2010 22nd International Conference on Tools with Artificial Intelligence

Ontology-guided Extraction of Complex Nested Relationships

Sushain Pandit, Vasant Honavar
Department of Computer Science
Iowa State University
Ames, IA 50014, USA
Email: sushain.pandit.isu@gmail.com

***Abstract*—Many applications call for methods to enable automatic extraction of structured information from unstructured natural language text. Due to inherent challenges of natural language processing, most of the existing methods for information extraction from text tend to be domain specific. We explore a modular ontology-based approach to information extraction that decouples domain-specific knowledge from the rules used for information extraction. We describe a framework for extraction of a subset of complex nested relationships (e.g., Joe reports that Jim is a reliable employee). The extracted relationships are output in the form of sets of RDF (resource description framework) triples, which can be queried using query languages for RDF and mined for knowledge acquisition.**

I. Introduction

Many text-based applications call for automatic extraction of structured information from unstructured natural language text. There are two main approaches to information extraction [13]: (i) knowledge-based methods that rely on handcrafted rules provided by experts and (ii) machine learning methods that use rules discovered from an annotated corpus. While machine learning methods have been effective on tasks like entity extraction [11], they have had limited success in extracting complex relationships, in large part, due to the lack of annotated training corpus. Most of the existing knowledge based methods for information extraction [2]–[6], [8]–[10] rely on extraction rules that encode domain-specific knowledge. Against this background, we present a knowledge-based approach to information extraction that decouples domain-specific knowledge (encoded in a domain ontology) from domain-independent rules that capture linguistic structures. We focus on extracting complex nested relationships (e.g., "Wall Street Journal reported that Apple shares are bound to take a hit as a result of increasing competition from other smart phone manufacturers.") from text. The rest of this paper is organized as follows. Section II covers the preliminaries. Section III describes our approach to domain-independent extraction of complex nested relationships. Section IV describes the results of our evaluation of the proposed approach. Section V concludes with a summary.

II. Preliminaries

Definition 1: (*Domain Ontology*) A domain ontology is encoded by a 5-tuple $\mathcal{DOI} = (\mathbb{R}, \mathbb{C}, \mathbb{Y}, h, r)$ where: $\mathbb{Y}$ is a set of instances in the domain, $\mathbb{C}$ is a set of concepts defined over $\mathbb{Y}$, and $\mathbb{R}$ is a set of relations defined over $\mathbb{Y} \times \mathbb{Y}$, h is a function that takes as input, an instance $y \in \mathbb{Y}$ and returns a set of concepts in $\mathbb{C}$ that contain the instance y; and r is a function that takes as input, an ordered pair of instances (x, y) and returns the set of relations in $\mathbb{R}$ that contain (x, y).

Definition 2: (*Dependency Graph)* Given an English language sentence T comprising of a word-set W representing the words in the sentence, a dependency graph, G of the sentence T is defined as a directed graph with node-set W and a labeled edge-set connecting the nodes in W s.t. for any two connected nodes, the label on the edge represents the dependency relationship between the words represented by the nodes. In this paper, we use the dependency relations found in the Stanford typed dependencies manual[1].

Definition 3: Let Z be text fragment consisting of sentences $\{T_i\}$ that contain the corresponding word-sets $\{W_i\}$; $\mathcal{DOI} = (\mathbb{R}, \mathbb{C}, \mathbb{Y}, h, r)$ a domain ontology. Structured information extraction on Z using $\mathcal{DOI}$ involves:

1) Determining a set T_{CTR} of candidate subject, predicate and object information constructs (constituting a triple) using an extraction algorithm.
2) Validating T_{CTR} to find a set $\{K = \{s_i, p_i, o_i\}\}$ of triples with respect to $\mathcal{DOI}$.
3) Representing the triples in K using a suitable representation mechanism.

III. Approach

We use a domain ontology to encode domain-specific concepts and relationships. We use *dependency graphs* in our approach to extraction of complex, nested and implicit relationships. We also make use of parse trees, i.e., ordered and rooted trees that represents the syntactic structure of a sentence using the Penn Treebank notation[2]. The main algorithm (ExtractInformation) parses the document into sentences and utilizes Stanford Parser [12] to get the

[1] http://nlp.stanford.edu/software/dependencies_manual.pdf
[2] ftp://ftp.cis.upenn.edu/pub/treebank/doc/manual/notation.tex

1082-3409/10 $26.00 © 2010 IEEE
DOI 10.1109/ICTAI.2010.98

dependency graph and parse tree representations of candidate sentences. These structures are then consumed by the extraction (Fig. 6) and representation (Fig. 7) algorithms, which apply a set of rules to the dependency graph and parse tree structures to produce a set of RDF triples. We use the RDF reification mechanism to capture assertions about assertions.

A. Composite Rule Framework

We use a set of information extraction rules of the following form:

Definition 4: (*Extraction Rule*) For a dependency graph G, is of the form $r_i : \{p_i\} \longrightarrow \{c_i\}$, which is interpreted as - if premises p_i of rule r_i hold, perform the consequences c_i of r_i.

1) Identifying and Defining the Relationship Types: Given an English language sentence, T, we define the complexity of the relationship(s) by the type and number of dependencies that exist within the words of T.

1) Simple Relationships
   Formally, we say that a sentence T contains a simple relationship if all of the following conditions hold for the dependency graph G of T:
   - T contains no more than one subject and object each. This further implies that it has at most one dependency of type $nsubj$ and one from the set $\{dobj, pobj\}$.
   - T does not contain any clause-level dependencies, conjunctions, or a clausal subject. Further, it can only contain noun-compound, or adjectival modifiers[3]
2) Complex Relationships
   For purposes of this paper, we limit the scope of complex relations to the following specific cases.

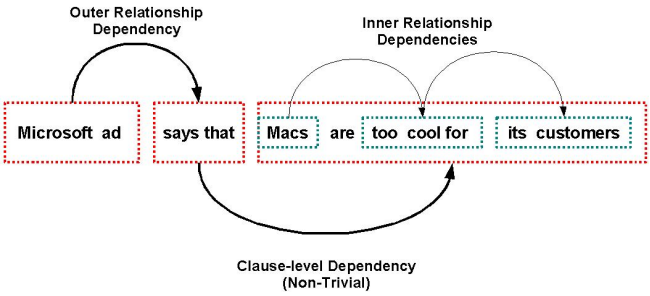

Figure 1. Example for a Relationship with Internal Clauses

a) **The relationship has internal clauses**
   This relationship type has a main subject that refers to an internal clause through a verb. The internal clause is often interpreted as the object of the dependent verb. Such relationships associate entities with facts, instead of two simple entities. This is illustrated in Fig. 1.

b) **Modifiers implicitly qualify the meaning of the relationship**
   In these relationships, there is a main clause whose meaning is qualified by a prepositional modifier. These relationships contain qualified information with respect to the source (the source can in turn be captured using the extraction rules for the clausal relationship), which may or may not be true on its own at the time of extraction.

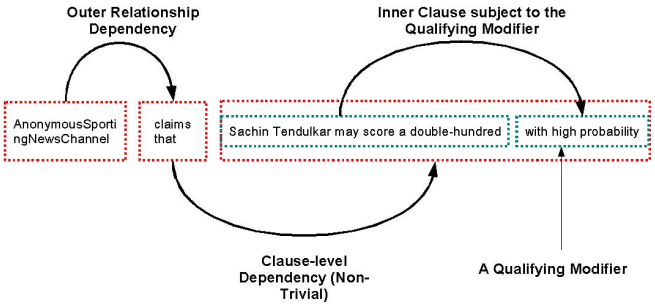

Figure 2. Example for a relationship with qualifying modifier

This idea is concisely illustrated in Fig. 2, which contains both cases 1 and 2.

c) **Multiple relations are formed by coordinating conjunctions**
   This type of relationship represents all the sentences with at least one conjunction *and, but, or, yet, for, nor, so*. For reducing the complexity of the representation step, we only handle the case with the and-conjunction.

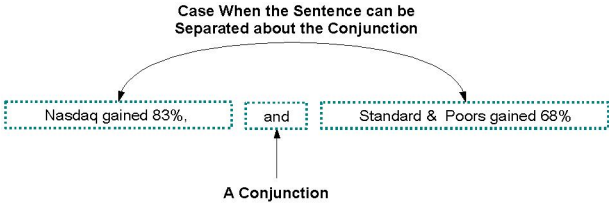

Figure 3. Example with a Simple Placement of Conjunction

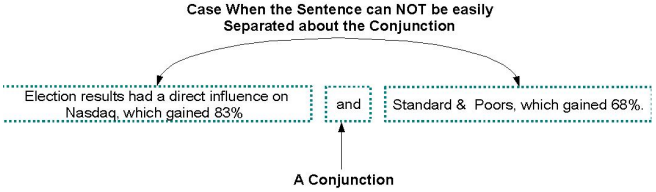

Figure 4. Example with a Complex Placement of Conjunction

[3]In terms of Stanford dependencies, this implies that T does not contain any dependencies from the set *ccomp, xcomp, acomp, compl, conj, csubj, csubjpass*. Also, T can only have *amod, quantmod, nn* as modifiers

The simplest case occurs when the conjunction separates two simple or complex clauses. In this case, the extraction task simply requires the separation of the sentence about the conjunction, followed by usual processes required for individual simple/complex clauses (refer Fig. 3). However, in many other cases, the second clause is dependent on the subject or predicate of the first clause. This is illustrated in Fig. 4. The latter case requires special handing with respect to the information extraction task since it contains one or more of the other relationship types described above.

2) Formulating Rules for Relationship and Entity Extraction: Since complex relationships are characterized by implicit and explicit dependencies between parts of the sentence, we formulate extraction rules that reflect the structure of the dependency graphs as follows:

1) From the set of Stanford dependency labels, choose the ones that fit the structure for each relationship type.
2) Form conditions and actions by deciding which nodes to extract as constructs when a dependency label (or a sequence of labels) is found along a set of edges in the dependency graph.
3) Express premises and consequents to form the extraction rule.

The application of these steps is illustrated below in the case of relationships with internal clauses (2a above).

1) **Rule formulation for relations with internal clauses:** For this case, we base our rule on two main clausal dependencies from the set of Stanford dependencies - *Clausal complement (ccomp) and Parataxis (parataxis)*. While formulating the consequent for the rule, we do not consider the constructs within the inner clause since it either comprises a simple relationship, which is dealt with a separate set of rules, or a complex relation, which can be recursively reduced to a simple relationship. This notion is utilized in algorithm EXTRACTINFORMATION. Formally, the general rule-set for this case is as follows.

   Extraction Rule 1: (*Relationships with Internal Clauses*) Given a dependency graph $G(V, E)$ with a label function l for an English-language sentence T, the extraction rule-set for relationship with internal clauses is given as,

   - $r_{RIC_1} : \{\exists u, v, w \in V, \exists e_1(u, v), e_2(v, w) \in E \,|\, l(e_1) = \text{“}nsubj'' \cap (l(e_2) = \text{“}ccomp'' \cup l(e_2) = \text{“}parataxis'')\} \longrightarrow \{pred_1 = \{v\}, sub_1 = \{u\}\}$

   - $r_{RIC_2} : \{\exists u, v, w, t \in V, \exists e_1(u, v), e_2(v, w), e_3(u, t) \in E \,|\, l(e_1) = \text{“}nsubj'' \cap (l(e_2) = \text{“}ccomp'' \cup l(e_2) = \text{“}parataxis'') \cap l(e_3) \in \{\text{“}nn'', \text{“}quantmod''\}\} \longrightarrow \{sub_1 = sub_1 \cup \{t\}\}$

   Result of application of this rule to the sentence in Fig. 1 is shown in Fig. 5.

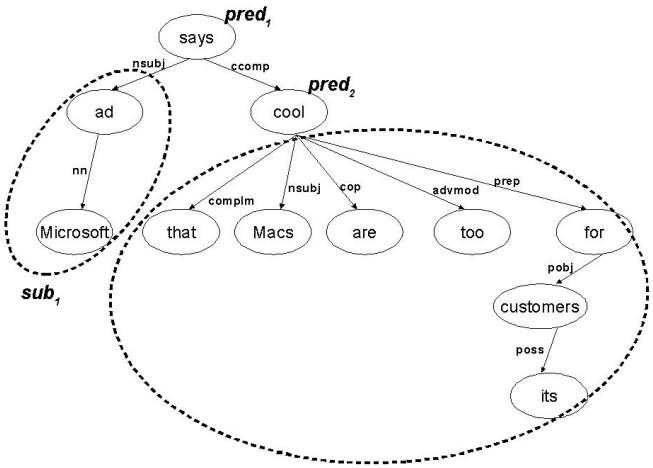

Figure 5. Extraction Rule Application for a Relationship with Internal Clauses

2) **Rule formulation for relations with qualifying modifiers:** For this case, we base our rule on two modifier dependencies - *Prepositional modifier (prep) and Adjectival modifier (amod)*. Prepositional and Adjectival modifiers capture the modifying qualifier and value of qualification respectively (details omitted due to space constraints).
3) **Rule formulation for relations with conjunctions** Conjunctions connect parts of a sentence, which do not have an implicit or explicit dependency on any but their immediate successor and predecessor in the sentence. Thus, we handle this as an auxiliary case in our extraction algorithm by analyzing the structure of the parse tree representation: if the parse tree for the right part contains a simple declarative clause (*S*), treat it as a distinct sentence and utilize the pre-existing rule-framework for extracting information; If the parse tree contains a verb phrase (*VP*) and noun phrase (*NP*), append the extracted subject of the left part to the right part and treat it as a distinct sentence: If the parse tree contains only a noun phrase (*NP*), append the extracted subject and the predicate of the left part to the right part and treat it as a distinct sentence.
4) **Rule formulation for simple relations:**. These rules reflect simple dependencies in the dependency graph G. We omit the details due to space constraints.

For sake of simplicity in validation and representation framework, we perform a negation check by looking for the presence of *neg* dependency, and if found, we skip the sentence from further processing. The rule framework described in this section is used by algorithm EXTRACTCONSTRUCTS

1: **procedure** EXTRACTCONSTRUCTS(p, G, R, l)
2: $rawC$ = CALL EXECUTERULES(G, R)
3: **if** $flag_{clausal}$ is true **then**
4: Split p about $pred_1$ into $p_l(V_1)$, $p_r(V_2)$
5: $G_S(V_2, E_2)$ = Subgraph of G induced by V_2
6: $innerC$ = EXTRACTCONSTRUCTS(p_r, G_S, R, l)
7: **for all** ele in $innerC$ **do**
8: Form an $outerC$ using $(sub_1, pred_1, ele)$
9: Add $outerC$ to the List of $outerCL$
10: **end for**
11: Cache first element from $outerCL$
12: **return** $outerCL$
13: **else if** $flag_{conj}$ is true **then**
14: Split p about $conj_{and}$ into $p_f(V_1)$, $p_s(V_2)$
15: $G_{S1}(V_1, E_2)$ = Subgraph of G induced by V_1
16: $G_{S2}(V_2, E_2)$ = Subgraph of G induced by V_2
17: $outerCL_1$ = EXTRACTCONSTRUCTS(p_f, G_{S1}, R, l)
18: Cache first element from $outerCL_1$
19: **if** p_s contains 'S' **then**
20: $outerCL_2$ = EXTRACTCONSTRUCTS(p_s, G_{S2}, R, l)
21: **else if** p_s contains 'VP' and 'NP' **then**
22: Append sub from the Cache to p_s
23: $outerC_2$ = EXTRACTCONSTRUCTS(p_s, G_{S2}, R, l)
24: **else if** p_s contains 'NP' **then**
25: Append sub and $pred$ from the Cache to p_s
26: $outerC_2$ = EXTRACTCONSTRUCTS(p_s, G_{S2}, R, l)
27: **end if**
28: Add $outerCL_1$, $outerCL_2$ and $outerC_2$ to $outerCL$
29: **return** $outerCL$
30: **else if** $flag_{enrich}$ is true **then**
31: Get $outerCL$ from Enrichment module
32: **return** $outerCL$
33: **else**
34: Create construct from $(sub_1, pred_1, obj_1)$
35: Add the construct to $outerCL$
36: **return** $outerCL$
37: **end if**

Figure 6. Extracting Candidate Information Constructs

when it calls EXECUTERULES.

B. Semantic Validation Framework

We now describe the validation of the extracted information constructs against the domain

1) Performing Validation against the Domain: Our basic approach for validating candidate information constructs *(subject, predicate, object)* against a domain description is as follows:

1) Find an instance match for the subject and the object. For determining these matches, we perform simple syntactic comparisons sequentially on the entire set of subject (similarly object) candidates starting with the extracted head sub_1 or obj_1.
2) If a match for subject and object is found, find a matching relationship for the *predicate* in the domain. For this, we perform syntactic comparisons on the relationships and the predicate.
3) If we are able to find these matches, we check if the class concepts to which the instances for subject and predicate are asserted lie respectively in domain and range of the relationship matched.

If all these conditions hold, we add the construct *(instance for subject, property, instance for object)* to the set of validated constructs (details omitted due to space constraints).

2) Dealing with incomplete domain models: We (optionally) create new definitions in the domain model to account for qualifying relationships. In this case, extraction algorithm EXTRACTCONSTRUCTS invokes the enrichment module which extends the domain model before the validation is performed on the constructs and returns a list of validated and enriched constructs. Recall that the rule for this relationship type extracted the following constructs $\{sub, pred, obj, \{qual_i, val_i\}\}$. We utilize the basic validation rule (outlined above) to validate $\{sub, pred, obj\}$. Further, we find a match for val within the set of instances. If a match for the val is found, we proceed to the next step of enrichment. If no match is found, we skip enrichment. To capture a qualifications, we create a new concept, *QualRel* and three new relationships *hasQualSub, hasQualPred, hasQualObj*, each with domain *QualRel* and range as the most general concept in the concept hierarchy. Each time the enrichment module creates a new instance (e.g, *QualRel_1*) of type *QualRel*, it creates a new property (*qualifier*) with domain *QualRel* and range as the concept of which the instance (that we found for *value*) is asserted. This process is repeated for each {*qual, val*} pair. Finally, it returns the resulting constructs in {*subject, predicate, object*} form.

C. Representation Framework

We now proceed to describe the representation of the extracted and validated constructs, $\{K = \{s_i, p_i, o_i\}\}$ into an RDF graph G_{RDF} using a set of transformations.

We represent simple relationships using the following transformation.

Primitive Transformation: For a set $\{K = \{s_i, p_i, o_i\}\}$ of 3-tuples s_i, p_i, o_i (validated information constructs), a primitive transformation is defined on the elements of $K_{simple} = \{\{s_i, p_i, o_i\} | \{s_i, p_i, o_i\} \in K \cap |o_i| = 1\}$ as follows:
$T_{prim}(\{s_i, p_i, o_i\}) \longrightarrow G_{RDF}$, which creates a graph G_{RDF} consisting of node-set $\{s_i, o_i\}$ and edge-set $\{p_i\}$ such that the only edge connects the two nodes. An example of this is shown in Fig. 8 using RDF triple notation.

Generating RDF representations of structurally complex relationships is more involved. We illustrate this process in

1: **procedure** RECURSIVEVALIDATEANDREPRESENT(c, l)
2: **if** obj from c is a list **then**
3: $valC$ = RECURSIVEVALIDATEANDREPRESENT(obj, l)
4: **if** $valC$ is not *null* **then**
5: Reify $valC$ as a RDF Statement st_r
6: Create RDF triple using sub, $pred$ and st_r
7: **end if**
8: **else**
9: Get $valC$ by Validating c.
10: **if** $valC$ is not *null* **then**
11: Create an RDF $triple$ using $valC$
12: **return** $triple$
13: **end if**
14: **end if**

Figure 7. Representing Validated/Enriched Information Constructs

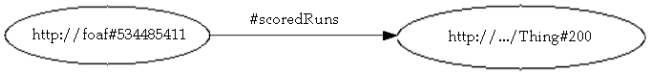

Figure 8. RDF representation of Simple Information

the case of relationships with internal clauses.

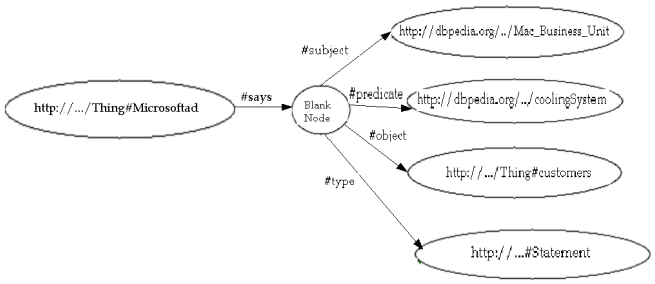

Figure 9. RDF representation of Reified Internal Clauses

Here, we need to represent a subject associated with another fact (extracted from the inner clause), which, in turn, may be associated with some other fact and so on. For this, we use a composite transformation as follows.

Composite Transformation: For a set $\{K = \{s_i, p_i, o_i\}\}$ of 3-tuples s_i, p_i, o_i (validated information constructs), a composite transformation is defined on the elements of $K_{complex} = \{\{s_i, p_i, o_i\} | \{s_i, p_i, o_i\} \in K \cap |o_i| > 1\}$ as follows:
$T_{comp}(\{s_i, p_i, o_i\}) \longrightarrow T_{prim}(\{s_i, p_i, t\})$
$\cap T_{prim}(\{t, obj, o_{o_i}\}) \cap T_{prim}(\{t, pred, p_{o_i}\})$
$\cap T_{prim}(\{t, sub, s_{o_i}\}) \cap T_{prim}(\{t, stmt, id\})$, where t is a special node, called *blank node*, $stmt$ is a special edge, called *Statement*, sub, obj and $pred$ are special edges, p_{o_i}, s_{o_i} and o_{o_i} are internal elements of o_i. This composite transformation triggers five primitive transformations creating subgraphs for each. Fig. 9 shows the RDF representation

1: **procedure** EXTRACTINFORMATION(D, l)
2: Tokenize D about delimiters to get a Set, $Sent$
3: **for all** T in $Sent$ **do**
4: (p, G) = Get Parse Tree, Dependency Graph from T
5: $outerCL$ = EXTRACTCONSTRUCTS(p, G, R, l)
6: **for all** c in $outerC$ **do**
7: RECURSIVEVALIDATEANDREPRESENT(c, l)
8: **end for**
9: **end for**

Figure 10. Information Extraction from a Text Source

of the information extracted from the sentence in Fig. 5. We model this using *reified* statements in RDF, where we reify the internal fact as a statement in RDF and then create a triple using the subject, predicate and this statement as the object. The same overall approach is used for transforming the other extracted relationships into RDF (details omitted due to space constraints).

Finally, EXTRACTINFORMATION (Fig. 10) controls the overall information extraction task by invoking the extraction, validation and representation procedures.

IV. IMPLEMENTATION AND EVALUATION

The above described framework and algorithms are implemented as Semantixs [1], a Java-based open-source web application. We used the implementation to perform some preliminary evaluation of our information extraction framework.

A. Text and Domain Ontology

Traditional benchmarks for information extraction systems e.g., the MUC benchmarks [7], are not suited for complex nested relationship extraction tasks. Hence, we used online news articles in our experiments. We queried *CBS News.com* with the keyword "Dow Jones", which resulted in about 5228 articles, videos and other related material. We randomly selected 4 text articles from the results of the query for use in our experiments. Because the documents contained relatively simple relationships, we augmented the text with a few complex relationship types. Table I shows the counts of relationships that we manually identified in the text[4].

Table I
COUNTS OF POSITIVE AND NEGATIVE INSTANCES IN TEXT

Text	Simple	Clausal	Qualified	Conjunction	Reference
1	7/6	7/1	1/2	2/1	4/2
2	13/0	6/0	3/0	3/0	2/2
3	23/5	12/2	2/0	2/1	6/2
4	18/14	5/0	1/0	2/0	1/2
Tot	61/25	30/3	7/2	9/2	13/8

[4]The data set is available from the authors upon request

We utilized the latest DBpedia ontology[5] and a subset of instances[6].

Table II
COUNTS OF CORRECT POSITIVE AND NEGATIVE INSTANCES EXTRACTED

Text	Simple	Clausal	Qualified	Conjunction	Reference
1	7/3	5/1	0/2	2/1	3/0
2	13/0	5/0	2/0	3/0	2/0
3	22/3	10/1	1/0	2/1	5/1
4	16/10	4/0	1/0	2/0	1/1
Tot	58/16	24/2	4/2	9/2	11/2

B. Results

The results of information extraction are presented in Table II. This table shows the counts for correctly classified information (extracted, validated and represented correctly, or rejected). In coming up with Table II, we used the following (necessarily imperfect) correctness criteria: For clausal and qualified relations, the correctness was judged based on whether (i) correct structural representation was extracted for complex relationship (ii) correct semantic representation was extracted for the simple relationship(s) included in a complex relationship. Correct and complete extraction of all the relationships present in a sentence contributed to each individual count for the relationship types. Partially-correct extraction of relations present in a sentence still contributed to the corresponding counts for those relationship(s) that were correctly extracted.

Table III
RESULTS: PRECISION, RECALL AND F-MEASURE

	Simple	Clausal	Qualified	Conjunction	Refs
Precision	0.86	0.96	1.0	1.0	0.65
Recall	0.95	0.80	0.57	1.0	0.85
F-measure	0.90	0.87	0.73	1.0	0.74

The precision, recall and f-measure for individual relationship types are reported in Table III. In the case simple relations, errors can be traced to shallow syntactic comparisons used to determine matches between the candidate information constructs and the domain concepts, relationships or instances. False negatives in extracted clausal relations can be traced to multi-level dependency structures and cross-sentential references. The false negatives in qualified relations could be traced to shallow syntactic comparisons to determine a match. Other sources of error could be traced to poor co-reference resolution, gaps in the methods used for pronoun resolution, etc.

[5]http://wiki.dbpedia.org/Downloads34#dbpediaontology
[6]http://wiki.dbpedia.org/Downloads34#ontologytypes

V. CONCLUSION

We have designed and implemented an ontology-based system for extraction of a subset of complex nested relationships from text. A novel feature of the system is its modular design, which separates the domain ontology from the extraction rules, making it easy to port to new domains simply by selecting an appropriate domain ontology. The system outputs the extracted relationships in the form of RDF triples, suitable for querying and data mining.

REFERENCES

[1] S.Pandit, "Semantixs: A system for extraction of domain-specific information from text including complex structures," http://sourceforge.net/projects/semantixs, 2010.

[2] C. Ramakrishnan, K. J. Kochut and A.P. Sheth, "A Framework for Schema-Driven Relationship Discovery from Unstructured Text," International Semantic Web Conference: 583-596, 2006.

[3] J. Saric, L. J. Jensen, R. Ouzounova, I. Rojas and P. Bork, "Extraction of regulatory gene/protein networks from Medline," Bioinformatics, Vol. 22 no. 6: 645650, 2006.

[4] E. Riloff, "Automatically constructing a dictionary for information extraction tasks," Proceedings of the 11th National Conference on Artificial Intelligence, AAAI-93: 811816, 1993.

[5] S. Huffman, "Learning information extraction patterns from examples," Workshop on new approaches to learning for natural language processing, IJCAI-95: 127142, 1995.

[6] J. Kim and D. Moldovan, "Acquisition of linguistic patterns for knowledge-based information extraction," IEEE Transactions on Knowledge and Data Engineering, 7(5): 713724, 1995.

[7] R. Grishman, B. Sundheim, "Message Understanding Conference - 6: A Brief History," Proceedings of the 16th International Conference on Computational Linguistics (COLING), I, Kopenhagen: 466471, 1996.

[8] S. Soderland, et al, "Crystal: Inducing a conceptual dictionary," Proceedings of the 14th International Joint Conference on Artificial Intelligence, IJCAI-95: 13141319, 1995.

[9] K. Fundel, R. Kuffner and R. Zimmer, "RelExRelation extraction using dependency parse trees," Bioinformatics, Vol. 23 no. 3: 365371, 2007.

[10] C. Friedman, et al, "GENIES: a natural-language processing system for the extraction of molecular pathways from journal articles," Bioinformatics, Vol. 17 Suppl. 1: 1367-4803, 2001.

[11] "Critical Assessment of Information Extraction Systems in Biology," http://www.mitre.org/public/biocreative.

[12] "The Stanford Parser: A statistical parser," http://nlp.stanford.edu/software/lex-parser.shtml.

[13] N. Bach and S. Badaskar, "A survey on relation extraction," Language Technologies Institute, Carnegie Mellon University, 2007.