Encoding Extraction as Inferences

J. William Murdock[1], Paulo Pinheiro da Silva[2], David Ferrucci[1], Christopher Welty[1], Deborah McGuinness[2]

[1]IBM Watson Research Center
19 Skyline Drive
Hawthorne, NY 10532, USA
{murdockj,ferrucci,welty}@us.ibm.com
[2]Knowledge Systems Laboratory
Stanford University
Stanford, CA 94305, USA
{pp,dlm}@ksl.stanford.edu

Abstract

The analysis of natural-language text involves many different kinds of processes that might be described in multiple ways. One way to describe these processes is in terms of the semantics of their requirements and results. Such a description makes it possible to view these processes as analogous to inference rules in a theorem-proving system. This analogy is useful for metacognition because there is existing theory and infrastructure for manipulating inference rules. This paper presents a representational framework for text analysis processes. We describe a taxonomy of text extraction tasks that we have represented as inference rules. We also describe a working system that encodes the behavior of text analysis components as a graph of inferences. This representation is used to present browsable explanations of text extraction; in future work, we expect to perform additional automated reasoning over this encoding of text analysis processes.

Introduction

Many domains require that conclusions be drawn from an assortment of unstructured sources (e.g., text, audio, video). For example, intelligence analysis involves challenges such as detecting emerging threats that may be addressed by sources such as news reports. Addressing these problems typically involves some combination of extracting structured knowledge from these sources and then performing additional reasoning to infer consequences of that knowledge. There are a variety of metacognition goals that are applicable in this context, e.g., explaining how results were obtained, vetting results from untrusted or unreliable processes, and selecting follow-on tasks to perform. To be complete, such explanations and analyses should be based on integrated representations of extraction and of additional reasoning. However, while representations of processes for reasoning over structured knowledge are fairly common (e.g., Davis, Buchanan, & Shortliffe, 1977; Swartout, Paris, & Moore, 1991; McGuinness, 1996), there is much less work on metacognition over processes that extract structured knowledge from unstructured information.

One requirement for representing analysis of unstructured information is a *taxonomy* of analysis tasks that is complete enough to accurately describe task functionalities and abstract enough to hide unnecessary technical details. Given such a taxonomy, it is possible to observe the tasks that analysis systems are performing and record those observations (Ferrucci, 2004).

We have built such a taxonomy for the analysis of natural-language text. We have also implemented a system that records processes (using the terms of this taxonomy) in a manner that is consistent and compatible with explanations for processes that use extracted knowledge to perform additional reasoning. Our system uses these records to provide browsable explanations, and we intend to also explore other metacognition goals such as vetting results and selecting new tasks to perform.

Two existing projects serve as a starting point for our work on metacognition over knowledge extraction. UIMA, the Unstructured Information Management Architecture, is a framework for integrating software components that analyze unstructured information such as text (Ferrucci & Lally, 2004). The Inference Web is a framework for explaining answers from question answering systems that manipulate structured information, and now unstructured information (McGuinness & Pinheiro da Silva, 2004). These established frameworks provide a basis for storing and using records of knowledge-extraction processes.

Background

UIMA provides an infrastructure for integrating analysis components. An aspect of UIMA that is particularly important for enabling metacognition is the fact that analysis components are specified using a declarative formalism with the following characteristics:

- The formalism is hierarchical, i.e., aggregate components may be constructed out of a combination primitive components and/or other aggregate components.

- The formalism describes input requirements and output capabilities of a component using a formal ontology.[1]

Inference Web provides an infrastructure for providing explanations from distributed hybrid question answering systems. It includes the following components:

- A *language* supporting interoperability of information manipulation systems. PML – the Proof Markup Language (Pinheiro da Silva, McGuinness & Fikes, 2004) is used to encode information manipulations. PML is specified in OWL[2], and thus is compatible with semantic web applications in XML, RDF, and OWL. It includes content such as sources, inference rules, inference steps, and conclusions.
- A *registrar* for managing and storing proof-related meta-information encoded in PML.
- A *browser* for viewing information encoded in PML thus supporting an interactive interface.
- A *PML checker service* for identifying incorrect applications of rules in PML documents.
- An *explainer* for presenting users with explanations for answers and enabling them to ask follow-up questions about answers and their explanations.
- An *abstractor* for rewriting PML documents from machine-level inferences into human-level inferences thus supporting explanations at an appropriate (user-selectable) level of abstraction.

If the existing UIMA architecture encoded its intermediate and final analysis products as declarative assertions, then it would be relatively easy to encode its processes as Inference Web rules. However, UIMA's internal representations, which are motivated by common practice among developers of components that analyze unstructured information, typically do not produce intermediate results in the form of declarative assertions (even when their final products are declarative assertions). Building a complete trace of a UIMA extraction process thus requires additional insight into those processes. Consequently, we have built a taxonomy of reasoning tasks that occur during knowledge extraction. For each element in the taxonomy, we have identified the structures that are populated by this task, and we have created formal assertions that describe the semantics of those structures. Given this taxonomy, it is possible to take the results of UIMA-based text analysis and infer what tasks were performed, what information was used, and what conclusions were drawn from each of these tasks.

Illustrative Example

Consider the following English sentence: "Joseph Gradgrind is the owner of Gradgrind Foods." A simple knowledge extraction system might produce the following conclusions given that sentence:

- The string "Joseph Gradgrind" refers to an entity instance of type `Person`.
- The string "Gradgrind Foods" refers to an entity instance of type `Organization`.
- The entire sentence refers to an `OwnerOf` relation instance.
- The "Joseph Gradgrind" (`Person`) reference is the subject of that `OwnerOf` relation instance.
- The "Gradgrind Foods" (`Organization`) reference is the object of that `OwnerOf` relation instance.

Figure 1 illustrates these results graphically. The relatively compact view presented in that figure is ideal for users (or components) that are exclusively interested in the final results and not in how those results were obtained.

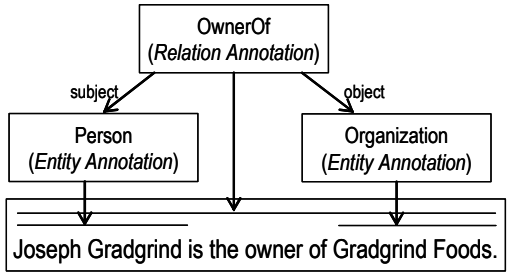

Figure 1: Sample extraction results**.**

Results of this sort might be produced by a UIMA aggregate analysis engine. Figure 2 shows a representation of an extraction process for a set of analysis components producing these results. At the top of the figure are four references to two sources (one reference to Document 1, and three references to the extraction ontology). Below

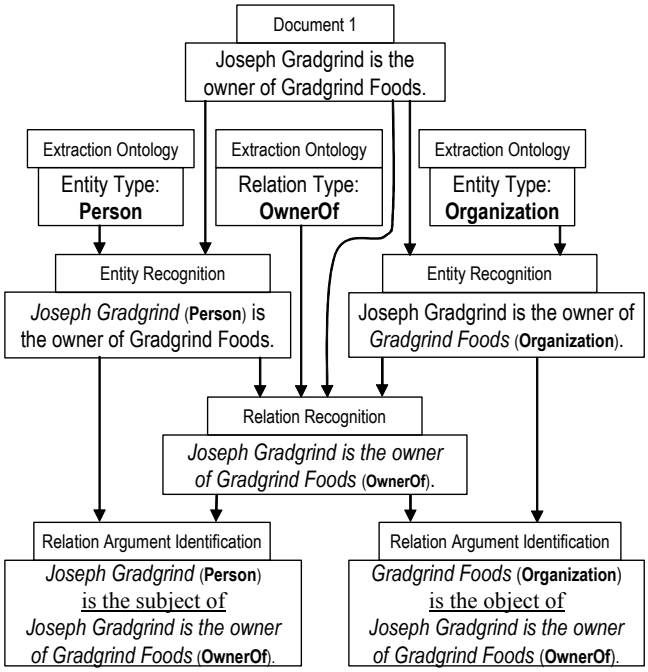

Figure 2: A sample reasoning trace

[1] Example ontologies of this sort are included in the UIMA SDK: http://www.ibm.com/alphaworks/tech/uima

[2] http://www.w3.org/TR/owl-features/

that are the five conclusions drawn during this process. These conclusions are drawn by three different extraction tasks (*Entity Recognition*, *Relation Recognition*, and *Relation Argument Identification*). These are three of the nine tasks in the taxonomy presented in the next section.

The perspective illustrated in Figure 2 is particularly appropriate for metacognition for a variety of reasons:

- Each conclusion is represented by a distinct node with a link to the task that drew the conclusion. This facilitates reasoning (either by a human or a software system) about how the individual tasks contributed to the end result.
- Original sources (i.e., the text and the elements of the ontology) are also represented by nodes. This facilitates reasoning about the underlying basis of the extraction.
- The arrows between the nodes show the information flow from original sources to task conclusions. This facilitates reasoning about the dynamics of the extraction process including knowledge provenance (Pinheiro da Silva, McGuinness, & McCool, 2003).

Taxonomy of Primitive Tasks

We divide the process of extracting knowledge from text into three areas: *annotation*, *coreference resolution*, and *knowledge integration*. We have identified three distinct primitive tasks in each of these three areas.

The annotation tasks involve making assertions about individual spans of text (a span is defined by a beginning character position and an ending character position within a given document, e.g., characters 0-8 of *foo.txt*). The example in the previous section includes only annotation tasks. Below is our list of annotation tasks:

1) **Entity Recognition**: Requires text and an entity type in the *extraction ontology* (i.e., the set of terms that define the extraction capabilities). Concludes that a *span* (i.e., a segment of the original text) refers to an unspecified entity instance of the specified entity type. In other words, this task produces a new data item (called an *entity annotation*) that is associated with an entity type and contains a reference to a span of text.
   Example: Inferring that the span 0-16 (i.e., "Joseph Gradgrind") of Document 1 refers to some (unspecified) instance of the `Person` entity type.
2) **Relation Recognition**: Requires text and a relation type in the extraction ontology. Concludes that a span refers to some unspecified relation instance of the specified relation type; i.e., produces a *relation annotation*.
   Example: Determining that the span 0-50 (i.e., "Joseph Gradgrind is the owner of Gradgrind Foods.") of Document 1 refers to some (unspecified) instance of the `OwnerOf` relation type.
3) **Relation Annotation Argument Identification**: Requires a relation annotation and another annotation (either an entity annotation or a relation annotation). Concludes that the latter fills a *role* (e.g., `subject`, `object`) in the former.
   Example: Determining that the `subject` role in an `OwnerOf` relation annotation is filled by a specified `Person` entity annotation.

The coreference resolution tasks build on the results of annotation. These tasks determine what specific instances the annotations refer to; for example, if one sentence states that "Joseph Gradgrind is the owner of Gradgrind Foods" and another states that "Joe Gradgrind lives in New Jersey," a coreference process might determine that there is a single person instance that is referred to separately as "Joseph Gradgrind" and "Joe Gradgrind."

The end result of coreference resolution is an *extracted knowledge base* (i.e., a set of entity and relation instances encoded in the extraction ontology) and links from the instances in the knowledge base to the annotated text from which those instances were derived. Coreference resolution tasks are listed below:

4) **Entity Identification**: Requires a set of (one or more) entity annotations. Concludes that these annotations refer to a particular *entity instance* (assigning them a common, new identifier).
   Example: Determining that the entity annotations on "Joseph Gradgrind" and "Joe Gradgrind" refer to entity instance `UID1` and adding `UID1` to the extracted KB.
5) **Relation Identification**: Requires a set of (one or more) relation annotations. Concludes that these annotations refer to a particular *relation instance* of a specified relation type with specified values for its roles.
   Example: Determining that the relation annotation on "Joseph Gradgrind is the owner of Gradgrind Foods" refers to a relation instance `(OwnerOf UID1 UID2)` and adding that instance to the extracted KB.
6) **Extracted Entity Classification**: Requires an entity instance and a set of entity annotations that refer to it. Concludes that it has a particular entity type in the extraction ontology. Typically, the types assigned to the entity annotations (by Entity Recognition) are used to select a type for the entity; for example, a voting scheme may be used to select the most common type among the annotations for the entity.
   Example: Determining that the entity type of instance `UID1` is `Person`.

Knowledge integration takes an extracted knowledge base (encoded using the extraction ontology) and produces a target knowledge base (encoded using the target ontology, for which the knowledge extraction system is intended to produce results). In the following examples we introduce namespace prefixes (Ex: and Ta:) to clearly distinguish the IDs of entities and relations in the two ontologies.

Differences between the two ontologies can be an incidental consequence of using legacy extraction components to address new requirements. Alternatively,

differences may be deliberate, in order to optimize the ontologies for their distinct applications. In either case, tasks such as the following are required to map between these ontologies:

7) **Entity Mapping**: Requires a set of entity or relation instances in the extraction KB. Concludes that an entity instance in the target KB is derived from these instances. Note that it is possible for relations to map to entities or vice versa, depending on representation choices made in the two ontologies.
   Example: Determining that the `Ta:X1` entity instance is derived from the `Ex:UID1` entity instance (and adding the former to the target KB).
8) **Relation Mapping**: Requires a set of entity or relation instances in the extraction KB. Concludes that a relation instance in the target KB is derived from these instances.
   Example: Determining that the `(Ta:hasOwner Ta:X2 Ta:X1)` relation instance is derived from the `(Ex:OwnerOf Ex:UID1 Ex:UID2)` relation instance (and adding the former to the target KB).
9) **Target Entity Classification**: Requires an entity instance in the target KB and a set of extracted entity and/or relation instances that this entity instance is derived from. Concludes that the entity instance is a member of an entity type in the target ontology (based on the types assigned to the instances from which it was derived).
   Example: Determining that the entity type of entity instance `Ta:X1` is `Ta:PERSON`.

Representation

Metacognition is possible if a record of the tasks performed and their results can be stored, understood, and reused. We have built a system that enables metacognition using PML as an inter-lingua representation for justifications of knowledge extracted by UIMA-compliant components. PML documents may be used for representing, combining, exchanging, abstracting, and presenting information. Thus our PML-based representation enables the exchange of information between extraction components and tools in a declarative and reusable manner.

Each extraction task has a conclusion that follows from its inputs. In Figure 2, for example, one of the steps performed involves the Entity Recognition task determining that "Joseph Gradgrind" refers to a person. From this example we generate a *node set* in PML with the following content:

- An *inference step* that links to a representation of the task performed (Entity Recognition) encoded as a PML *inference rule*.
- A *conclusion* that describes the results of that task (that "Joseph Gradgrind" refers to a person).
- A set of *antecedents*, i.e., other node sets whose conclusions provide inputs to this task. Antecedents of an Entity Recognition rule would include a node set for the original text and a node set for an entity type from the ontology.

PML allows some conclusions to exist with no antecedents. These node sets are considered *direct assertions* and they use *sources* rather than inference steps for justification. In our text extraction processes, we deal with two different kinds of sources: text and ontologies. The node sets for those sources act as antecedents to the node sets that involve extraction inference steps.

PML does not require a conclusions to be written using one language. Instead, each node set is required to include a reference to meta-information about the language used to write the conclusion. In records of extraction processes, some node sets are direct assertions of raw text; each of these node sets has that raw text as its conclusion and the natural language in which the text is written as its language. Other node sets represent extracted knowledge. Each of these node sets has a conclusion that is represented in some logical formalism and includes a link to a description of that formalism.

Implementation

We have implemented our approach integrating UIMA with Inference Web. The UIMA platform has been extended to produce PML traces for the nine primitive tasks in our taxonomy. Inference Web allows a user to ask for source provenance and extraction conclusion explanations. The conclusions in the node sets produced by this extension fall into the three categories: natural language direct assertions, traditional knowledge representation end results, and intermediate statements about the text. The IWBrowser already included support for displaying conclusions encoded in a traditional KR language (KIF). We have extended IWBrowser to also provide explicit support for displaying node sets whose conclusions are represented internally as natural-language text or as logical statements about text. Text is displayed without any special formatting, while statements about text are displayed as markup on that text. For example, the statement indicating that characters 0 to 16 of some document refer to a member of the `Person` class is displayed by highlighting the span from characters 0 to 16 and placing the type, `Person`, next to that highlighting, e.g., "*Joseph Gradgrind* **`(Person)`** is the owner of Gradgrind Foods." This format is illustrated in Figure 2 and is used in the IW Browser to present information.

Our resulting system provides an end to end prototype that has been tested in the domain of intelligence documents. The prototype allows users to run a set of UIMA-compliant analysis components on a corpus of documents and browse the results as an interrelated set of entity and relation instances. When looking at specific instances, the users can ask to see the trace for the

extraction process that generated those instances; this trace is presented in the Inference Web browser.

Future Work

We intend to explore a variety of additional uses of our taxonomy of extraction tasks and to provide a variety of extensions to that taxonomy. One of the tradeoffs that we will explore is the level of granularity that is most useful in extraction inference specification. While we do not want to overwhelm end users with detail, we do want to provide enough information so that they can understand the reasoning process and quickly identify any assumptions, uncertainties, or unauthoritative sources that would raise concern about conclusions. We also intend to perform automated reasoning over our encoding of the text analysis processes. This will enable us to, for example, check some of our extractions for compatibility with other known facts.

Additional Metacognition Capabilities

PML documents based on extraction tasks can potentially be used for many different metacognition capabilities including the following: generation of explanations describing how knowledge is extracted from natural-language text, computing trust values for extracted knowledge, and automatically selecting extraction tasks to perform given some requirements.

Users of UIMA-based extraction systems can interact with the system through the IWExplainer dialogue interface. They may wish to understand a particular statement in the knowledge base and how it was determined. The IWExplainer can present a summary of the sources used, the meta-information associated with those sources including any trust ratings, a summary of assumptions used, and an abstraction of any extraction inferences used. IWExplainer can also invoke the IWBrowser to let the user interactively explore and visualize the information manipulation steps involved. For the example in Figure 2, it could be useful for users to know that only two sources were used (Document 1 and the extraction ontology). Users may be interested in information about those sources such as the authors, currency, and authoritativeness. Additionally we can abstract some of the extraction inferences to provide a simpler visualization of the information manipulation process. IWBrowser extensions are under development for abstracting extraction traces at browsing time.

Considering the diversity of UIMA-compliant components, users may ask if intermediate and final results are internally consistent. For example, if the conclusion of a node set for Entity Recognition is the statement that characters 0 to 16 of a document refer to a `Person`, then a metacognition component could check that 0 to 16 are valid characters in the text and that `Person` is a valid type in the ontology. The taxonomy of primitive tasks plays a key role towards checking PML records of extraction processes. In fact, IW tools such as the PML Checker Service can retrieve formal specifications of inference rules and match them against a node set's conclusions and antecedents. Also, IW includes an inference meta-language (called InferenceML; Pinheiro da Silva, *et al.*, 2004) that is used to specify inference rules. The PML Checker currently is being extended to support verification of assertions about natural language text, thus enabling the checking of extraction processes.

In addition, users may not be familiar with one or more sources used in the knowledge extraction process. Users, however, may trust other users who may are familiar with the sources. Using the IW infrastructure, users may ask for trust values for extracted knowledge. Trust values for extracted knowledge are computed from trust values for sources, which are computed from trust relations among users and between users and sources. The computation of trust values is possible if inferences are stored in PML, node sets storing told information are associated with IWBase provenance elements, and provenance elements are part of a network of trust (Zaihrayeu, Pinheiro da Silva, & McGuinness, 2004).

Finally, a system could perform metacognition to select which extraction components to use given some specifications of user requirements. Having explicit representations of the inference engines available to the system and the tasks that those engines perform can provide the knowledge needed to automatically select components that are applicable to a given problem. IWRegistrar provides a mechanism for formally representing, storing, and accessing components (*inference engines*) and the tasks they perform (*inference rules*); these mechanisms may be applicable to automated selection and composition of knowledge extraction software components.

Extensions to the Taxonomy

The taxonomy described earlier in this document describes a number of primitive tasks that we have identified. However, this is not a complete taxonomy of every sort of task performed by knowledge extraction components. For example, one common task in extraction involves identifying *canonical* and *variant* names for instances. We intend to extend our taxonomy to handle a broader range of extraction tasks.

Furthermore, the taxonomy of tasks presented in this document is limited to extraction from *text*. However, UIMA is designed to support analysis of a wide variety of unstructured modalities such as images, audio, and video. Valuable potential extensions include (1) representing primitive tasks that are specific to different modalities; (2) identifying tasks that are performed across different modalities, possibly by creating more general formalizations of tasks in our existing taxonomy; and (3) building a working system that combines multi-modal extraction with an Inference Web registering component. We have begun, for example, to consider the task of explaining extractions from geospatial graphical information.

Finally, the organization of the primitive tasks presented here into different sections (annotation, coreference resolution, knowledge integration) suggests that these primitive tasks are components of larger, more abstract tasks. In future work, we will try to formally characterize these abstract tasks and thus potentially provide more abstract presentations. The Inference Web explanation facilities (McGuinness & Pinheiro da Silva, 2004) provide support for abstract presentations of PML proofs.

Discussion

One approach to explaining reasoning is to list the inference rules that were invoked along with the specific values on which those invocations occurred (Davis, Buchanan, & Shortliffe, 1977). However, such lists are often too long and complex to act as useful explanations; one way of addressing this limitation is to present users with relatively abstract explanations initially and allow them to "drill down" into the details of specific inferences only when needed (Swartout, Paris, & Moore, 1991; McGuinness, 1996; McGuinness & Pinheiro da Silva, 2004). This drill-down process typically grounds out at the level of atomic, directly asserted facts. In past work, such facts may have included a link to an original source (Pinheiro da Silva, McGuinness, & McCool, 2003), but did not provide the capability to drill down further to get an explanation of how it was determined that the fact appears in the source. Instead, it was assumed that the directly asserted facts were always in some structured form that required no analysis or that the analysis process that extracted these facts was atomic and not amenable to explanation. The taxonomy and representation described in this paper make it possible to overcome this limitation of past work by enabling detailed explanation of text analysis processes. This combination of explaining text analytics in combination with hybrid deductive reasoning is a unique contribution of this work.

There is a significant amount of existing work on building causal and/or explanatory representations of the *results* of text analysis (e.g., Ram, 1994; Mahesh, *et al.*, 1994; Moldovan & Rus, 2001). Representing analysis *processes* is less common. One system that does engage in metacognition over text analysis processes is Meta-AQUA (Cox & Ram, 1999), which generates explanations of reasoning failures in the domain of story understanding to facilitate automated learning. However, Meta-AQUA's representations and learning goals are largely oriented toward answering questions that are only implicitly addressed in the text being analyzed. Consequently, the tasks of interest are ones such as retrieving scripts and predicting outcomes that are relevant to extracting implicit information from text. These tasks are complementary to the tasks described in this paper, which involve extracting information that is explicitly stated in text.

We have presented a design and prototype implementation for a system that integrates text analysis, reasoning, and explanation to support metacognition. This paper introduces the key to the integration: a taxonomy of inference rules capturing the information manipulation tasks performed in text analysis.

Acknowledgements: This work is partially funded by the Advanced Research and Development Activity under the Novel Intelligence from Massive Data (NIMD) program. Contract #:2003*H278000*002-2.

References

Cox, M.T. & Ram, A. 1999. Introspective Multistrategy Learning: On the Construction of Learning Strategies. *Artificial Intelligence*, 112:1-55.

Davis, R, Buchanan, B., & Shortliffe, E. 1977. Production Rules as a Representation for a Knowledge-Based Consultation Program. *Artificial Intelligence*, 8:15-45.

Ferrucci, D. 2004. Text Analysis as Formal Inference for the Purposes of Uniform Tracing and Explanation Generation. IBM Research Report RC23372.

Ferrucci, D. & Lally, A. 2004. UIMA by Example. *IBM Systems Journal* 43(3):455-475 (2004).

McGuinness, D.L. Explaining Reasoning in Description Logics. Ph.D. Thesis, Rutgers University, 1996. Technical Report LCSR-TR-277.

McGuinness, D.L. & Pinheiro da Silva, P. 2004. Explaining Answers from the Semantic Web: The Inference Web Approach. *Journal of Web Semantics* 1(4):397-413.

Mahesh, K., Peterson, J., Goel, A., & Eiselt, K. 1994. KA: Integrating Natural Language Understanding with Design Problem Solving. In *Working Notes from the AAAI Spring Symposium on Active NLP*.

Moldovan, D. & Rus, V. 2001. Explaining Answers with Extended WordNet. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, Toulouse, France.

Pinheiro da Silva, P., Hayes, P. J., McGuinness, D. L., & Fikes, R. 2004. PPDR: A Proof Protocol for Deductive Reasoning. Technical Report, Knowledge Systems Laboratory, Stanford University.

Pinheiro da Silva, P., McGuinness, D. L., & Fikes, R. 2004. A Proof Markup Language for Semantic Web Services. *Information Systems Journal* (to appear).

Pinheiro da Silva, P. McGuinness, D. L., & McCool, R. 2003. Knowledge Provenance Infrastructure. *IEEE Data Engineering Bulletin*, 26(4):26-32.

Ram, A. 1994. AQUA: Questions That Drive the Explanation Process. In *Inside Case-Based Explanation*, Schank, Kass, & Riesbeck (eds.), pp 207-261.

Swartout, W.R., Paris, C., & Moore, J. D. 1991. Explanations in Knowledge Systems: Design for Explainable Expert Systems. *IEEE Expert Systems*, 6(3):58-64.

Zaihrayeu, I., Pinheiro da Silva, P., & McGuinness, D.L., 2004. IWTrust: Improving User Trust in Answers from the Web. Technical Report DIT-04-086, Informatica e Telecomunicazione, University of Trento, Italy.