

XAR: An Integrated Framework for Semantic Extraction and Annotation

Naveen Ashish and Sharad Mehrotra
University of California-Irvine
ashish@ics.uci.edu

Keywords: Data Extraction, Data Semantics, Text Processing Software, Logic Programming, Natural Language Processors, Relational Model.

INTRODUCTION

The vision of semantic interoperability on a large-scale, such as that envisioned by the concept of the Semantic-Web (Berners-Lee, Hendler & Lassila, 2001), continues to sustain interest and excitement. The availability of automated tools for *semantic annotation* of data on the open Web is recognized as critical for Semantic-Web enablement. In the process of semantic annotation we annotate significant entities and relationships in documents and pages on the Web, thus making them amenable for machine processing. The time and investment of marking and annotating Web content manually is prohibitive for all but a handful of Web content providers, which leads us to develop automated tools for this task. As an example, consider Web pages of academic researchers with their biographies in free text as shown in Fig 1.

Professor Deborah Estrin is a Professor of Computer Science with a joint appointment in Electrical Engineering at UCLA, holds the Jon Postel Chair in Computer Networks, and is Founding Director of the NSF-funded Center for Embedded Networked Sensing (CENS). Estrin received her Ph.D. in 1985 in Computer Science from the Massachusetts Institute of Technology, her M.S. in 1982 from M.I.T. and her B.S. in 1980 from U.C. Berkeley. Before joining UCLA she was a member of the University of Southern California Computer Science Department from 1986 through the middle of 2000. In 1987, Professor Estrin received the National Science Foundation, Presidential Young Investigator Award for her research in network interconnection and security. During the subsequent



Professor <name> Deborah Estrin</name> is a <title> Professor</title> of Computer Science with a joint appointment in Electrical Engineering at UCLA, holds the Jon Postel Chair in Computer Networks, and is Founding Director of the NSF-funded Center for Embedded Networked Sensing (CENS). Estrin received her <degree> Ph.D.</degree> in <PhDDate> 1985</PhDDate> in Computer Science from the <PhDSchool> Massachusetts Institute of Technology</PhDSchool>, her <degree> M.S.</degree> in 1982 from M.I.T. and her B.S. in 1980 from U.C. Berkeley. Before joining UCLA she was a member of the University of Southern California Computer Science Department from 1986 through the middle of 2000. In 1987, Professor Estrin received the National Science Foundation, Presidential Young Investigator Award for her research in network interconnection and security. During the subsequent

Fig 1 Semantic Annotation of Web Content

The annotation of significant concepts on such pages, such as a researcher's current job-title, academic degrees, alma-maters and dates for various academic degrees etc (as shown in Fig 1) can then enable Semantic-Web agent or integration applications over such data. Such annotation or mark-up tools are largely based on information extraction

technology. While information extraction itself is a widely investigated area, one still lacks powerful, general purpose, and yet easy-to-use frameworks and systems for information extraction, particularly the extraction of information from *free text* which is a significant fraction of the content on the open Web. In this chapter we describe XAR, a framework and system for free text information extraction and semantic annotation. XAR provides a powerful extraction and annotation framework by permitting the integrated use of hand-crafted extraction rules, machine-learning based extractors, as well as *semantic* information about the particular domain of interest for extraction. In this chapter we will describe the XAR framework which permits the integrated use of 1) Hand-crafted extraction rules, 2) Existing machine-learning based extractors, and 3) *Semantic* information in the form of database *integrity constraints* to power semantic extraction and annotation.

We have designed XAR to be an open-source framework that can be used by end-user application developers with minimal training and prior expertise, as well as by the research community as a platform for information extraction research. Over the last year we have used XAR for semantic annotation of Web documents in a variety of interesting domains. These applications range from the semantic annotation of details of particular events in online news stories in an overall application for internet news monitoring, to the semantic annotation of free text clinical notes as part of a business intelligence application in the health-care domain. This chapter is organized as follows. In the next section we provide an overview of XAR from a user perspective i.e., as a framework for developing extraction applications. We then present the technical details of our approach including the XAR system architecture, algorithmic issues, and implementation details. We present experimental evaluations assessing the effectiveness of the system in a variety of different domains. We also describe use case studies of application development using XAR in two different organizations. Finally, we discuss related work and provide a conclusion.

THE XAR SYSTEM

We first describe XAR from a user perspective i.e., as a framework for developing extraction applications and performing annotation tasks. The extraction step in annotation is treated as one of *slot-filling*. For instance in the researcher bios task, each Web page provides values for slots or attributes such as the job-title, academic degrees, dates etc. The two primary paradigms (Feldman et al., 2002) for automated information extraction systems are (i) Using hand-crafted extraction rules, and (ii) Using a machine-learning based extractor that can be trained for information extraction in a particular domain. Essentially, extraction applications in XAR are developed by using either hand-crafted extraction rules (Feldman et al., 2002) or machine-learning based extractors (Kayed 2006), which are further complemented with semantic information in the form of integrity constraints. We describe and illustrate each of these aspects.

Declarative Extraction Rules

XAR provides the user with a declarative Datalog style extraction rule language using which she can manually specify extraction rules for particular slots. These rules are essentially horn-clauses which state conditions based on which certain tokens get

assigned to certain slots in the extraction process. An example of such an extraction rule, continuing with researcher bios domain that we introduced, is:

phd-date(X) ← phd-degree(P), date(X), insamesentence(P,X) R1

which should be read as follows – “any token in the text that is of *type* date and is *in the same sentence* as another token of the type phd-degree, is a value for the phd-date slot”.

The rule head refers to a slot that is to be filled, for instance the rule R1 above is a rule to instantiate the **phd-date** slot. While we present the rule language in detail in the next section we wish to emphasize a couple of key aspects:

- (i) We see that the body of the rule R1 above contains predicates that describe properties of tokens in the text, for instance there are properties such as the type of the token i.e., whether of type date, degree etc. There are also predicates capturing relationships across tokens, for instance whether 2 tokens are in the same sentence etc. A space of such predicates describing properties of tokens and also their relationships is made available *automatically* to the user and she is abstracted from the details of their generation.
- (ii) Another key aspect of the XAR rule language is that it provides application developers the ability to represent, and (selectively) exploit features of *multiple* kinds and at different richness levels for information extraction. The rule R1 above illustrated predicates that capture only what are referred to as *shallow features* of the text. Shallow features are basically properties that can be determined by a shallow analysis of the text i.e., through the use of tools such as tokenizers, named-entity or other part-of-speech taggers, etc. The type of a token, its absolute or relative position in the text, etc., are all examples of shallow features. One can also distill what are called *deep features*, which are those obtained after a deeper analysis such as a complete natural language parse of the sentences in the text. A fact such as a token being a verb and another token being the subject of the first token is an example of a deep feature. We claim that there are advantages to having a framework that permits the availability and access to features at *multiple* richness levels, and their selective use in an adaptive fashion. We will look at this aspect in detail in the experimental evaluation section.

Using Machine-Learning Based Extractors

Besides hand-crafted extraction rules, a second paradigm for automated information extraction is the use of machine-learning based extractors and techniques. Typically a machine-learning based extractor is trained by providing the extractor with the slots to be extracted and training examples in the form of extracted data for several cases. The system then induces extraction rules which can be then applied to extract data from other unseen cases. There are several systems in this category such as RAPIER, WHIRL, TIES, LP2, and also CRF-based systems (Kayed, Girgis & Shaalan, 2006) to name a few. The XAR framework provides the option of using any off-the-shelf machine-learning based extractor.

The user essentially has a choice as to whether to use hand-crafted XAR extraction rules or any off-the-shelf machine-learning based extractor for an initial “basic” extraction step. This extraction is then further enhanced with the use of semantic information as integrity constraints.

Semantic Information as Integrity Constraints

The integration of semantic information as integrity constraints with either hand-crafted rule driven extraction or machine-learning driven extraction is one of the key features of the XAR framework. Table 1 below illustrates an example of the utility of integrity constraints in information extraction. We continue with the researcher bios domain, and the first row illustrates a specific portion of text that data is extracted from. The row below provides the results of information extraction for the slots of the PhD, Masters, and Bachelors degrees and their alma-maters and dates using a state-of-the-art extractor. Note that such extraction could have been done by either the hand-crafted XAR extraction rules or a machine-learning based extractor. Some of the extracted values are clearly erroneous. The XAR system then exploits 3 particular integrity constraints that have been specified for the researcher bios domain, namely:

- 1) The date of any degree is a SINGLE VALUE.
- 2) A person receives a doctoral degree only *after* his masters degree which in turn is received only after a bachelors degree (true for the same major at least).
- 3) There is contiguity in the universities or institutions a person attends for his various degrees (true in majority of the cases).

Table 1 Semantic Constraints in Extraction

Original Text								
<i>He received the PhD degree from Stanford University in 1966 and the BS and MS degrees from the University of Michigan in 1960 and 1961 respectively.</i>								
PhD	Stanford university	1966	MS	University of Michigan	1960 and 1961	BS	Stanford University	-
PhD	Stanford university	1966	MS	University of Michigan	1961	BS	University of Michigan	1960

With the knowledge of these 3 constraints the XAR system can correct the erroneous extracted values in to the (correct) values in the last row in Table 1. XAR provides a general purpose framework where 1) In any domain such semantics can be specified by a user as integrity constraints. We consider each semantic annotation task as that involving extracting a *relation*, for instance the researcher bios annotation task is essentially that of populating a researcher-bios relation that has attributes or slots such as an individual’s job-title, academic degrees, alma-maters, etc. 2) The XAR system applies such integrity

constraints over basic extraction performed using either hand-crafted rules or machine-learning.

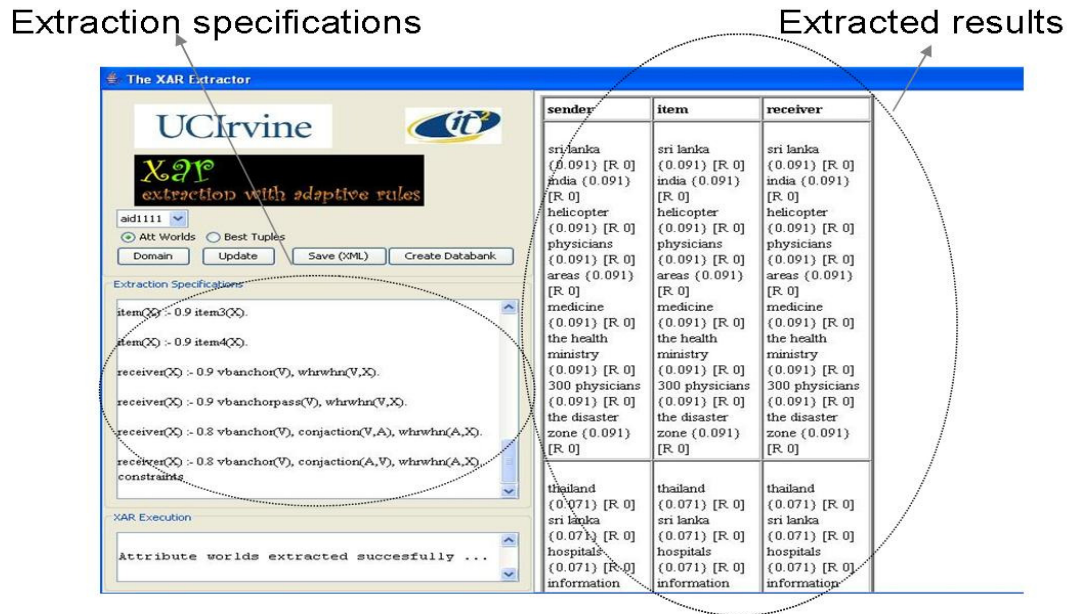


Fig 2 XAR Interface

The XAR user interface is illustrated in Fig 2. For any new application the user provides:

- (i) A **schema**, which is an SQL (Ullman & Widom, 2007) style schema describing the relation that is to be populated in the process of semantic annotation. The schema specifies the various slots or attributes in the relation to be extracted, the types and classes of such slots, whether single or multi-valued etc. In Fig 3 below we illustrate a schema for the researcher-bios domain.
- (ii) Basic extraction powering. This is done using:
 - a. A set of hand-crafted **extraction rules**, or
 - b. Using an off-the-shelf machine-learning extractor which has to be trained for each new extraction application.
- (iii) Semantic information about the relation to be extracted, in the form of **integrity constraints** (Ullman & Widom, 2007). For instance in the researcher bios domain we know that the first computer science degrees were awarded beginning only in the sixties. This is an example of an *attribute level* constraint which is constraining the value that the `phd-date` attribute can take in the researcher-bios relation. We specify this in the schema below. As an example of another constraint, we know that the year in which a person was awarded a doctoral degree must be greater (later) than the year in which he was awarded a bachelor's degree. This is an example of a *tuple level* constraint where we are specifying semantics *between* two extracted values. Finally we could also have constraints at the level of the relation, called *relation constraints* that ascertain properties that

the collection of tuples in a relation must satisfy as a whole. Some integrity constraints for the researcher bios domain are illustrated in Fig 3 below.

XAR Schema and Constraints
<pre>create table researcher-bios (name: person, job-title: title, employer: organization, phd-degree: degree, phd-alma-mater: organization, phd-date: date, master-degree: degree, master-alma-mater, master-date: date, bachelor-degree: degree, bachelor-alma-mater: organization, bachelor-date: date, previous-employers: organization) check phd-date > 1959 check phd-date > bachelor-date</pre>

Fig 3 researcher-bios schema and example integrity constraints

The notion of semantics in general is indeed very broad. One can consider semantics of many different kinds and consequently different formalisms to represent it. For instance semantics could be represented in something as simple as a lexicon, to a very complex formalism such as an ontology in an expressive ontology language. We have chosen the formalism of integrity constraints as a means of scoping the semantics we employ in this particular work. Besides, integrity constraints are a database formalism and thus lend themselves naturally to expressing semantics about relations, including relations that are to be extracted.

Integrity constraints in XAR are specified in SQL over the relation representing the information to be extracted. The XAR system extracts information using the application schema, basic extraction in the form of either extraction rules or a trained machine-learning based extractor, and semantic information in the form of integrity constraints. In the next section we provide the technical details of how the framework achieves this.

ARCHITECTURE AND TECHNICAL DETAILS

A schematic overview of the XAR system architecture is provided in Fig 4. The overall extraction process proceeds as follows: for any new extraction application and task the system first applies one or more text analyzers that extract different kinds of features from the text. Such features are essentially properties of and relationships amongst significant tokens and entities in the text. Having generated such features, the system then

applies one of declarative extraction rules or a machine-learning based extractor to perform basic extraction over the input text data. The output of such basic extraction is stored in an *uncertain* database relation (Dalvi & Suci, 2005). As a final step, semantic information in the form of integrity constraints is applied to this uncertain relation to *refine* it. This refinement essentially incorporates additional knowledge from the integrity constraints into the uncertain relation. The final extracted output is obtained from this refined uncertain relation.

We describe each of these aspects and steps in detail.

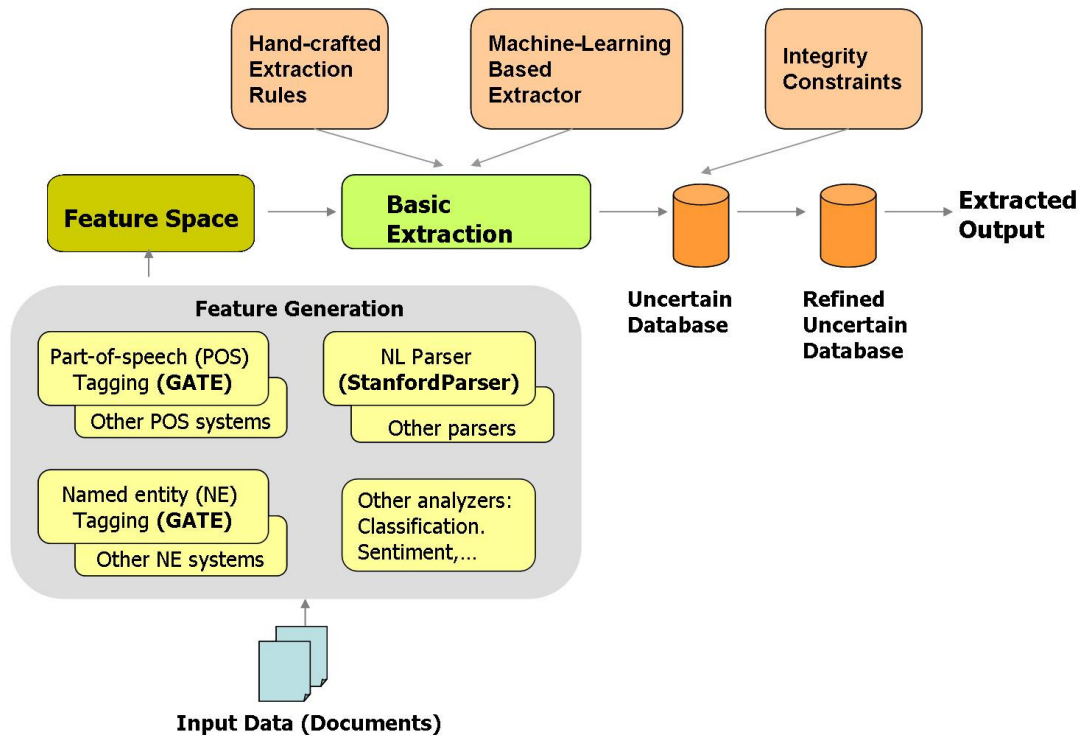


Fig 4 XAR System Architecture

Feature Generation

By default, XAR does a “shallow” analysis of any input text providing features such as the identification of significant tokens and entities, their types, position in the text, sentence demarcation etc. In the current implementation of XAR such shallow analysis is done using GATE (Cunningham, Maynard, Bontcheva & Tablan, 2002), which is an open-source framework for text analysis which we use for the identification of named-entities, other significant tokens, parts-of-speech, etc. Many types of important entities (such as person names, locations, organizations etc) can be recognized with reasonable accuracy and with no user input or training.

Let us consider an example to illustrate the kinds of features that are identified. Consider a sentence such as:

“He was awarded the university teaching excellence award in 1996 for exemplary undergraduate teaching.”

An analysis of the above sentence by GATE yields information about tokens in what is called a GATE annotation:

```
AnnotationImpl: id=81; type=Token; features={category=NN, kind=word, orth=lowercase, length=5, string=award}; start=NodeImpl: id=80; offset=226; end=NodeImpl: id=81; offset=233
```

For instance in the annotation above, the token “award” has been identified as a noun and other properties such as its position, and offset in the text are also identified. We extract the information from such GATE annotations (using wrappers) and represent it in predicates. Optionally, we can use a second text analysis tool, in this case a natural language parser for “deep” analysis of the text. The particular parser we have used is the StanfordParser (StanfordParser, 2008) which is also an open-source system that is a complete (statistical) natural language parser and (like GATE) can be used “as-is” i.e., without any additional user input or training. The StanfordParser can parse the same sentence and provide an output such as:

```
nsubjpass(awarded-3, He-1)
auxpass(awarded-3, was-2)
det(award-8, the-4)
nn(award-8, university-5)
nn(award-8, teaching-6)
nn(award-8, excellence-7)
dobj(awarded-3, award-8)
prep_in(award-8, 1996-10)
amod(teaching-14, exemplary-12)
amod(teaching-14, undergraduate-13)
prep_for(awarded-3, teaching-14)
```

which is a *typed dependencies collapsed* representation of the parse tree of this sentence. The typed dependencies representation is essentially a representation of the relationships in the parse tree of a sentence, in relational form. We extract important information about actions of interest from such typed dependencies. For instance in this example, the action “awarded” is of interest. From the typed dependencies we can (in many cases) extract who awarded what and to whom. In fact such associations (subject, object etc.) are typical of literally any action i.e., verb. As with GATE annotations, the extraction of such information from a typed dependency representation is done through a wrapper for such a representation.

.Table 1 FEATURE and RELATIONSHIP Tables

FEATURE

<i>OBJECT</i>	<i>PROPERTY</i>	<i>VALUE</i>
award	type	thing
1996	type	date
1996	position	33
d57	category	bio
....		

RELATIONSHIP

<i>RELATION</i>	<i>OBJECT1</i>	<i>OBJECT2</i>
member	award	s3
what	awarded	university-teaching-excellence-award
member	s3	d57
....		

One could also consider other kinds of text analysis depending upon the task at hand. For instance we may use a text categorizer to determine what category (say sports, business, politics etc) each document belongs to. Or one may use a “sentiment classifier” (Pang 2002) to classify the sentiment in each sentence (positive, negative, neutral). How do we represent such diverse features and relationships ? We note that features of objects, regardless of how they are obtained, are of two kinds. We have *properties* of individual objects and we have *relationships* between objects. Any property then is essentially a 3-ary relationship i.e., we have a certain property of a certain object having a certain value. Relationships on the other hand can be n-ary in general. Any n-ary relationship can however be converted into a set of (n) binary relationships (a unique identifier serves to glue the binary relationships together). We thus represent properties in a FEATURES table (Table 1) with three columns i.e., the object, the property, and the actual value. Relationships are captured in a RELATIONSHIPS table as shown in Table 1 which shows a number of interesting relationships of different kinds.

As a final step, the entries in the FEATURES and RELATIONSHIPS tables are converted to logical predicates representing the same information. The set of predicates capturing properties and relationships for all tokens in all text segments that data is to be extracted from, form the feature databank for that dataset. An important aspect to note is that the two FEATURES and RELATIONSHIPS tables provide a unified representation for features obtained from different kinds of text analyzers and at different semantic levels. In logic database parlance, the feature databank forms what is the extensional database or “EDB”.

Basic Extraction

We describe the two options for powering basic extraction in XAR.

XAR Extraction Rules

The first option for achieving basic automated information extraction in XAR is with the use of hand-crafted extraction rules. The XAR extraction rules are essentially Datalog rules with syntactic sugar. Each rule is a horn-clause of the form:

$$S(X) \leftarrow C \ B1, B2, \dots, Bm$$

where S , B_i s are atoms. S , the head corresponds to a slot to be extracted, the B_i s are predicates corresponding to conditions based on which tokens are assigned to slots, and C is an (optional) confidence value [0,1] that is a measure of the precision of the extraction rule. C reflects the maximum confidence with which we can state that a value inferred for the head predicate S by that rule is actually a value for the slot corresponding to S . The following are the key features of this rule language:

1. *The rules are essentially horn-clause style rules.* This follows from the rule definition above.

2. *The predicates in the body may be either slot predicates or feature predicates.*

$$\text{phd-date}(X) \leftarrow 0.7 \text{ phd-alma-mater}(Y), \text{ date}(X), \text{ insamesentence}(X,Y) \quad \mathbf{R2}$$

The rule R2 above provides an example where in the rule body one of the predicates refers to another slot i.e., `phd-alma-mater`.

3. *A token is assumed to be “consumed” with the application of a rule, unless stated otherwise.*

Consider a set of predicates and rules such as:

`country(usa)`

$$\text{sender}(X) \leftarrow \text{country}(X) \quad \mathbf{R3}$$

$$\text{receiver}(X) \leftarrow \text{country}(X) \quad \mathbf{R4}$$

In a traditional logic program we would infer both sender (i.e., “usa”) and receiver (i.e., “usa”) to be true after the application of the rules above. In the XAR semantics however a token can be “used” by only one rule. We can infer *either* sender(usa) or receiver(usa) but not both. To clearly define the semantics in such cases we consider any pair of rules whose bodies could be satisfied by the same token(s) as competing rules. A relative priority order must be explicitly specified between any pair of competing rules. For instance in the above example if we gave a higher priority to R3 over R4, we would infer sender(usa) but not receiver(usa). Of course there may be cases where we allow for the same token to be consumed (simultaneously) by more than one rule, i.e., as in regular logic programming. If no explicit order is specified between two competing rules then this traditional semantics is used.

4. *Multiple rules are permitted for the same slot.*

5. A precision value is (optionally) associated with each rule.

The precision value, if stated explicitly, is interpreted as a lower bound on the confidence that a value inferred by the rule is actually a value for the slot associated with the head of the rule. In rule R2 for instance we have a confidence of (at least) 0.7 that any value inferred by this rule is a value for the slot 'phd-alma-mater'. In case multiple rules infer the same value for a particular slot, we simply associate the precision value that is highest with that value from amongst the different rules that inferred that value. In case no precision value is explicitly stated, it is assumed to be 1. The probabilistic framework is adapted, with simplification, from a general probabilistic logic framework developed in (Lakshmanan & Sadri, 1994). The work proposes both *belief* and *doubt* rules that capture in horn-clause form why a certain fact should (or should not be) true. A notion of probabilistic confidence (or rather a range defined by a lower and upper bound) is associated with each fact (predicate) and rule. A probabilistic algebra and semantics is also described. In our framework we, at present, use only belief rules i.e., rules that state when a token should be a value for a slot. In extraction there may be room for doubt rules as well though we have not investigated the merits of this yet. We associate confidence values only with rules and not with predicates (i.e., in the rule body) at this point. Again this is a level of detail that we could incorporate in future. The semantics of associating confidence values with inferred values in our framework is simpler as well; we simply associate with each inferred value the highest value amongst the precision values of the rules that led to its inference.

6. Negation is permitted in the rule body.

We allow for writing rules with negated predicates in the rule body (only). However we make the requirement that the set of rules be *stratified* wrt negation (no cyclical dependencies amongst rules that involve negation), this is to ensure consistent fixpoint semantics. For instance:

tempsender1(X) \leftarrow anchor(A), who(X,A) R5

tempsender2(X) \leftarrow anchor(A), immbefore(X,A), not tempsender1(Z) R6

illustrates two intermediate rules where R6 is satisfied only if R5 is *not* satisfied.

7. Some predefined predicates are provided for the user's convenience in writing rules.

These include predicates for determining things properties and relationships such as whether one token is before or after another in the text, references to ground values of tokens, choosing the first or last element in a set of tokens, etc.

Application of Rules

A set of rules in the extraction language above can be translated to a regular Datalog program in a straightforward fashion. For instance in the case of token consumption, the capability of prioritizing rules can be achieved in regular Datalog by simply using negation i.e., if a rule R1 has a higher priority than another rule R2, then the body of R2

is augmented with the predicate not R1(X). The association of precision values or support for the user defined predicates can also be provided, we do not provide the translation details here. The complexity of inference with any extraction language is a valid concern. Inference in the extraction language we have described is tractable, in fact the inference can be done in time polynomial in the number of extraction rules and/or the number of tokens or entities in each segment. This is because any set of rules in the XAR extraction language can be translated to a set of rules in regular Datalog, the number of rules in the translated Datalog program being (at most) twice the number of original XAR extraction rules. (Bottom-up) inference in regular Datalog (including Datalog with stratified) negation is polynomial in the number of rules and/or base predicates (Ullman, 1988).

Using a Machine-Learning Based Extractor

The second option for basic extraction is to use any machine-learning based extractor. The XAR framework allows for the integration of any machine-learning base extractor. Any such extractor is treated as a black box and is provided with a set of training examples. The extracted output is represented in an uncertain relation. So far we have integrated in extractors as AutoSlog, RAPIER, and TIES (TIES, 2008).

Uncertain Database Representation

The output of basic extraction, whether done using hand-crafted XAR rules or a machine-learning based extractor, is represented in an *uncertain* database relation. An uncertain database relation (Dalvi & Suciu, 2005) allows for representing uncertainty in database relations where we essentially represent a space of possible relations associated with a probabilistic distribution. In the case of using XAR rules the realization of the intensional database provides us for each head predicate, a set of (zero or more) values that satisfy that predicate each associated with a confidence value. Each such value is then treated as a possible value for the slot that that head predicate corresponds to. At the end of such logical inference we have with each slot associated a set of possible values for that slot, each value associated with a probabilistic confidence. The set of such values is used to generate an *attribute world* associated with that attribute which is a probabilistic space of possible values for that slot. An uncertain database relation representation allows us to capture such attribute worlds for each slot. When using a machine-learning based extractor, many such systems now provide a *set* of possible extracted values for a slot as opposed to a single value. Each such possible value is associated with a confidence score. We translate the set of possible values for each slot and their associated confidences to attribute worlds which are then represented as an uncertain database relation.

name	job-title	phd-alma-mater	phd-date
jim	professor	Oxford University (0.7) University of Oregon (0.3)	1995 (0.6) 1992 (0.4)
mary	scientist	MIT	2000 (0.5) 1995 (0.5)

An example of an uncertain relation showing a subset of the attributes for the researcher-bios domain is illustrated above, which shows how the uncertainty for the `phd-alma-mater` and `phd-date` attributes is represented.

Integrating Semantic Information as Integrity Constraints

At the end of the basic extraction step, using either hand-crafted extraction rules or a machine-learning based extractor, we have the extracted data in an uncertain relation. The application of integrity constraints is essentially a process of *refining* the uncertain extracted relation with the knowledge in the integrity constraints. The refinement results in a recalibrated uncertain extraction relation in which the possibilities inconsistent with the integrity constraints are eliminated. The additional knowledge thus provided by the integrity constraints is taken into account in that instances inconsistent with such constraints are eliminated from consideration.

The general problem of systematically incorporating integrity constraints into uncertain relations is complex. We provide a study of this problem including a scalable practical solution as part of separate work which is described in our technical report on the topic (Ashish, Mehrotra & Pirzadeh, 2008).

System Implementation and Availability

The current version of the system is implemented in Java and also incorporates some other off-the-shelf tools. Shallow feature analysis is done using GATE and deep feature analysis is done using the StanfordParser. TuProlog (Denti, Omicini & Ricci, 2001), a Java based prolog engine, is used for the rule language inference. The output of the extraction and eventually the semantic annotation is made available in (i) XML format. Essentially in each document or Web page we put XML tags around the values for the various slots identified, as illustrated in Fig 1. (ii) As RDF triples. Each tuple is treated as a resource. For instance in the researcher bios domain each Web page corresponds to an individual and is treated as a resource. The different slots or attributes to be annotated are essentially properties of each resource and their actual values are the values of these properties. This lends itself to a natural RDF representation and XAR provides the option of saving identified semantic annotations as RDF triples.

An open-source version of XAR has made available for community use under a Creative Commons License. We encourage potential users interested in either developing extraction applications or researching information extraction to consider using this system. The system source code and documentation is available at <http://www.ics.uci.edu/~ashish/xar> . We have also provided the details of a number of applications developed using XAR including the application schemas, XAR extraction rules, and semantic integrity constraints. We believe this will be illustrative to new users wishing to develop extraction and annotation applications using this framework.

EXPERIMENTAL EVALUATION

As mentioned earlier we have applied XAR for semantic annotation in a number of interesting domains. In some of these domains we have also empirically assessed the effectiveness of the XAR framework for semantic annotation. We have made the empirical evaluations from two primary perspectives: **1) The Extraction Rule Language and Features:** We have provided a comprehensive declarative rule language for extraction. We have also provided a framework where features at multiple semantic levels can be made available and the rule language can exploit such features in a seamless fashion. The first aspect of our evaluation assesses this declarative rule language, we evaluate the effort and complexity of writing extraction rules for different annotation tasks, we also assess the accuracy of the extraction achieved as a result. We then also quantitatively assess the benefits of access to an integrated space of features at multiple semantic levels, **2) The Integration of Semantic Information:** We quantitatively evaluate how the integration of semantic information as integrity constraints improves on the basic extraction done by either hand-crafted rules or machine-learning based systems.

Extraction Domains and Datasets

We have conducted the quantitative assessments on three different real-world datasets and extraction tasks, namely a) The researcher-bios extraction task over a corpus of 500 computer science researcher bios on the open Web, b) The MUC-6 (MUC, 1995) task of extracting management succession events (close to a 100 such instances) over a corpus of WSJ news stories, and c) A task of extracting details about instances of aid or relief being or having been dispatched or sent by a country or organization to another in the event of a disaster (see illustration below). This is over a corpus of 4000 online news stories related to the S. E. Asian Tsunami disaster. We will refer to these as the researcher-bios, management-succession, and aid-dispatched tasks respectively.

Many countries have sent aid supplies and relief materials. The United States dispatched medical supplies to Indonesia. Besides this there



Many countries have sent aid supplies and relief materials. The <sender> United States</sender> dispatched <item> medical supplies</item> to <receiver> Indonesia</receiver>. Besides this there

aid-dispatched semantic annotation

Extraction Rule Language and Features

This evaluation was conducted over the aid-dispatched and management-succession extraction tasks. The following are the schemas for the relations that we extracted.

aid-dispatched (sender: organization country, item: thing, receiver: thing)

management-succession(name: person, company: organization, status [IN/OUT])

Extraction Rules

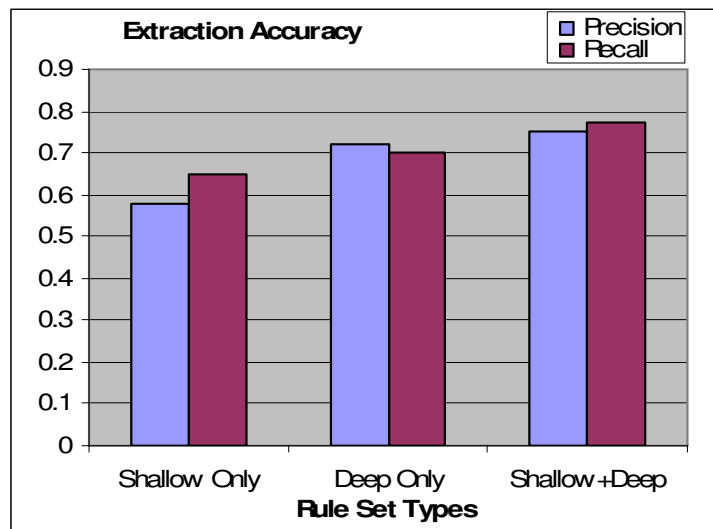
For each of these extraction tasks we wrote XAR extraction rules based on observations of regular patterns in the text. We considered rule sets of different types i.e., sets of rules over only shallow features, sets of rules over only deep features, and sets of rules over both shallow and deep features. Table 2 provides a listing of some of the XAR extraction rules (of these different types) for the aid-dispatched domain, for the “sender” slot.

Table 2 Example of XAR rules for aid-dispatched

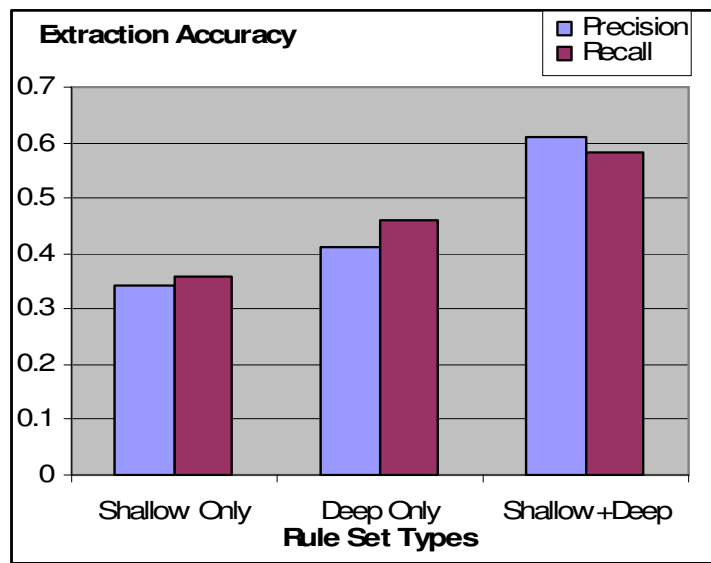
Shallow Only Rules
sender(X) \leftarrow 0.8 sender1(X).
sender(X) \leftarrow 0.6 sender2(X).
sender1(X) \leftarrow anchor(V), location(X), before(X,V), insamesentence(X,V), not myout(X).
sender1(X) \leftarrow anchor(V), organization(X), before(X,V), insamesentence(X,V), not myout(X).
sender2(X) \leftarrow anchor(V), location(X), before(X,V), insamesentence(X,V), not sender1(Z).
sender2(X) \leftarrow anchor(V), organization(X), before(X,V), insamesentence(X,V), not sender1(Z).

Deep Rules
sender(X) \leftarrow anchor(V), who(V,X).
item(X) \leftarrow anchor(V), what(V,X).

We first show (Fig 5) the extraction quality obtained with different sets of extraction rules for the aid-dispatched and management-succession tasks. We provide measures of extraction precision and recall, aggregated over all slots. The results show that we can obtain fairly good extraction quality i.e., in the range of precision and recall as high as 0.7 or above. The number of extraction rules required is reasonable as well. For aid-dispatched, a total of 18 rules are used in the case of shallow rules only, for MUC we use a total of 9 shallow rules. The number of rules required when using deep rules only is much less, for instance for aid dispatched we use only 3 rules, one rule per slot.



(a) aid-dispatched



(b) management-succession

Fig 5 Extraction accuracies for various sets of rules

Of course this is because we have access to the semantic predicates where much of the extraction information has already been captured. The combination of both shallow and deep rules provides an even higher boost to the extraction accuracy in both the domains. The reason we see an improvement, over the case of using deep rules only, is that deep rules have their limitations as well. In some cases sentences are quite complex (for instance describing several events or facts) and synthesizing information from the resulting (complex) semantic parse may be difficult.

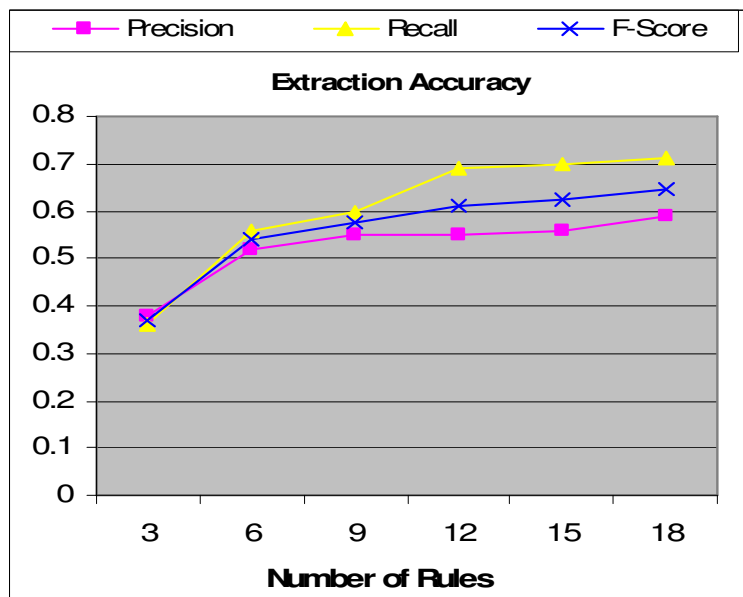


Fig 6 Extraction accuracy vs number of rules.

The number of extraction rules for an application is an estimate of the effort and complexity in developing the extraction application. In Fig 6 below we show, for the aid dispatched domain, how the extraction accuracy increases as we increase the number of rules used (shallow rules only).

Useful Features

In writing rules the user can choose from rules and predicates in such rules of different kinds. For instance we make a distinction between shallow and deep rules and the above rules indicate that deep rules generally capture more information, high accuracy extraction is obtained with relatively much fewer rules. The predicates themselves represent either a feature (property) or a relationship. Such features or relationships can further be of different kinds. Apart from the shallow vs deep distinction, amongst the shallow features we identify the categories of (i) named-entity features, (ii) other part-of-speech features, and (iii) “structural” features such as the position of a token in the segment, which sentence it is part of etc. We make this distinction as different kinds of analyzers are required to extract these different types of features. From an extraction perspective a matter of interest is the relative importance or effectiveness of features of different kinds in extraction.

<i>Predicate Categories</i>	<i>Examples</i>
POS: Part-of-speech features	action(X), to(T), thing(X)
NE: named-entity features	organization(X)
TR: Structural features	position(X,33)
REL-TR: Structural relationships	insamesentence(X,T), before(X,Y)
SEM-TR: Semantic relationships	who(X,Y), what(X,Y)

Type	POS	NE	TR	REL-TR
Freq %	21	27	16	36

Fig 7 Frequency of Predicates

The usage frequencies (percentage) of the different feature and relationship types, determined by the application of XAR extraction rules in the above domains, are displayed in 7. We observe that features or relationships of each of the four different classes are used with considerable frequency, also there is no one class of features or relationships that is used with overly high or low frequency. One of the design decisions one can make based on this frequency distribution is that relatively equal resources must be invested towards low level text analyzers of all kinds, so that all the above categories of text features are accurately and comprehensively extracted and made available to the rule writer. A more skewed distribution, on the other hand, would direct that we bias more resources towards text analyzers that generate features that are more commonly used.

Features at Multiple Semantic Levels

Certainly, writing rules over deep features is simpler where high extraction accuracy can be achieved with using very few rules as demonstrated. One issue however with using deep features is the cost of feature extraction. Deep analysis, such as that based on natural language parsing of any sentence, takes time that is an order of magnitude larger than that required for shallow analysis. Consider the following comparison. The average processing time for shallow feature extraction per sentence, using GATE, is 132 ms. The experiments were conducted on a DELL Inspiron 19200 machine with an Intel Pentium 1.8 GHz processor, 1GB RAM, running Windows XP. Under the same configuration, the average processing time per sentence when using the StanfordParser, is 7.8 sec ! In a real-time setting such a high per sentence processing time for feature extraction may be unacceptable. Having access to an *integrated* space of shallow and deep features provides us with the flexibility of using shallow and deep features in an adaptive manner. We have developed an approach where the deep analysis of sentences is done only selectively, for those sentences that are determined as “complex”. A sentence is determined as complex based on heuristics, such as the presence of multiple verbs in the sentence, the presence of many (a relatively large number of) entities or tokens in the sentence etc. Deep features are extracted and made available for (only) the complex sentences while shallow features are extracted and made available for the entire text.

Table 3 shows cases where we used different (logical) conditions to flag a sentence as complex, and the associated (total) processing time for feature extraction (for a corpus of 285 documents) and the extraction accuracy achieved. We observe that with the right set of conditions (to determine whether a sentence is complex) one can achieve a fairly high extraction accuracy (albeit not as high as using deep rules for all cases) and still keep the processing time modest. The percentage of sentences identified as complex is also shown, which directly reflects on the total processing time. It should be obvious that such conditions should be chosen carefully, a condition that is too conservative will lead to lower processing time but also lower extraction quality. On the other hand a condition that is too relaxed will lead to higher extraction quality but with a high processing time.

Table 3

<i>Complexity condition</i>	<i>Extraction F-Score</i>	<i>% sentences complex</i>	<i>Processing Time (sec)</i>
Multiple verbs in sentence	0.71	55	300
Multiple verbs in certain portion of sentence	0.68	43	234
Multiple verbs, and presence of commas and “and” token	0.64	27	162
Number of entities greater than threshold	0.7	59	318

This experiment also demonstrates the advantages of having an integrated space of both shallow and deep features. We are able to strike a balance of achieving high extraction accuracy as well as keeping the processing time for feature generation under control.

Integration of Integrity Constraints

Finally, we have extensively evaluated the effectiveness of incorporating semantic information as integrity constraints. We consider both the options for basic extraction i.e., with hand-crafted XAR rules and with a state-of-the-art machine-learning based extractor and evaluate the improvement in extraction accuracy obtained as a result of incorporating constraints. As this particular aspect of the work is the central topic of other technical reports and papers we do not describe it in detail here. We will summarize the results with the facts that we were able to state meaningful integrity constraints for each of the 3 domains we evaluated here, the number of integrity constraints varied from 6 to over 40 in one of the domains. We achieved a significant increase in extraction accuracy with the incorporation of integrity constraints, achieving about as much as 40% improvement in the F-measure (per slot, averaged over all slots) in one of the domains. We refer to (Ashish, Mehrotra & Pirzadeh, 2008) for a detailed description of these experiments and results.

APPLICATION CASE STUDIES

We describe the use cases of the XAR system in the context of application development for two organizations. The first use case we describe was developed for and in collaboration with the IT department of a city emergency and disaster response government organization. The names and references of both organizations for which use cases are described have been kept confidential upon request. The organization is responsible for effective and prompt response in natural and man-made disaster situations in the city, it is also responsible for providing information awareness and other services to citizens on a continual basis i.e., even in non emergency situation times. The organization relies significantly on information technologies to provide such services effectively and efficiently. The Disaster Portal (DisasterPortal, 2008) is a flagship example of an internet information service provided by the organization to the city public. One of the important tasks that personnel in such an organization are involved with immediately after or during a disaster is to monitor the spread of rumors among the community regards the disaster. Rumor information such as erroneous or factually wrong reports of serious issues such as the presence or number of casualties, infrastructure damage etc., can have negative consequences in the disaster response. The internet with several online local news sites and other sources is thus also a potential modality for rumors, as such a significant analyst effort is spent is manually monitoring internet information in a disaster. To alleviate this burden we are developing an automated “Internet News Monitoring” module for the Disaster Portal. The task of this module is to poll information at various local internet news sources and indentify key events and facts reported related to a disaster. These include facts such as casualties and injuries, reports of road closures or schools shut downs etc. Events and facts extracted by XAR are then fed into a database that the Disaster Portal can access. In the development of this application the definition of what news events and facts are of interest and what details in these events or facts are of interest was identified by IT analysts at the disaster response organization. Based on these requirements, developers familiar with XAR developed extraction rules for this domain.

The second use case we describe is an application developed for and in collaboration with the IT department of a county level community clinic. This organization provides voluntary health services to underserved populations in the area and receives significant monetary aid as well as aid in the form of food items, medications, etc., from the local community. It relies significantly on the use of IT for tasks such as maintaining databases of patient records, inventories of donated items, referrals of cases to other hospitals and organizations, etc. An important piece of information associated with each patient for each visit is a “clinical note” which is a free form text note of information capturing information about the patient, case history, medications recommended, progress, etc. For aggregate reporting applications, such as an annual report of the organization, thousands of such clinical notes have to be analyzed manually. Key details are extracted from the notes, entered into a database and then integrated with other information such as demographic or local geographic information. We developed an application where key details are extracted from clinical notes in an automated fashion using the XAR system.

The experience of application development in both organizations, challenges faced, benefits achieved, and current status are provided in Table 4 below.

Table 4 Application Experience

	Requirements and Task Definition	Effort	Reported Benefit	Status
Disaster Response Organization (IT Department)	Organization IT personnel were able to do this without difficulty. Organization personnel not comfortable with rule language and SQL. Task left to XAR developers.	4 man weeks	<ul style="list-style-type: none"> • Improved coverage of news information. • Significant reduction in news monitoring and analysis time. 	Adopted by organization and in active use.
Community Clinic (IT Department)	Organization IT personnel were able to articulate the needs clearly.	6 man weeks	<ul style="list-style-type: none"> • Improved follow up with individual patients. • Significant reduction in time and effort for aggregate reporting. 	Prototype being evaluated by organization.

As Table 4 shows, both organizations were able to clearly articulate their information integration and consequent semantic annotation requirements. However the application development using XAR was not something that could be achieved by the average analyst in either organization. This highlights that a reasonable level of SQL and logic programming expertise is required to use the XAR system and that we must develop mechanisms using which the skill level required to use XAR is lowered. The effort in developing these applications, of the order of a few man weeks, is reasonable given the scope of the applications. There is also significant benefit to both organizations. The

disaster response organization has integrated access to disparate news information sources for monitoring rumors. The community clinic has automated access to structured information in thousands of clinical notes that further enables integration with other key information. In both cases, there is significant savings in time and effort for annotation tasks that were done manually prior to the application of the XAR system.

RELATED WORK

The problem of semantic annotation has been recognized since the early efforts towards the Semantic-Web. (Carr, Bechhofer, Goble & Hall, 2001) and (Kahan, Koivunen, Prod'Hommeaux & Swick, 2001) describe frameworks for semantic annotation of documents in a manual and collaborative fashion. A Dagstuhl workshop on "Machine Learning for the Semantic-Web" (Ciravegna, Doan, Knoblock, Kushmerick & Staab, 2005) highlighted the critical need for automated technologies for semantic annotation. KIM (Kiryakov, Popov, Terziev, Manov & Ognyanoff, 2004) is a framework for semantic annotation that exploits domain knowledge as ontologies i.e., the extracted output is improved with the knowledge present in domain ontologies. A similar semantic annotation system is SemTag (Dill et al., 2003) that makes use of the (precompiled) TAP knowledge base for annotation. Earlier efforts include systems such as S-CREAM (Handschuh, Staab & Ciravegna, 2002) that are based on machine-learning techniques for semantic annotation. (Michelson & Knoblock, 2005) is representative of work on automated semantic annotation of semi-structured data such as Web pages. The area of information extraction per se is well investigated with many systems and techniques for automated extraction including that at the slot-filling level. (Kayed et al., 2006) provides an extensive survey of many such systems and approaches. The XAR framework advances these efforts in many ways. First, from a user perspective, it abstracts the user from having to generate features of the text which are essential to any automated extraction over the text. The user can develop extraction and annotation applications at a high level with only a working knowledge of Datalog or Prolog style rules and SQL. Next, we have provided a comprehensive declarative extraction rule language. While many other systems such as DIAL (Feldman et al., 2002), XLog (Shen, Doan, Naughton & Ramakrishnan, 2007), and Lixto (Gottlob, Koch, Baumgartner, Herzog, & Flesca, 2004) provide logic based extraction rule languages, the XAR language provides more comprehensive capabilities. Specifically, we allow for the inclusion of probabilistic reasoning with the rule language, provide higher level predicates capturing text features and relationships, and define and support advanced features such as token consumption and stratified negation in the rule language and semantics. Finally, a pioneering aspect of XAR is that it allows the incorporation of semantic information as integrity constraints in the extraction and annotation process. We have demonstrated, through evaluations in several domains, how the use of integrity constraints improves on the extraction obtained with any existing approaches.

CONCLUSIONS

In this chapter we described the XAR framework for semantic extraction and annotation. There are several interesting directions of future work. Integrity constraints are but one form of semantic information about a domain. We can consider semantics in the other forms for instance that present in ontologies that could be applied to the task. Or one may

also consider semantics that may be present *in* the data itself, for instance patterns that we could mine in the available data and apply it for improving extraction.

Finally, we encourage the community to use the available XAR system for semantic annotation tasks and welcome feedback.

ACKNOWLEDGEMENTS

The authors wish to acknowledge Pouria Pirzadeh and Zheng Zhang, who led the efforts on the experimental evaluation of the system.

REFERENCES

- Shen, W., Doan, A., Naughton, J., and Ramakrishnan, R. (2007). Declarative information extraction using datalog with embedded extraction predicates. Proc ACM SIGMOD 2007
- Gottlob, G., Koch, C., Baumgartner, R., Herzog, M., and Flesca, S. (2004). The lixto data extraction project - back and forth between theory and practice. Proc ACM PODS 2004
- Cunningham, H., Maynard, D., Bontcheva, K., and Tablan, V (2002). GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL'02). Philadelphia, July 2002.
- Naughton, M., Kushmerick, N., Carthy, J. (2006). Clustering sentences for discovering events in news articles. ECIR, 2006,
- Lakshmanan, L., and Sadri, F. (1994). Probabilistic deductive databases. SLP 1994
- Ullman, J.D. (1988). Bottom-up beats top-down for datalog. ACM PODS 1988
- Feldman, R., Aumann, Y., Finkelstein-Landau, M., Hurvitz, E., Regev, Y., and Yaroshevich, A. (2002). A comparative study of information extraction strategies. ACL 2002
- Jayram, T.S., Krishnamurthy, R., Raghavan, S., Vaithyanathan, S., and Zhu, H. (2006). Avatar Information Extraction System. IEEE Data Engineering Bulletin, 2006.
- Kayed, M., Girgis, M.R., and Shaalan, K.F. (2006). A Survey of Web Information Extraction Systems. IEEE Transactions on Knowledge and Data Engineering, vol. 18, 2006.
- Ceri, S., Gottlob, G., and Tanca, L. (1989). What you always wanted to know about Datalog (and never dared to ask). IEEE Transactions on Knowledge and Data Engineering 1(1), 1989, pp. 146–66.
- Proceedings of the 6th message Understanding Conference, MUC-6, Columbia MD, 1995
- Pang, B., Lee, L., and Vaithyanathan, S. (2002). Thumbs up? Sentiment classification using machine learning techniques. Proceedings of EMNLP.
- Denti, E., Omicini, A., and Ricci, R. (2001). tuProlog: A Light-Weight Prolog for Internet Applications and Infrastructures. PADL 2001, Las Vegas, NV
- ACE Automatic Content Extraction <http://www.nist.gov/speech/tests/ace/>
- Dalvi, N., and Suciu, D. (2005). Foundations of Probabilistic Answers to Queries. Tutorial, ACM SIGMOD 2005
- Berners-Lee, T., Hendler, J., and Lassila, O. (2001). The Semantic Web. Scientific American 284(5):34-43
- Carr, L., Bechhofer, S., Goble, C., and Hall, W. (2001). Conceptual linking: Ontology based open hypermedia. WWW10 Conference Hong Kong

Handschuh, S., Staab, S., and Ciravegna, F. (2002). S-CREAM: Semi-automatic CREation of Metadata. EKAW 2002

Kahan J., Koivunen, M., Prod'Hommeaux, E., Swick, E. (2001). Annotea: An open RDF infrastructure for shared web annotations. WWW10 Conference, Hong Kong.

Dill S., Eiron, N., Gibson, D., Gruhl, D., Guha, R., Jhingran, A., Kanungo, T., Rajagopalan, S., Tomkins, A., Tomlin, J.A., and Zien, J.Y. (2003). SemTag and Seeker: Bootstrapping the semantic web via automated semantic annotation. WWW12 Conference Budapest.

Kiryakov, A., Popov, B., Terziev, I., Manov, D., and Ognyanoff, D. (2004). Semantic annotation, indexing and retrieval. Journal of Web Semantics. 2(1), pp49-79

TIES 2008. "TIES: Trainable Information Extraction System," <http://tcc.itc.it/research/textec/tools-resources/ties.html>

Michelson, M. and Knoblock, C. (2005). Semantic annotation of unstructured and ungrammatical text. *IJCAI*, 2005

Ciravegna, F., Doan, A., Knoblock, C., Kushmerick, N., and Staab, S. (2005). Machine Learning for the Semantic Web. Seminar 05071 at Schloss Dahstuhl 2005

Ashish, N., Mehrotra, S., and Pirzadeh, P. (2008). Incorporating Integrity Constraints in Uncertain Databases. UCI Technical Report 2008. Online at <http://www.ics.uci.edu/~ashish/techreport>

Ullman, J. and Widom, J. (2007). A First Course in Database Systems. Prentice Hall, 2007.

StanfordParser (2008). Web: <http://www-nlp.stanford.edu/downloads/lex-parser.shtml>

DisasterPortal (2008) Web: <http://www.disasterportal.org>