Purple SOX Extraction Management System

Philip Bohannon[♭], Srujana Merugu[♭], Cong Yu[♭], Vipul Agarwal[♭],
Pedro DeRose[♮], Arun Iyer[♭], Ankur Jain[♭], Vinay Kakade[♭],
Mridul Muralidharan[♭], Raghu Ramakrishnan[♭], Warren Shen[♮]
[♭]Yahoo! Research [♮]University of Wisconsin Madison
plb@yahoo-inc.com

ABSTRACT

We describe the Purple SOX (PSOX) EMS, a prototype Extraction Management System currently being built at Yahoo!. The goal of the PSOX EMS is to manage a large number of sophisticated extraction pipelines across different application domains, at the web scale and with minimum human involvement. Three key value propositions are described: *extensibility*, the ability to swap in and out extraction operators; *explainability*, the ability to track the provenance of extraction results; and *social feedback support*, the facility for gathering and reconciling multiple, potentially conflicting sources.

1. INTRODUCTION

Most documents, including electronic documents such as web pages or emails, are created for *human* consumption. Nevertheless, significant value may be added by processing these documents computationally to extract entities and/or structured data. For example, it may be useful to a financial services firm to analyze news stories for rumors of corporate takeovers, or to a consumer to determine the price at which an item of interest is being offered for sale on a particular vendor's web page. These tasks (and many others) are examples of Information Extraction (IE) (see, e.g. [11]). While techniques for IE have steadily improved, the dramatic growth of the Web strongly motivates continued innovation in this area. For example, a variety of popular Internet portals are based on structured information, some or all of which is automatically extracted from other web pages. Examples include ZoomInfo, OpenClinical, and Citeseer, which deal with professional, medical and bibliographic information respectively.

Purple SOX (Socially Oriented eXtraction) is an information extraction project at Yahoo! Research with two key goals. First, PSOX should develop extraction operators capable of working well on a semantic domain even across sites that format that information differently. Second, PSOX should develop an Extraction Management System, "PSOX EMS", that supports rapid development and large-scale deployment of on information extraction operators and pipelines, and the ability to accept and effectively manage intermittent and noisy "social" feedback on the quality of extracted data. The current prototype of the PSOX EMS is described in this paper.

The need to effectively manage information extraction has been recently discussed [7, 5]. The job of any information Extraction Management System (EMS) is to support the execution of the many component extraction *operators*—classifiers, sectioners, language analyzers, wrapper generators, etc.—as they operate on a variety of documents. In the examples mentioned above, these documents are snapshots of web pages crawled from the web.

PSOX EMS provides three key benefits: extensibility, explainability and support for social feedback. *Extensibility* means that it should be easy to add a new operator by adapting the input and output signature of an existing operator, and to quickly prototype the use of this operator in an information extraction pipeline with easy perusal of the results. The key enabler of extensibility is a declarative infrastructure for information extraction "operators." As new operators are added by humans, or trained from minimum-supervision transfer-learning techniques, a new facility becomes important: the ability to *explain* extraction results. Explanation allows the user to ask questions about extracted results, much as the questions asked by users of scientific data management systems (e.g. [4]). For example, a user looking at an extracted bibliographic record might ask how the information was produced and what other information was produced in the same way. A key application of such questions is operator debugging: by tracing back the chain of inferences that led to an extracted datum, the extraction designer can more easily and quickly localize problems to the operators or data sources at fault. Finally, PSOX EMS supports the light-weight gathering of *social feedback* on the quality of extracted results, and combining this feedback with the explanation capabilities to develop quality profiles of different opera-

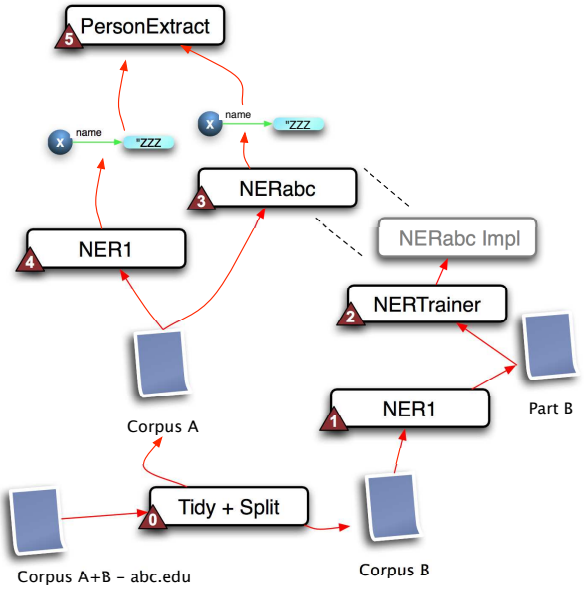

Figure 1: Academic Homepage Extraction

tors and of the feedback itself. This final capability is critical for *large scale* information extraction, as only low-supervision techniques can scale, and feedback *after* extraction will naturally be required as supervision *before* is decreased.

The rest of the paper is organized as follows. In Section 2, we introduce our running example. Section 3 describes the operator model, which supports the extensibility. The provenance model for explainability is discussed in Section 4, while the scoring model for social feedback is described in Section 5. Finally, we discuss related work and conclude in Sections 6 and 7.

2. MOTIVATING EXAMPLE

In this section, we introduce a motivating example of an information extraction pipeline.

EXAMPLE 1. *In Figure 1, an example execution of a hypothetical pipeline for extraction from academic home pages on a site* abc.edu *is shown. First, a set of downloaded web pages, "Corpus A+B" from a fictional academic website,* abc.edu*, is crawled and tidied (to create well-formed* HTML*). The corpus is then split into two sets: a small set "A" for training and set "B" for prediction. Next a "named entity recognition" operator, NER1 is used to label parts of the corpus as entities like "Person Name" or "School Name". Designed to work across different sites, NER1 is tuned for high precision but perhaps low recall. NERtrainer uses the output of NER1 "Mark A" to create the operator NERabc. The idea of NERtrainer is to adapt to features of site* abc.edu *and thus potentially allow NERabc to do better than NER1. However, since it is trained by NER1's output rather than human annotated examples, it may end up amplifying some errors in NER1 and in fact do worse - a situation typical of low-supervision techniques for information extraction. Both operators, NER1 and NERabc are then applied to the rest of the corpus (set B). Finally, a PersonExtractor operator accumulates evidence from the predictions of NER1 and NERabc to identify person entities, educational institutions, and the relevant relationships.*

Figure 2 illustrates part of a data instance in the PSOX data model that might result from the plan execution shown in Figure 1. This data instance is a graph with *score* annotations on the edges. The model is similar to RDF [9], but we use a slightly different terminology - "entity" rather than "resource" and "relationship" instead of "statement". Other features of our system that further distinguish it from most triple stores, such as support for provenance and uncertain information, are discussed below.

EXAMPLE 2. *There are four major entities (E1-E4, circles) in our example. E1 and E2 represent* system entities*: E1 represents a snapshot of a web page, as indicated by its* type *relationship, which targets the atomic entity "WebPageSnapshot". (We use round rectangles to represent atomic entities.) E2 represents the web page itself, and has a* url *that may change over time. Intuitively, E1 and E2 participate in the* snapshot-of *relationship. These entities and relationships would result from the* Tidy *operator in Figure 1.*

On the right of the figure, E3 and E4 represent semantic entities*: E3 is a school while E4 is a person with position "Professor" at that school. While all relationships (arrows) can be mapped into entities and thus be the source or target of other relationships, only two such situations are shown in Figure 2, the* name *relationship (E5) between E4 and the atomic "Lisa S. Martasyou" and the* position *relationship (E6) between E4 and the atomic "Professor". The* mentions *relationships, which connect E1 and E5/E6, illustrate the cases where a relationship itself is involved in another relationship (similar to a reified statement in RDF). Finally, every relationship has a score associated with it, but only two scores are shown:* 0.8 *on the name of the person and* 0.95 *on the name of the school). The topic of how scores are arrived at and what they mean is discussed in detail in Section 5.*

One important point is that the "type" relationship in Figure 2 is treated as a normal relationship in terms of having a score, but also represents the type of the object. Having such a "soft type" is critical in information extraction as types in practice are associated with objects by some form of classifier, and thus may be subject to error just like any other attribute.

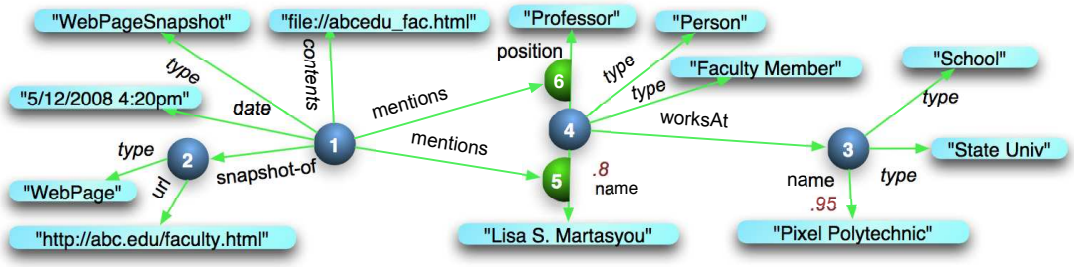

Figure 2: PSOX Data Model Example

3. OPERATORS AND EXTENSIBILITY

The extensibility of PSOX derives from a policy of *declaratively* specifying operators. In this section, we introduce the modeling of extraction operators at logical and physical levels. It is worth noting that operators are themselves represented within the PSOX data model.

Operators correspond to basic units of functionalities (e.g., page classification) and are defined at both a *logical* and *physical* level. At the logical level, an operator specifies the information it needs from the data instance and the information it produces for populating the data instance. As such, it is defined by an *operator specification*, consisting of the input it consumes and the output it produces, where the input is a collection of retrieval queries and the output is a collection of assertion queries. At the physical level, an operator is further defined by the executables, the associated arguments, the actual input and output files locations, and the necessary environment variables comprising the *operator implementation*.

A key assumption of PSOX, the *black box assumption*, is that the system should not need much visibility into *how* an operator works. At the logical level, this implies that scores from operators may not be comparable. At the physical level, this implies that we offer a language-neutral model of operators as independent executables. Compared to other extraction pipelines, this emphasizes flexibility, similar to Cimple [6] more than performance on single documents, as in UIMA [8].

Given the separation between the logical and physical levels, specific operator implementations can be easily swapped into and out of the extraction pipeline without affecting the rest of the system, as long as the logical specification is fixed. Consequently, operators written by third parties can be seamlessly leveraged inside the PSOX system.

3.1 Operator Specification

EXAMPLE 3. *We illustrate in Figure 3 a simple operator specification. Intuitively, it takes input from PSOX as specified by the input relation (i.e. retrieval query) WebPages, generates output in the format as specified by the output relation Faculty, and asserts information back into PSOX according to the output assertion (i.e., assertion query) FacultyAssertion.*

Input Specifications : The first component of an operator specification is the input specification, comprised of a set of named *retrieval queries*. While the results of a retrieval query are the *output* from the PSOX data instance, they serve as the *input* to the operator. Semantically, before the operator executes, a set of files are materialized, each containing the results of one retrieval query in the operator's input spec. Note that for performance, the input query may not always be executed: for example, if an output file of a previous operator is known to be semantically equivalent to the input relation of the next operator. We now describe retrieval queries.

A PSOX retrieval query rq is a relatively straightforward language for querying data graphs. More formally, each query a 4-tuple $(\mathsf{name}, \mathsf{V}, \mathsf{ICols}, \mathsf{CE})$, where name is the name of the query, V is the set of entity or score variables, ICols is the set of variables ($\mathsf{ICols} \in \mathsf{V}$) whose values are to be retrieved from the PSOX data instance, and CE is a constraint expression, which is recursively defined as $\mathsf{CE} = \mathsf{c}|(\mathsf{CE}' \text{ and } \mathsf{CE}'')|(\mathsf{CE}' \text{ or } \mathsf{CE}'')|$ (not CE'), where $\mathsf{c} \in \mathsf{C}$ and CE' and CE'' are themselves constraint expressions. The satisfaction of a constraint expression follows the typical logic rules. For example, a constraint expression ($ce1$ and $ce2$) is satisfied if both $ce1$ and $ce2$ are satisfied. The answer to the query is the set of tuples T. Each $t \in T$ is a set of value assignments to ICol, and there exists a set of value assignments o to variables ($\mathsf{V}' = \mathsf{V} - \mathsf{ICol}$) such that CE is satisfied given t and o.

In our example operator spec, we have the retrieval query, "WebPages", whose SELECT clause contains the three variables in ICols, and whose WHERE clause describes the CE. Intuitively, this operator asks for snapshots of web pages with URLs matching the pattern "faculty".

Output Specifications : The goal of the output specification is similar to the goal of an "ETL" script—it specifies the relationship between an operator's output (and it's input) and new data that should be added to the PSOX data instance as a set of assertions. Note that the "new data" can include new assertions on existing rela-

OPERATOR ExtractFaculty

INPUT RELATION WebPages **AS**
SELECT X, X.url, Y.contentPointer
WHERE X.type = "WebPage" and Y.type = "WebPageSnapshot"
and IsSnapShotOf(X, Y, s2) and X.url *like* "faculty"

OUTPUT RELATION Faculty
(conf, name, nameConf, pos, posConf, page)

OUTPUT ASSERTION FacultyAssertion **AS**
FROM Faculty(c, n, nc, p, pc, g)
WHERE p = "professor"
ON ENTITIES X = f(n,g)
ASSERT type(X, "Faculty Member", c)
and name(X, n, nc) and position(X, p, pc)

Figure 3: Example Operator Spec

tionships, so it may be that no new entities or attribute values are added.

An output specification contains two parts, a set of *output relation specifications* and a set of *assertion queries*. The output relation specifications simply describes the schema of a particular output file produced by the operator. Our example operator produces the relation "Faculty", which contains a list of flat tuples for extracted faculty members with attributes corresponding to: *overall confidence* about the tuple (conf), *name* of the faculty and *confidence about the name* (n, nc), *position* of the faculty and *confidence about the position* (p, pc), and where the information is extracted from (page).

Assertion queries describe how to assert the extracted information from the operator back into the PSOX data instance. They are defined in a similar way to retrieval queries, with the addition of *assertion constraints*, which are 4-tuples corresponding to new relationships being added to the data instance. In our example, the assertion query "FacultyAssertion" asserts $type$, $name$, $position$ relationships for each extracted faculty member with a position "professor".

Basic de-duplication: The variables in the ON ENTITIES clause (e.g., X) guide the creation of new entities, and the optional function following allows "single-operator" deduping. The problem is that pages may include many mentions of the same entity (e.g. bibliography pages), and it may be prohibitively expensive to create dozens or hundreds of entities only to subsequently combine them in a deduping step. In this example, we use "f(n,g)" to indicate that only one new entity X should be created for each unique (name,webpage) pair. A second mechanism allows "key functions" associated with each type. Unlike relational keys that prevent inserts, these functions ensure dedupping across extraction events. (Note that this does not replace entity resolution, and is only used when equality of entities is certain.)

3.2 Operator Implementations

IMPLEMENTATION SVMExtractor
IMPLEMENTS ExtractFaculty

RUNNING Python PROGRAM
EXTERNAL AT "/usr/bin//extractors/svm-faculty.py"
WORKING IN "/data/faculty"

INPUT FILE "pages.txt" AS WebPages
OUTPUT FILE "faculty.txt" AS Faculties

Figure 4: Example Operator Spec

Figure 4 illustrates an example operator implementation of our ExtractFaculty operator. As mentioned before, operator implementation describes the details of how the operator should be invoked. Here, it is a python program that should be executed within the directory "/data/faculty", and that it takes input file "pages.txt" (which corresponds to the input relation WebPages) and produces output file "faculty.txt" (which corresponds to the output relation Faculties).

Operator training: As mentioned before, PSOX maintains operators (both specification and implementations) as part of the data instance. As a result, an operator can assert a new operator into the data instance just like it can assert a new regular relationship. This feature allows the modeling of *training operators*, which can be considered as higher order operators that produce other operators (e.g., a classifier trainer operator can take training data and produce a classifier operator, which can then classify web pages). Often, this simply involves inserting a new operator implementation satisfying an existing operator specification.

4. PROVENANCE MODEL AND EXPLAINABILITY

In this section, we describe the way execution traces are represented in the data model.

4.1 Execution Model

Plan: Composing multiple operators together gives us a PSOX plan. Similar to the operators, plans are defined at both the logical level - as a DAG of operator specifications - and at the physical level - as a DAG of operator implementations. Currently the choice of physical operators for each spec is up to the user, but the architecture is designed to support both *plan generation* (i.e., how to produce a plan with a simple task specification like "find persons and institutions on abc.edu") and *plan optimization* (i.e., how to select the right operator for each step to maximize extraction quality and minimize extraction cost – see, e.g., [10]).

Execution : Extraction results are produced through the execution of an extraction plan, which in turn consists of executions of each individual operators in the plan. Formally, a unique *execution* x of an operator implementation, o, represents a (successful) run of physical plan step s_P where $\mathsf{oplmpl}\ (s_P) = o$. For each

execution, PSOX keeps track of the *operator* responsible for the execution, *time* of the execution, *environment* of the execution. For each execution result (i.e., an entity or relationship being asserted into the data instance), PSOX maintains the entities and relationships that lead to its generation. There are also many open research challenges. For example, *optimizing plan re-execution* (i.e., re-executing a plan that has been executed before) and *asynchronous plan execution* (i.e., separating the assertion of execution results from the plan execution to maximize throughput).

Assertion : In a data model instance I that conforms to an extraction schema, every relationship is either an axiomatic relation generated by the system or is the result of an assertion from at least one *execution event*. We define an *assertion* as a special type of relationship that has as its source an execution event and as target a relationship that is itself not an assertion. In Figure 5, the three blue arrows each indicate an assertion relationship from the operators NER1, NERabc and user Joe to the target relationship 5 (name).

The supporting evidence used by the operator in making the assertion is captured via *basis relationships*, which are defined as relationships with the evidence assertion (generated by the system or other operators) as the source and the inferred assertion as the target. For example, in Figure 5, each of the three red arrows from the system assertion associated with relationship 7 (contents) to the assertions (blue arrows) made by the different operators are examples of a basis relationship, i.e., the page contents support the operators' assertions. The notion of *basis* relationships captures the dependence between the output and input of an operator execution. However, we do not explicitly model the dependence between the output assertions of a single operator execution. The underlying assumption is that *the output assertions are conditionally independent of each other given their respective basis assertions*. This assumption is critical for tractable storage and computation, but could result in loss of information when the assumption is violated, for example in the case of collective prediction tasks.

Mention : The *mentions* relationship is another special type of relationship for supporting provenance and captures the fact that the contents of a text artifact support a relationship. Specifically, a virtual mentions relationship m from e to r_2 is supported by the PSOX query executor whenever there is an assertion a and the basis b of a, such that $\mathsf{src}(b) = e$, $label(b) =' contents''$ and $\mathsf{tgt}(a) = r_2$. Figure 2 shows example of a mentions relationship between 1 and relationship 5 (name).

4.2 Lineage

The existence of *basis* relationships enables us to readily obtain forward or backward lineage for any assertion in the data instance. Let $Asrt$ denote the set of all assertions in the data instance I and $\mathcal{P}(Asrt)$ its power set. We define the functions $Basis : Asrt \mapsto \mathcal{P}(Rel)$ and $Cons : Asrt \mapsto \mathcal{P}(Rel)$ (Consequence) as mappings from an assertion to all the supporting assertions and all the assertions that depend on it respectively, i.e., $\forall r \in Asrt$,

$$Basis(r) = \{r' | \exists (r', r, \text{"}basis\text{"}) \in Rel\}$$

$$Cons(r) = \{r' | \exists (r, r', \text{"}basis\text{"}) \in Rel\} \ \forall r \in Asrt.$$

Given the causal nature of the assertions, the data instance restricted to only "basis" relationships turns out be an directed acyclic graph. Let $Basis^*(r)$ and $Cons^*(r)$ be the transitive closure of $Basis$ and $Cons$ respectively.

In Figure 5, the subgraph restricted to red edges clearly shows the backward lineage of the operator relOp's assertion on the $worksAt$ relationship between 4 and 3, i.e., it depends on the fact that the names of person 4 and organization 3, which in turn were extracted by other NER operators are mentioned in close proximity in web page 1. Similarly, one can also identify all the assertions inferred from the contents of web page 1 by considering the forward lineage of relationship 7 (contents). Maintaining this lineage is essential for credit assignment along the pipeline, identifying erroneous operators and data, and timely reruns of operators to ensure high quality extraction. For example, if a new assertion substantially lowers the score of a relationship r as discussed in the next section, then relationships in $Cons^*(r)$ might need their scores re-evaluated, and operators that input these relationships might need to be re-run.

5. SCORING AND SOCIAL FEEDBACK

In this section, we discuss how scores on assertions and relationships are computed and updated, and outline the process of belief revision in the face of changing scores.

5.1 Assertion and Relationship Scores

Each assertion in the data instance is associated with a score, which can be interpreted as a function of the operator's estimated probability that the target relationship is true given the basis assertions. In case of axiomatic target relationships, there is no uncertainty and the assertion either supports or does not support the relationship. We choose to interpret the score as a function of probability rather than the probability value itself in order to accommodate a wide variety of execution scenarios that are common in a realistic extraction pipeline. A prime example is one where the operator implicitly assumes extra conditions (e.g., training and test data have identical distributions or restriction to a subset of all possible outcomes) so that the scores do not exactly correspond to conditional probability given the basis assertions. Another important scenario involves operators that output

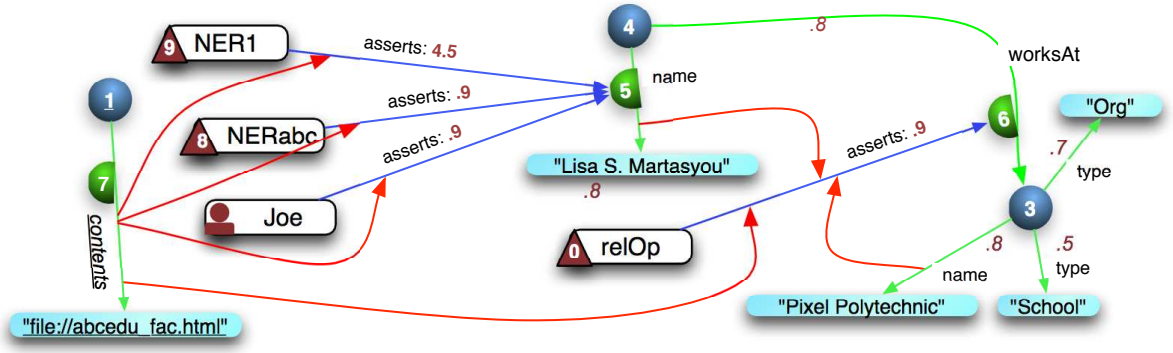

Figure 5: Provenance and Chained Inference Example

scores that cannot be readily interpreted as conditional probabilities over outcomes, e.g., SVM classifiers and margin-based predictors. Thus, the interpretation of the assertion score could vary depending on the operators as well as the nature of the target relation and the associated tasks (e.g., collection of text artifacts, classification and segmentation of text, record assembly, deduping, etc.)

Assertions by the system (or system reconciler to be exact) constitute an important subset of all assertions. In fact, each non-assertion relationship is the target of at least one system generated assertion. Furthermore, the score of a non-assertion relationship r is defined as the score of the most *recent* system assertion associated with r. For this special class of assertions, the scores can be interpreted as the probability that a relationship is true given the schema constraints and various system-specific assumptions.

Figure 5 shows this distinction between the three assertion scores on the blue arrows and the relationship (system) assertion score (0.8) on the green arrow corresponding to the relationship (names,4,"Lisa S. Martasyou") . Operator NER1 might return margins and the assertion score (4.5) cannot be viewed as a probability value, but the system adjusts for abthese variations to assign a probability of 0.8 to the target relationship.

5.2 Scoring and Social Feedback

Scoring each relationship in the data instance is a critical component of an extraction management system and requires taking into account the following key issues:

Varied Operator Behavior. In the real-world, extraction pipelines frequently involve operators with varying bias, scale and confidence levels, and often provide conflicting assertions. For instance, in figure 5, we have two automated operators (NER1, NERabc) and a user Joe providing different scores for the same target relationship (names,4,"Lisa S. Martasyou"). Hence, it is imperative to adjust for these variations in operator assertion scores by monitoring how these correlate with the "true" probability scores.

Social Feedback. Incorporating feedback from non-editorial human users enables one to rapidly obtain large amounts of training data as well as naturally scale up extraction process across various application domains. In PSOX, these human users (e.g. Joe in Figure 5) are modeled as operators with fairly general I/O spec based on their data access and editorial privileges. Compared to automated operators, human users have expertise in a large number of heterogeneous domains (e.g., text classification, segmentation, entity deduping, etc.). Further, the feedback is often incomplete and corresponds to a biased sample. Anonymity on the Internet also creates additional challenges by allowing malicious behavior and collusion among users.

Schema Constraints. Since the relationship scores are conditioned on the specified extraction schema, it is also critical to ensure that there are no violations of the schema constraints pertaining to typing, inheritance, relationship cardinality, mutual exclusion, etc. These constraints in general translate to linear equalities and inequalities over the relationship scores that determine a feasible region. For instance, in figure 5, the probability of entity 3 being a school is less than that of it being an organization, i.e., $Score(type, 3,'' school'') < Score(type, 3, ''organization'')$ since *school* is a subtype of *organization*.

Oracular Assumptions. Calibration of operator and human assertion scores requires making certain "oracular" assumptions about how they correlate to the "true" probabilities of the relationships. Such assumptions could take the form of knowledge of "true" probabilities on limited number of relationships or a functional mapping from the assertion scores to the "true" probabilities for a small set of operators.

Bayesian Solution. To address this problem, we adopt a Bayesian approach that relies on modeling the process of generating the assertion scores as a stochastic transformation of the unknown "true" probabilities of the relationships. The key idea is to use all the available operator assertions, oracular information as well as the schema constraints to estimate the most likely para-

metric model for the operator (user) behavior. The interpretation of the operator specific parameters depends heavily on the nature of assertion scores and the allowed class of transformations. For example, in Figure 5 the parameters could correspond to a linear scaling of the relationship probabilities, for example, $(9,\ 1,\ 0.9)$ for the operators NER1, NERabc and Joe respectively and the final score 0.8 assigned to (names,4,"Lisa S.") is obtained by appropriate adjustment of the assertion scores of these operators, i.e., $0.8 = (1/3) \times (4.5/9 + 0.9/1 + 0.9/0.9)$. In general, the parameters need not be specific to individual operators, but relate to observed characteristics of the operators, such as training dataset, and of the target relationships, for example, gender/profession of a person.

6. RELATED WORK

Our effort is closely related to several extraction management system projects. The first notable one is the Cimple project [6] at Wisconsin. Cimple aims to build an EMS as part of a larger community building platform, with a focus on developing declarative information extraction language and optimization techniques [10] and handling evolving data [3]. PSOX seeks to complement the capabilities of Cimple, adding further support for low-supervision extraction on a wide variety of domains, and with an emphasis on explainability and social feedback. Another closely related project is GATE [5], which provides both an architecture and a framework for natural language engineering. GATE employs a document-centric model where extracted entities are inherently associated with the documents they come from. In contrast, PSOX employs the entity-centric model, where documents and entities are both first class citizens in the model and the entities can be used and reasoned independent of the documents they are coming from. Similar to GATE, the UIMA project [8] at IBM provides an even richer framework for information extraction and its integration into various product platforms (e.g., WebSphere). While UIMA moves one step away from the document-centric model by allowing entities to be maintained independently from documents, it does not emphasize much on the after-extraction processing of those entities. In contrast, support for after-extraction entity processing like entity reconciliation is an integral part of the PSOX platform.

Scientific Data Management [1] is an active research area that is closely related to the PSOX effort regarding the plan execution model, which consists of individual operators. Efficiently and effectively tracking provenance for complex data manipulation systems (e.g., scientific data management systems) has also been receiving steady attention for quite some time [4, 2]. PSOX seeks to apply progress in this area to information extraction.

7. CONCLUSION

We have presented the fundamental architectural and modeling decisions of the Purple SOX (PSOX) Extraction Management System. We emphasized three desired characteristics of PSOX: *extensibility*, which is supported by the flexible operator model; *explainability*, which is accomplished through an extensive provenance model, and *support for social feedback*, which is achieved through an effective scoring model.

Acknowledgments We thank Michael Benedikt for several discussions on the data and scoring model. We thank AnHai Doan, Mani Abrol, Krishna Chitrapura, Keerthi Selvaraj, Minos Garorfalakis, Nilesh Dalvi, Ashwin Machanavajjhala, Arup Choudhury, Prakash Ramanan and Alok Kirpal for helpful discussions on various aspects of the system architecture.

8. REFERENCES

[1] Ilkay Altintas et al. Introduction to scientific workflow management and the Kepler system. In *SC*, 2006.

[2] A. Chapman, H. V. Jagadish, and P. Ramanan. Efficient provenance storage. In *SIGMOD*, 2008.

[3] F. Chen, A. Doan, J. Yang, and R. Ramakrishnan. Efficient information extraction over evolving text data. In *ICDE*, 2008.

[4] S. Cohen, S. Boulakia, and S. Davidson. Towards a model of provenance and user views in scientific workflows. In *DILS*, 2006.

[5] H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. GATE: A framework and graphical development environment for robust NLP tools and applications. In *ACL*, 2002.

[6] P. DeRose, W. Shen, F. Chen, A. Doan, and R. Ramakrishnan. Building structured web community portals: A top-down, compositional, and incremental approach. In *VLDB*, 2007.

[7] A. Doan, R. Ramakrishnan, and S. Vaithyanathan. Managing information extraction: state of the art and research directions. In *SIGMOD*, 2006.

[8] D. Ferrucci and A. Lally. UIMA: an architectural approach to unstructured information processing in the corporate research environment. *Natural Language Engineering*, 10(3-4):327–348, 2004.

[9] F. Manola and E. Miller. RDF Primer W3C Recommendation, 2004.

[10] W. Shen, A. Doan, J. Naughton, and R. Ramakrishnan. Declarative information extraction using datalog with embedded extraction predicates. In *VLDB*, 2007.

[11] J. Turmo, A. Ageno, and N. Català. Adaptive information extraction. *ACM Comput. Surv.*, 38(2):4, 2006.