nodes.

4.3 Initialization

The first phase in the genetic programming process is the initialization of the rules. During the initialization, a population of N individuals is created. For initialization, each node needs to be created such that it is syntactically correct. In addition, a maximum number of nodes per tree and a maximum tree depth helps constraining the rule size and complexity.

Generally, in genetic programming, information extraction rules are generated randomly at initialization phase. A commonly used method is ramped-half-and-half, which is proven to produce a wide variety of trees of various sizes and shapes. The ramped-half-and-half initialization procedure consists of two methods, i.e., *full* and *grow*. The full method generates trees for which the leaves (terminal nodes) are all at the same level (i.e., $maxdepth$), while the grow method generates more variously shaped trees. Because neither of the methods provide a wide variety of individuals, half of the population is constructed using the full method and half of the population is constructed using the grow method.

4.4 Fitness Evaluation

Each individual in the population is evaluated for each generation in order to determine its fitness. We compare the extracted information with manually annotated information by evaluating the F_1-measure and the number of nodes within a tree. The F_1-measure is defined as the harmonic mean of precision (the correctly found items) and recall (the correctly found items with respect to the should-be-found items). We calculate the number of nodes within a tree in order to control the amount of bloat (i.e., uncontrolled growth of information extraction rules during the evolutionary process) in the population. Both measures are combined into one fitness measure that determines how well an individual performs compared to others, where F_1-scores of longer rules are penalized more than those of rules showing less bloat.

A common problem in genetic programming is tree size explosion. Often, rules are learned that have the same fitness, but that are slightly different. In order to overcome the problem of learning rules consisting of unnecessary nodes, we introduce some parsimony pressure by including a small penalty in the overall fitness measure for the total number of nodes of the rule. Let α denote the amount of bloat (optimized later on) and R represent a rule, then the fitness of a rule (when taking into account both F_1 and rule length l) is determined as:

$$Fitness(R) = \begin{cases} 0 & \text{if } F_1(R) = 0 \\ F_1(R) - \alpha \cdot l(R) & \text{if } F_1(R) > 0 \ . \end{cases} \tag{1}$$

4.5 Selection

For each genetic operator one or more individuals from the population need to be selected. According to the Darwinian principles, the strongest individuals survive, therefore it is better to select individuals based on their fitness. A common

selection method is tournament selection. One of the advantages of this method is that the selection pressure, which determines the degree to which it favors fit individuals over less fit individuals, remains constant. In tournament selection, ts (tournament size) individuals are selected randomly from the population, and the individual with the highest fitness is selected. By adjusting the tournament size, the selection pressure can be adapted.

4.6 Genetic Operations

After the rules have been initialized, the actual process of evolving can be initiated. During the evolution, several genetic operators are applied, i.e., elitist selection, cross-over, and mutation.

Elitist Selection. The first operation, elitist selection, resembles the survival of the fittest principle from Darwin. After the fitness of each individual in the population has been determined, the best r performing individuals are selected and copied to the next generation. The user may alter the portion of the population that is allocated for selection. Generally r is set to a value between 5% and 10%, in order for the algorithm to keep just a small set of the best performing individuals.

An advantage of applying the selection operator is that it helps the process to remember the best performing individuals until a better one is found. If the operator is omitted, these well performing rules might disappear from the population due to the cross-over and mutation operators.

Cross-over. During the cross-over operation two parents are selected from the population to produce either one or two offsprings. The former method randomly selects a cross-over point in both parents and interchanges the selected nodes, producing two children. The latter also randomly selects a cross-over point in both parents, but generates one child by combining the selected parts from both parents. Each parent is chosen based on its fitness using tournament selection and could be selected more than once in each generation, making it possible to use the same individual for multiple cross-over operations.

The selection of the cross-over points is generally not done with uniform probability, since the majority of the nodes will be terminal nodes. In order to overcome this problem, we select 90% of the time a function and 10% of the time a terminal node. While individuals are selected based on their fitness, the nodes interchanged during cross-over are selected in a random manner. This can result in offspring that do(es) not necessarily perform well, while the originating trees can have a relatively good performance. This is the case if a node (including its child nodes), also called a subpattern, is almost never discovered in the text.

Mutation. The mutation operator aims to introduce more variety into the population. Several approaches are identified in mutation for genetic programming. The first is subtree mutation, also known as headless chicken cross-over, where a

random point in the tree is replaced by a randomly generated subtree. A second approach is point mutation, where only the randomly selected point is replaced by a function or terminal. If no replacement is possible (i.e., if the randomly generated node is not allowed within the selected parent node), the mutation is not performed. We implement the headless chicken cross-over method, because of its reported good performance with respect to the other approaches [1, 13].

4.7 Termination Criteria

A genetic programming run terminates when one of the termination criteria is satisfied. In our system we have implemented two termination criteria, one for a run and one for a rule group. Each run generates a certain maximum number of generations, which can be specified by the user. Because of the wide variety in sentence structures it is not plausible that one rule would be able to achieve high recall and precision values, yet a group of rules might be able to achieve this goal for a particular event. Once a termination criterion has been fulfilled, the rule with the highest fitness is saved to the assembled rule group. This group is a set of rules that intend to extract the same information (i.e., triple type). For example, it is likely that one needs several rules to extract all instances of the CEO relationship that has been mentioned in examples earlier. If the triple to extract is defined by *Company hasCEO Person*, at least two rules are needed to extract both the instance in "*Apple's chief executive, Steven P. Jobs*" and "*Steve Ballmer, Microsoft's chief executive*" as the order of the company and the CEO is different in these two cases.

Once a rule is learned and added to the rule group, the information extracted by this rule is excluded while learning additional rules. If a rule does match a previous annotation, it is not taken into account for its fitness, and hence each rule will extract different information. After the termination criterion for the current population fires, the rule with the highest fitness is only collected in the rule group if it causes the rule group to achieve a higher overall fitness value. In case it lowers the fitness of the entire group, it is omitted. The entire rule learning process, i.e., assembling the rule group, terminates when T iterations of updates have passed in a sequence, which did not manage to produce rules that increased the fitness of the rule group, meaning the algorithm is stuck in a (local, possibly sub-optimal) solution.

5 Implementation

We implemented our information extraction language and rule learning approach in the Hermes framework [10], which can be found at `http://people.few.eur.nl/fhogenboom/hermes.html`. At the core of the Hermes framework lies a lexicalized financial domain ontology which specifies domain concepts and their relationships. The implementation, the Hermes News Portal (HNP), provides components for the import and classification of the news articles extracted from various RSS news feeds. During this process the classifier adds annotations, such

as syntactical categories, orthographical categories, and concepts, to the text which can subsequently be used for creating and matching information extraction rules. The details of this process can be found in our previous work [10].

In our rule learning environment, the user is able to keep track of the current generation, the learned rules, and their fitness. Several controls are put in place for managing the rule learning process. Additionally, current generations and learned rules are displayed. Last, the user is able to fine-tune the algorithm parameters.

6 Evaluation

To evaluate the performance of our information extraction language and the genetic programming approach to automatic rule learning, we have selected 500 news articles from the Web with an average length of 700 words from the financial and technology domain originating from various sources, including New York Times, Reuters, Washington Post, and Businessweek. Each news item has been processed using Hermes, with at the back-end a knowledge base containing over 1,200 concepts, including companies, persons, products, financial terms, etc. The learned rules are employed for fact extraction (relations between concepts, i.e., triples that denote an event) within the financial domain.

Three domain experts have annotated the documents, while distinguishing between ten different financial relations, such as profits, products, CEOs, and competitors of companies. In order to decrease the amount of subjectivity we have used a democratic voting principle for the selection of annotations, meaning two out of three annotators should have proposed the annotation to consider it valid. As displayed in Table 1, this resulted in an average Inter-Annotator Agreement (IAA) of 71% for 1,153 unique annotations among all the relations. The table shows that there is a clear difference between the different relations. For instance, the *Competitor* relation is often subjective and therefore hard to determine whether a clear competitor relationship is stated in the text. The same can be argued for the *Partner* relation, which indicates a partnership between two companies. This is in contrast to, for instance, the *CEO* relation, which is often indicated by words like "CEO", "chief", or "chief executive".

Furthermore the table shows the number of annotations per relation found by the annotators in the set of 500 news items. While the knowledge experts have selected subjects and objects that appeared in separate sentences, which is shown in the second column of Table 1, we have made a selection of annotations for which the subject and object appeared in the same sentence, displayed in the third column. The reason for doing this, is that restricting it to finding relations in a single sentence speeds up the algorithm significantly, while losing only a small portion of the annotations. In future work we intend to experiment with matching a rule onto several sentences, instead of just one. This may also increase the recall, because it often occurs that the subject and the object lie within a certain range from each other, while such an approach still takes less computation time compared to matching the full news item.

Table 1. Inter-Annotator Agreement (IAA) for each of the 10 considered relationships

Relation	Articles	Sentences	IAA
Competitor	157	126	0.62
Loss	56	31	0.67
Partner	61	59	0.63
Subsidiary	115	97	0.63
CEO	161	135	0.83
President	64	58	0.68
Product	344	300	0.73
Profit	68	46	0.72
Sales	45	20	0.78
ShareValue	82	77	0.78
Total	1153	949	0.71

Using a hill-climbing procedure, we optimized our algorithm parameters. When learning rules using the genetic programming algorithm with ramped-half-and-half initialization, tournament selection (with a tournament size of 0.25), and a population size of 100, a tree depth of 3 and a maximum amount of children of 7 yielded the best results. Here, the mutation rate and elitism rate are 0.3 and 0.05, respectively, whereas the bloat parameter α equals 0.001, making it only effective for situations where F_1-measures are approximately the same. The group size equals 10, and in our optimal configuration, we only allow for $T = 50$ generations with the same fitness values. Also, during rule learning, we put an emphasis on precision scores with $\beta = 0.3$ for F_β, i.e., an increase in precision is considered to be more important than an increase in recall.

The results of the evaluation are presented in Table 2, which underlines that, when compared to a full manual approach to rule creation, the use of genetic programming for rule learning can be useful for the considered relations within our evaluated financial domain. The learned rules are used for extracting relations between subjects and objects (facts), i.e., both subject and object have to be correctly identified, as well as the other components used in the rules. Small errors in classification of individual tokens (words) easily disrupt relation detection. Correct classification of relations thus is less trivial than regular named entity recognition, leading to lower results than one would initially expect [10].

For automatic rule learning, the *CEO* relation performs best with a precision, recall, and F_1-measure of 90%. In a similar manner rules are learned for the *President* and *Product* relations. For the latter relation we obtain a rule group with a precision and recall of 79%, yielding a 79% F_1-measure. For the *President* relation, we measure a precision and recall of 82% and 79%, respectively, resulting in a slightly higher F_1-measure of 80%. The *President* relation hence performs slightly worse than the *CEO* relation, even though the structure of text is somewhat similar. This may be caused by the lower number of annotations for the *President* relation. In addition, we have shown in Table 1 that the IAA for this relation is slightly lower compared to the *CEO* relation.

Table 2. Precision, recall, and F_1 scores for all 10 financial relations (rule groups) after 5 hours of automatic rule learning (left) and manual creation (right)

	Automatic Learning			Manual Creation			
Relation	Precision	Recall	F_1-measure	Precision	Recall	F_1-measure	Δ%
Competitor	0.667	0.508	0.577	0.875	0.280	0.424	36.0%
Loss	0.905	0.613	0.731	0.818	0.333	0.474	54.3%
Partner	0.808	0.356	0.494	0.450	0.391	0.419	18.0%
Subsidiary	0.698	0.309	0.429	0.611	0.239	0.344	24.8%
CEO	0.904	0.904	0.904	0.824	0.700	0.757	19.5%
President	0.821	0.793	0.807	0.833	0.455	0.588	37.2%
Product	0.788	0.793	0.791	0.862	0.596	0.704	12.3%
Profit	0.960	0.522	0.676	1.000	0.273	0.429	57.7%
Sales	0.900	0.450	0.600	0.455	0.455	0.455	32.0%
ShareValue	0.939	0.805	0.867	0.530	0.778	0.631	37.5%
Total	0.839	0.605	0.703	0.726	0.450	0.555	26.6%

For the *Competitor*, *Subsidiary*, and *Partner* relations, the precision, recall, and F_1-measure are lower in comparison with the aforementioned relations, approximately ranging between 40% and 60%. This could be caused by the fact that both the subject and the object of these relations are expected to be of type *Company*, while for other types of relation – e.g., *Product* and *CEO* – the subject and object are of different types, increasing the importance of finding contextual concepts that specifically describe the relation at hand. Additionally, in retrospect, the structure of the sentences in our data describing such relations is more complex than for other relations. In order to find more suitable patterns, the patterns need to be more complex by, for instance, adding more *and* and *not* operators, with the risk of overfitting. Future work should therefore focus on determining how patterns can be learned from more complex sentences, by for instance pre-analyzing the rules for often returning concepts and increasing the probability of appearance for these concepts during initialization and mutation.

The remaining relations, i.e., *Loss*, *Profit*, *Sales*, and *ShareValue* are all data properties, meaning they do not require a concept for the object of the relation. Examples of the data property values are "10.5 million euros", "$12", or "53 thousand yen". In order to match those values one may need a complex pattern, and hence we decided to use the classification component of Hermes to annotate currency values as a single token. For example, the string "10.5 million euros" is annotated with a single annotation, e.g., *CurrencyValue*, which can be used in the information extraction rules. This allows us to treat these data properties in a similar manner as the object properties.

Last, the results for automatic rule generation depicted in Table 2 show that among the data properties *ShareValue* achieved the highest F_1-value, i.e., 87%, followed by the *Loss* relation, which measured an F_1-value of 73%. The *Profit* and *Sales* relations performed slightly worse, resulting in F_1-measures between 60% and 70%.

Our experiments show that the used fitness function – defined in (1) – is expensive because the F_1-measure has to be calculated for each rule in each generation of a population, and is heavily dependent on available computing power. On our machine, using an Intel® 2.66 GHz Core™ i7 920 processor with 6 GB of RAM, jobs finished within 5 hours each. On average, the generation of a rule group representing a relation takes approximately 4 and a half hours. The largest amount of time needed for one rule group was 5 hours, whereas the smallest amount of time required was 3 and a half hours.

We also let a domain expert create rules manually for 5 hours per rule group on the same machine to ensure a fair comparison of our automatic system with the manual creation of rules. Again, most time is consumed by evaluating rules, yet a manual approach is less efficient. Where the genetic programming approach generates precision, recall, and F_1-values of 84%, 61%, and 70%, respectively, on average, the manually created rule groups show lower performances. For manual rule creation, the resulting F_1-values are on average about 27% lower (displayed under Δ% in the rightmost column of Table 2). Hence, within the same amount of time (i.e., 5 hours per rule group), a domain expert manually writing rules would end up with worse performing rules than an automated genetic programming-based approach. We do not question the potential quality of the rules manually created by the experts when allowing for more time, yet within the limited amount of time advantages of automatic generation are clearly shown. We do, however, observe similar performance patterns as have been described above.

The largest improvements (up to 58%) we observe for relations that involve data properties that deal with more complex constructions (e.g., using datatype variants), which are cumbersome for human experts to include in their rules, hence leading to lower recall. For example, *Loss* and *Profit* involve complex sentences with currencies, which have many different variants in our data set. On the other hand, rule groups that cover many structurally homogeneous examples for which the subject and object are concepts having different types, e.g., *Product* and *CEO*, show improvements as low as 12%, as these are straightforward to implement for domain experts, thus diminishing the need for automation.

For the domain expert, the actual writing took up a few percent of the total time (5 to 10 minutes). A considerable amount of time was used for reading news messages, analyzing matched patterns, verifying results, etc. Additionally, perfecting rules took up increasingly more time, as one needs to abstract away from the given examples in the training set. When increasing the training set size, it would become virtually impossible for domain experts to keep up with a genetic programming-based approach, underlining the added value for automatic rule generation for detecting complex semantic relations in large data sets.

7 Conclusions

Answering to the need for ontology update languages, in this paper we have introduced the Hermes Information Extraction Language (HIEL). The language supports many of the features that exist in regular expressions, such as sequences,

literals, logical operators, repetition, and wildcards. In addition to this, syntactic and orthographic categories are supported. In order to allow the user to create generic information extraction rules for a domain we made use of semantic entities in rules by employing ontological elements, such as classes and instances.

Information extraction rules are often used in automatic information extraction, yet they are usually manually constructed. We have presented a genetic programming-based approach for automatically learning these rules from financial news. Genetic programming approaches provide rules expressed in a user-understandable way, and usually find adequate solutions within a reasonable amount of time. In general, our system performs good in terms of recall and precision, and hence also yields good F_1-values of 70% across all considered financial relations. Our experiments show that compared to information extraction rules constructed by expert users, we are able to find rules that yield a higher F_1-value (i.e., 27% higher on average) after the same amount of time (i.e., 5 hours). A frequently encountered problem for the genetic programming approach is that the quality of the initial population is too low, because the probability that the right concepts are initially chosen becomes smaller as the total number of concepts in the knowledge base increases.

As future work, we aim to investigate solutions to the aforementioned problem, e.g., by implementing heuristics and bootstrapping our algorithms. We hypothesize that frequently appearing concepts in a certain domain can be given a higher probability during initialization to increase the quality of the initial population. Moreover, manually derived rules can be useful when deployed in the initial population. Also, we plan to extend our evaluation to also include single rule matching on multiple sentences. Additional directions are extracting other types of information (from different domains than the financial domain, such as the political, medical, and weather domains), as well as connecting automatic rule learning with (semi-)automated ontology updating mechanisms [15] that process the extracted information and update ontologies accordingly.

Acknowledgment

The authors are partially supported by the NWO Physical Sciences Free Competition project 612.001.009: Financial Events Recognition in News for Algorithmic Trading (FERNAT), the Dutch national program COMMIT, and the NWO Mozaiek scholarship project 017.007.142: Semantic Web Enhanced Product Search (SWEPS).

References

1. Angeline, P.J.: Subtree Crossover: Building Block Engine or Macromutation? In: 2nd Ann. Conf. on Genetic Programming (GP 1997). pp. 9–17. Morgan Kaufmann (1997)
2. Black, W.J., M^{c}Naught, 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, UMIST (2005)

3. Borg, C., Rosner, M., Pace, G.J.: Automatic Grammar Rule Extraction and Ranking for Definitions. In: 7th Int. Conf. of Language Resources and Evaluation (LREC 2010). European Language Resources Association (2010)
4. Carlson, A., Betteridge, J., Wang, R.C., Hruschka Jr., E.R., Mitchell, T.M.: Coupled Semi-Supervised Learning for Information Extraction. In: 3rd Int. Conf. on Web Search and Data Mining (WSDM 2010). pp. 101–110. ACM (2010)
5. Castellanos, M., Gupta, C., Wang, S., Dayal, U.: Leveraging Web Streams for Contractual Situational Awareness in Operational BI. In: Int. Workshop on Business intelligencE and the WEB (BEWEB 2010) in conjunction with EDBT/ICDT 2010 Joint Conf. pp. 1–8. ACM (2010)
6. Chang, C.H., Kayed, M., Girgis, M.R., Shaalan, K.: A Survey of Web Information Extraction Systems. IEEE Transactions on Knowledge and Data Engineering 18(10), 1411–1428 (2006)
7. Cunningham, H., Maynard, D., Bontcheva, K., Tablan, V.: 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). pp. 168–175. Association for Computational Linguistics (2002)
8. Domingue, J., Motta, E.: PlanetOnto: From News Publishing to Integrated Knowledge Management Support. IEEE Intelligent Systems 15(3), 26–32 (2000)
9. Etzioni, O., Cafarella, M., Downey, D., Popescu, A., Shaked, T., Soderland, S., Weld, D.S., Yates, A.: Unsupervised Named-Entity Extraction From The Web: An Experimental Study. Artificial Intelligence 165(1), 91–134 (2005)
10. Frasincar, F., Borsje, J., Hogenboom, F.: E-Business Applications for Product Development and Competitive Growth: Emerging Technologies, chap. Personalizing News Services Using Semantic Web Technologies, pp. 261–289. IGI Global (2011)
11. Hearst, M.A.: Automatic Acquisition of Hyponyms from Large Text Corpora. In: 14th Conf. on Computational Linguistics (COLING 1992). vol. 2, pp. 539–545 (1992)
12. IJntema, W., Sangers, J., Hogenboom, F., Frasincar, F.: A Lexico-Semantic Pattern Language for Learning Ontology Instances from Text. J. of Web Semantics: Science, Services and Agents on the World Wide Web 15(1), 37–50 (2012)
13. Jones, T.: Crossover Macromutation and Population-based Search. In: 6th Int. Conf. on Genetic Algorithms (ICGA 1995). pp. 73–80. Morgan Kaufmann (1995)
14. Maynard, D., Saggion, H., Yankova, M., Bontcheva, K., Peters, W.: Business Information Systems, Lecture Notes in Computer Science, vol. 4439, chap. Natural Language Technology for Information Integration in Business Intelligence, pp. 366–380. Springer (2007)
15. Sangers, J., Hogenboom, F., Frasincar, F.: Event-Driven Ontology Updating. In: 13th Int. Conf. on Web Information System Engineering (WISE 2012). Lecture Notes in Computer Science, vol. 7651, pp. 44–57. Springer (2012)
16. Snow, R., Jurafsky, D., Ng, A.Y.: Learning Syntactic Patterns for Automatic Hypernym Discovery. In: 18th Ann. Conf. on Neural Information Processing Systems (NIPS 2004). Advances in Neural Information Processing Systems, vol. 17, pp. 1297–1304. MIT Press (2004)
17. Soderland, S.: Learning Information Extraction Rules for Semi-Structured and Free Text. Machine Learning 34(1–3), 233–272 (1999)
18. Thompson, D.R., Bilbro, G.L.: Comparison of a Genetic Algorithm with a Simulated Annealing Algorithm for the Design of an ATM Network. IEEE Communications Letters 4(8), 267–269 (2000)