March 1998

Information Extraction & Object Views

Zoé Lacroix
University of Pennsylvania, lacroix@saul.cis.upenn.edu

University of Pennsylvania Institute for Research in Cognitive Science Technical Report No. IRCS-98-11.

Information Extraction & Object Views

Abstract

Information extraction consists in identifying classes of events and relationships between extracted instances of these classes. In general, extracted data usually fills slots in a template and is stored in tables. We propose to extend the usual approach to the use of an object database. Information extraction tools have a conceptual representation as schema components: *concept classes, meta-concepts* and *attributes*. The user expresses in his query a structure (*target structure*) which corresponds to his understanding of the domain and is used as a schema for the database. We use the object data model whose syntax matches both the user's target structure and the conceptual representation of extracting capabilities. Query evaluation consists in first determining the schema of the database as expressed by the user, and secondly populating the database through methods invoking extraction tools on a given source of documents. In a third step, it returns the output of the query against the resulting database. The two first steps define an *object view* of the given source(s) as a materialized extension of the current schema (each refinement of a query may add more structure, and thus more extracted data) followed by a non-materialized projection.

Our approach is user-oriented: the object representation of data provides the user with the flexibility of asking his query with his understanding of the domain, and object views are built on-the-fly according to the user's organization of data. The modularity of the conceptual representation of extraction capabilities in a pool of schema components enables easy plug-in of new extracting tools.

Keywords

Information extraction, object data model, object view.

Comments

University of Pennsylvania Institute for Research in Cognitive Science Technical Report No. IRCS-98-11.

Information Extraction & Object Views

Zoé Lacroix*
IRCS, University of Pennsylvania
Suite 400A, 3401 Walnut Street, Philadelphia PA 19104, USA
Phone: 1 (215) 898 0326 – Fax: 1 (215) 573 9247
lacroix@saul.cis.upenn.edu

Abstract

Information extraction consists in identifying classes of events and relationships between extracted instances of these classes. In general, extracted data usually fills slots in a template and is stored in tables. We propose to extend the usual approach to the use of an object database. Information extraction tools have a conceptual representation as schema components: *concept classes*, *meta-concepts* and *attributes*. The user expresses in his query a structure (*target structure*) which corresponds to his understanding of the domain and is used as a schema for the database. We use the object data model whose syntax matches both the user's target structure and the conceptual representation of extracting capabilities. Query evaluation consists in first determining the schema of the database as expressed by the user, and secondly populating the database through methods invoking extraction tools on a given source of documents. In a third step, it returns the output of the query against the resulting database. The two first steps define an *object view* of the given source(s) as a materialized extension of the current schema (each refinement of a query may add more structure, and thus more extracted data) followed by a non-materialized projection.

Our approach is user-oriented: the object representation of data provides the user with the flexibility of asking his query with his understanding of the domain, and object views are built on-the-fly according to the user's organization of data. The modularity of the conceptual representation of extraction capabilities in a pool of schema components enables easy plug-in of new extracting tools.

`Keywords:` Information extraction, object data model, object view.

1 Introduction and motivating example

Information Extraction (IE) tasks involve processing a linear textual document to fill slots in a template. We believe that IE systems may take advantage of database technology. Storing extracted data in a database first provides the user with a query language which aims at to manipulate data in a transparent way. Furthermore, query evaluation may be optimized (query rewriting, indexing, etc.). One may also think of using some probabilistic models to deal with uncertainty, or apply data-mining techniques. In general, IE systems use a relational database composed of tables with the template slots as attributes. In this paper, we present the benefits a system would take from the use of object-oriented database technology, in particular of an object view mechanism, to manipulate data extracted from documents. The paper is illustrated by an example introduced below.

Researchers are usually interested in Calls for Papers (CFPs) in subject areas related to their research. Calls for Papers are usually text documents with some minimal implicit structure. Typically, they provide information about: the name, the location of the conference/workshop (say event), the PC chair, the list of topics of interest, the last date for submissions, etc. Suppose that we have a source consisting of textual CFPs. We illustrate the advantages of our approach with several queries a user might ask against the source. The first queries **Q0**, **Q1** and **Q2** illustrate, if necessary, the improvements provided by IE as opposed to Information Retrieval (IR) to users.

*Work supported by NSF STC grant SBR-8920230.

- **Q0:** *"CFPs of conferences with a submission deadline in 1998 after March 1, 1998?"*

  Traditional IR tools do not extract any information and the user can only express boolean queries with keywords such as: `("March" OR "May" OR ... OR "December") AND "1998"` which cannot distinguish the date of submission from the date of the conference (or any other date mentioned in the CFP). Other approaches, such as RIPPER [Coh96] which provides automatic classification of textual documents based on "keyword-spotting" rules, face the same limits. To properly answer query **Q0**, the relationship `submission_deadline` between a conference and a date has to be extracted. *SuperTagging* [JS94], which provides rich syntactic labels, may be used with other tools (for example a co-reference tool [BDR+97]) to capture a variety of syntactic and semantic information, and thus meta-concepts such as `submission_deadline`. Accessing the structure of information contained in documents, information extraction goes beyond information retrieval.

- **Q1:** *"Conferences with a submission deadline in 1998 after March 1, 1998?"*

- **Q2:** *"Countries where are hosted the retrieved conferences?"*

  The user expects to be returned the names of conferences when asking **Q1** and a set of countries when asking **Q2**, and nothing else. Contrary to query **Q0**, the IE step is not used to filter CFPs out of several CFPs and thus improve IR tools, here, **Q1** and **Q2** must return information extracted from the documents of the source.

To answer queries **Q3** and **Q4**, not only data extraction but data manipulation is necessary. Furthermore, as illustrated with queries **Q5** and **Q6**, a database representation of extracted information enables the user to refine his query.

- **Q3:** *"How many conferences are located in France?"*

  If the extracted data is in a database, query **Q3** only consists in counting the number of objects returned by the query *"What are the conferences located in France?"*

- **Q4:** *"Which conferences have a PC chair who was a SIGMOD PC member last year?"*

  To answer **Q4**, one has to (1) extract the PC members of the conference SIGMOD of 1997, (2) extract the PC chair of all conferences of 1998, and (3) select the conferences of 1998 such that the PC chair is in the list of SIGMOD PC members. With no surprise, a database representation of extracted data facilitates the third step.

- **Q5:** *"Conferences with a submission deadline after May 1, 1998?"*

  Suppose that the user first asks query **Q1**, and query **Q5** as a refinement to **Q1**. If the data extracted to answer query **Q1** is represented in a database such that to each conference is associated a submission deadline, then **Q5** is no more than a `select ... from ... where ...` query against the base.

- **Q6:** *"Conferences with a submission deadline later than May 1, 1998 and located in France?"*

  Suppose that the user asks **Q6** as a refinement of queries **Q3** and **Q5**. The evaluation of **Q6** should only consist in an intersection of the outputs of queries **Q3** and **Q5**.

- **Q7:** *"Events with a submission deadline after May 1, 1998?"*

  Query **Q7** expresses a subsumption (generalization) of query **Q1**.

When combined with a database, traditional IE tools usually store their extracted information in tables [Paz97]. If the relational model is as expressive as the object one, it fails in offering the user a representation of data that corresponds to his knowledge and his understanding of the information he is looking for. In particular, a user has a *conceptual subsumption* understanding which matches the object representation of data. Our approach supposes available[1] IE tools which identify and extract *concepts*, *meta-concepts* and *attributes* from the source(s). The extraction capabilities are represented as a *conceptual schema* (presented

[1] IE itself is a difficult task which we do not address in the paper.

in Section 2) which corresponds to the user's understanding as well as the structure of the database where to store extracted information.

The object database where is stored extracted data is an *object view* of the given source(s) of documents. It is built on-the-fly according to the user's needs as explained in Section 3. As defined in Section 4, our view mechanism goes through a pipeline of materialized views, obtained by successive materialized *extensions* and non-materialized *projections* [dSDA94, LDB97].

2 Conceptualization of extraction capabilities

We consider three extraction capabilities: extracting *concepts*, *meta-concepts* and *attributes* which we represent in an abstract way in a *pool of schema components*.

2.1 Extraction capabilities

Extracting a *concept* goes beyond the usual retrieval of keywords, even though the extracting technique may use a list of keywords. We suppose that a *concept* is extracted when the following steps are accomplished: (1) recognition, and (2) identification (canonical representation). The *recognition phase* consists in selecting fragments of documents which satisfy a certain criteria. For instance, in a textual document, strings of characters which are dates such as *May 1, 1998* or *05/01/98*, may be selected as fragments. Each fragment is associated with a pair consisting of a file address and a span. The second step, consists in identifying fragments according to a *canonical representation*. The canonical representation of a date may be a list of three integers respectively representing a month, a day and a year and the two previous examples of fragments are identified with *[5,1,1998]*. It follows that an instance of a concept may correspond to several different fragments extracted from different documents. Extracting conferences also consists in extracting a concept. From the retrieved documents, all strings of characters mentioning a name of conference (such as *International Conference on Conceptual Modeling*) or acronyms (such as *ER'98*) must be first selected and then canonically represented. The acronym can be chosen as the canonical representation for a conference (a list of conference names and corresponding acronyms, gathered in a training phase may be used). To each extractable concept corresponds a *concept class*.

The concepts may be organized in a hierarchy. For instance, `Conference` and `Workshop` could be *sub-concepts* of concept class `Event`. This hierarchy can be accessible at the level of IE tools as *conceptual subsumption* rules such as presented in [Woo97]. In this paper, we only consider the extraction of *sub-concept*, which would express a *is-a* relationship between concepts. We suppose that the hierarchy is a forest (that is a concept class can be a subclass of only one concept class).

A *meta-concept* is a relationship between concepts. For example, `submission_deadline` is a meta-concept between concepts `Conference` and `Date`. Extracting meta-concepts consists in associating one instance of a concept class to another instance of a concept class. For instance, the instance *ER98* of class `Conference` will be associated with the instance *[4,3,1998]* of class `Date`. Each extractable meta-concept is conceptualized as an attribute of concept range. For instance, `submission_deadline` corresponds to an attribute of signature `Conference` $\rightarrow$ `Date`.

Extracting an *attribute* consists in extracting a set of fragments associated with a concept. For example, `topic` associates to each conference a string of characters, extracted from its CFP, listing the topics of interest. Each extractable attribute corresponds to an attribute of value range (such as string, integer, etc.). For instance `topic` is conceptualized as an attribute of signature `Conference` $\rightarrow \{string\}$.

An instance of a concept class represents several extracted fragments which refer to the same entity. This representation is compatible with the use of *co-reference* techniques. It is worth noting that the abstract representation of extraction capabilities is medium independent, even though the source of our motivating example is textual. Multimedia or hypermedia extraction capabilities may also be represented as concepts, meta-concepts and attributes.

2.2 Pool of schema components

The *pool of schema components* is the conceptual representation of the extraction capabilities of the system. It consists of *concept classes* organized in a *hierarchy*, and *attribute names* and *definitions*. An *attribute signature* is $c \rightarrow \gamma(c')$, where c is a concept class, c' may be either a concept class or a value class such as *string, integer*, etc., and $\gamma(c')$ is either $\{c'\}$ or c'. An *attribute definition* is a pair $(a,\ c \rightarrow \gamma(c'))$, where a is an attribute name and $c \rightarrow \gamma(c')$ an attribute signature. A attribute a is *definable at* c if there exists a definition $(a,\ c \rightarrow \gamma(c'))$. The set of all attributes definable at c is denoted by $A(c)$. A *concept class* is a 4-tuple $(c, K(c), D_K(c), S(c))$, where c is the name of the concept class, $K(c)$ the set of attribute names corresponding to its canonical representation, $D_K(c)$ the set of their definitions and $S(c)$ the name of its super-class in the subsumption hierarchy (if there is no such a class then $S(c) = \emptyset$). In a pool of schema components, an attribute may be associated with different signatures. For instance, if there is a concept class `Workshop` in the pool, then attribute `submission_deadline` may have two different signatures: `Conference` $\rightarrow$ `Date` and `Workshop` $\rightarrow$ `Date`. The definition of attribute `submission_deadline` at class `Conference` is (`submission_deadline`, `Conference` $\rightarrow$ `Date`). A *conceptual schema* is a 4-tuple $(C, A, \Delta, \prec)$ where C is the set of concept class names, A, the set of attribute names, Δ, the set of attribute definitions, and $\prec$, the *sub-classing relationship*. Each concept class $(c, K(c), D_K(c), S(c))$ defines a basic *conceptual schema* by itself, where $C = \{c\}$, $A = K(c)$, $\Delta = D_K(c)$ and $\prec = \emptyset$.

The operator *combination* permits to define more complex conceptual schemas with schema components. To an input consisting of a conceptual schema $S = (C, A, \Delta, \prec)$ such that $c \in C$, and an attribute definition $(a,\ c \rightarrow \gamma(c'))$, *combination* returns a conceptual schema $S' = (C', A', \Delta', \prec')$ such that $C' = C \cup \{c'\}$ (if c' is a concept class), $A' = A \cup \{a\} \cup K(c')$, $\Delta' = \Delta \cup \{(a, c \rightarrow \gamma(c'))\} \cup D_K(c')$ and $\prec' = \prec \cup \{(c', S(c')) \mid \text{if } S(c') \in C\} \cup \{(c'', c') \mid \text{if } c'' \in C \text{ and } S(c'') = c'\}$. Given a pool of schema components, the set of available conceptual schemas is the set obtained by applying the combination operator to concept classes and attribute definitions.

2.3 Example

Suppose that the pool of schema components available is the following.

Concept classes / Defined attributes / Super-class
(`Conference`, {`name`, `submission_deadline`, `location`, `topic`}, `Event`)
(`Workshop`, {`name`, `submission_deadline`, `joint_with`}, `Event`)
(`Event`, {`name`, `submission_deadline`}, $\emptyset$)
(`Date`, {`month`, `day`, `year`}, $\emptyset$)
(`Location`, {`city`, `state`, `country`}, $\emptyset$)

Attribute definitions
(`name`, `Conference` $\rightarrow$ *string*)
(`name`, `Worshop` $\rightarrow$ *string*)
(`name`, `Event` $\rightarrow$ *string*)
(`submission_deadline`, `Conference` $\rightarrow$ `Date`)
(`submission_deadline`, `Worshop` $\rightarrow$ `Date`)
(`submission_deadline`, `Event` $\rightarrow$ `Date`)
(`location`, `Conference` $\rightarrow$ `Location`)
(`topic`, `Conference` $\rightarrow$ {*string*})
(`joint_with`, `Workshop` $\rightarrow$ `Conference`)
(`month`, `Date` $\rightarrow$ *integer*)
(`day`, `Date` $\rightarrow$ *integer*)
(`year`, `Date` $\rightarrow$ *integer*)
(`city`, `Location` $\rightarrow$ *string*)
(`state`, `Location` $\rightarrow$ *string*)
(`country`, `Location` $\rightarrow$ *string*)

The conceptualization of extraction capabilities provides the user with a flexible way to express his query, as described in Section 3.

3 From the user's query to an object view

3.1 Query language

The use of a database representation enables the user to express his queries in a database query language. The Object Query Language (OQL) [Ba97] provided by the object data model enables the user to express a query by a `select ... from ... where ...` expression with path-expression along attributes. Some of the queries given in Section 1 are expressible in OQL by the following expressions.

Q1: *"Conferences with a submission deadline in 1998 after March 1, 1998?"*

```
select c.name
from   c in Conference
where  c.submission_deadline.month > 3 and
       c.submission_deadline.year = 1998
```

Q2: *"Countries where are hosted the retrieved conferences?"*

```
select c.location.country
from   c in Conference
```

It is worth noting that the conceptual representation of extraction capabilities is compatible with a Natural Language (NL) user interface such as described in [ART95]. In pattern-matching systems, a relational table is assumed and natural language patterns are associated with action rules. Similarly, with our approach action rules can correspond to concept classes and attributes [LSC98a], as illustrated below.

```
pattern: ... "conference name" ...
action:  select c.name from c in Conference
```

3.2 From the user's query to a conceptual schema

When asking a query, a user has a *conceptual representation* in mind expressed in his query. This conceptual representation is a conceptual schema. The conceptual schema associated with a query (called *target structure* in [LSC98a]) is the smallest schema obtained from the pool of schema components by combination. Suppose that the user first asks query **Q1**. The corresponding *conceptual schema* given in Figure 1 is the combination of class `Conference` with attribute `submission_deadline` of signature `Conference` $\rightarrow$ `Date`.

Figure 1: Conceptual schema inferred from query **Q1**

The conceptual schema is inferred from a query as follows:

1. C is the set of classes explicitly expressed in the query, $A = \cup_{c \in C} K(c)$, $\Delta = \cup_{c \in C} D_K(c)$ and $\prec = \{(c, S(c)) \mid \text{if } c \in C \text{ and } S(c) \in C\}$;
2. for each attribute of definition $(a, c \rightarrow \gamma(c'))$ mentioned in the query, then $C = C \cup \{c'\}$, $A = A \cup \{a\} \cup_{c'} K(c')$, $\Delta = \Delta \cup \{(a, c \rightarrow \gamma(c'))\} \cup_{c'} D_K(c')$ and $\prec = \prec \cup \{(c', S(c')) \mid \text{if } S(c') \in C\} \cup \{(c'', c') | \text{if } c'' \in C \text{ and } S(c'') = c'\}$.

The conceptual schema can be used as a template for the representation of the output: it is a perfect candidate for a schema of our object database.

3.3 From a conceptual schema to an object schema

We assume the reader is familiar with the concepts of object-oriented databases as presented in [AHV95]. We adopt the formalism of the data model presented in [LDB97]. Our approach is user-oriented: it does not pre-suppose any schema already defined in the object database. The modularity of IE tools thus corresponds to the modularity of an approach building an object schema on-the-fly with respect to the user's conceptual representation expressed in his query. The resulting database is an *object view* of the source(s).

In our approach, when asking a query a user expresses a *conceptual schema* which corresponds to the structure that IE tools have to extract from the documents and that is used as an *object schema* for the database populated by the extracted data. The correspondence is straightforward: a conceptual schema $S = (C, A, \Delta, \prec)$ is translated in an object schema $S = (C, A, \Delta, \prec)$. Each concept class c corresponds to an object class c of *key attributes* $K(c)$.

All classes possess an attribute `fragment` of type `set of Fragment` (the type `Fragment` is a pair of *string* denoting the file address and the span) which, for each instance of the class, points to the set of all extracted fragments identified by this object. Each class also has an `init` method which invokes IE tools to populate the class.

The object schema corresponding to the conceptual schema given in Figure 1 is defined in Figure 2 (we omit to precise the `init` method).

```
class Conference                              class Date
      type tuple (fragment: {Fragment},             type tuple (fragment: {Fragment},
                  name: string,                                 month: integer,
                  submission_deadline: Date)                    day: integer,
                                                                year: integer)
```

Figure 2: Object schema corresponding to query **Q1**

The notion of *identity* provided by the object data model expresses the notion of *identification* as well as *co-reference* extracted by IE tools. The representation of extracted data in an object database thus enables updates of the database with data newly extracted from the source(s). In Section 4, we illustrate the view mechanism consisting of successive refinements of query and thus successive processings of the sources.

4 View mechanism

Our view mechanism goes through a pipeline of successive and interleaved materialized views (obtained by successive materialized *extensions* and non-materialiazed *projections* [dSDA94, LDB97]). Initially, the database view is empty: no schema is defined, thus no base is defined either. Let S_0 and I_0 be respectively the empty object schema and the empty instance. The evaluation of each query defines the schema of the database, populates it, and returns the answer.

Since initially the current view is empty, the *view definition* v_1 which defines the schema of the database, simply translates the conceptual schema inferred from **Q1** into an object schema. The resulting object schema of the object view is $S_1 = v_1(S_0)$ and defined in Figure 2. Suppose now that the user refines his query with **Q2**. The evaluation of **Q2** as a refinement of **Q1** (which only means that the user did not explicitly required the cache to be emptied) consists in extending the current schema S_1 with a view definition v_2. The extended schema S_2 must contain class `Location` and a new attribute `location` from class `Conference` to class `Location`.

As presented in [AK89, dS95, LDB97], we consider an object view to be the result of two successive steps: an *extension* and a *projection*. Intuitively, an *extension* consists in defining a new schema extending the current schema with new classes and attributes, when a *projection* consists in hiding classes and attributes. A *view definition* is a succession of *extension* and *projection* expressions. Contrary to [LDB97], where object views are not materialized, here the extension step is materialized when the projection itself is not materialized. The current schema of the database results from several materialized extensions, when the

schema viewed by the user is the one expressed in his query and thus results from a virtual projection of the current schema.

We adopt a simplification of the data model presented in [LDB97]. The view mechanism we describe in this paper is restricted: multiple inheritance is not allowed. Thus, the only extension operators used to build the views are the *Specialization*, a restricted Join-Specialization as defined in [LDB97], and *Generalization*.

4.1 Data Model

Traditionally, in object models such as [AHV95], objects are associated with only one class, the class where they are defined, have a unique identifier and their attributes are valued. An object is commonly a pair *(oid, val)*, where *oid* is an object identifier and *val* its value (the value of all its attributes). Our view mechanism aims to represent migration of objects to specialized classes. The migration follows two steps: a extension when a new class c' specializing c is created and populated with the same objects as in class c and a projection when c is removed in the schema and thus its population from the base. In order to express the ability of objects to belong to several classes (class c and its specialization c') and to easily extend all attributes of range c to attributes of range c', we adopt the concept of *referent* introduced in [LDB97]. A referent is a pair $\langle o, c\rangle$, where o is an object identifier and c a class name. The restricted use of the referent data model in our view mechanism lighten the definition of *referents* introduced in [LDB97]. Let $\mathbf{O} = \{o_1, o_2, ...\}$ be the set of object identifiers, and $\mathbf{C}$ the set of class names. The set $\mathbf{R}$ of referents is the set of $\langle o, c\rangle$, where $o \in \mathbf{O}$ and $c \in \mathbf{C}$.

In the Referent Model, classes are populated with referents. Let $(C, A, \Delta, \prec)$ be a schema. $\prec$ denotes the sub-classing relationship, when $\prec^*$ is its transitive closure. A *referent assignment* π associates to each class name in C a finite set of referents such that for each $o \in \mathbf{O}$ and $c, c' \in C$, if $\langle o, c'\rangle \in \pi(c)$ then $c' = c$, that is two populations associated to two different classes are disjoint. In addition, subclass extensions are subsumed by superclass extensions: for all $c, c' \in C$, if $c \prec^* c'$ then for all $\langle o, c\rangle \in \pi(c)$, $\langle o, c'\rangle \in \pi(c')$.

A total function is assigned to each attribute name, when the value of attribute a of signature $c \to c'$ (resp. $c \to \{c'\}$) is not known on object o, then $\langle o, c\rangle.a = nil$ (resp. $\langle o, c\rangle.a = \{\}$) We use a *path expression* $\langle o, c\rangle.a$ to denote the value of attribute a of definition $(a, c \to \gamma(c))$ of the object identified by o in class c. In addition, μ must satisfy the following condition which insures the consistency of attribute assignment with overloaded attributes on an inheritance path. If $d = (a, c_1 \to \gamma(c'_1))$ and $d' = (a, c_2 \to \gamma(c'_2))$, and $c_1 \prec^* c_2$ then for all $o \in \hat{\mathbf{O}}$, if $\langle o, c_1\rangle \in \pi(c_1)$ then $\langle o, c_1\rangle.a = \langle o, c_2\rangle.a$. An *instance* I of a schema $S = (A, C, \prec, \Delta)$ is defined by a referent and a function assignment.

4.2 Extension

Each new class c' of schema S' extending schema S is either a class directly subclass of root or the *specialization*, and thus a subclass, of a class c of schema S, or the *generalization*, and thus superclass, of several classes of schema S. The expression $c' \Leftarrow \text{Specialization}_P(c)$, where P is a set of definitions of attributes at c', expresses that c' is a new class deriving from c, subclass of c (and thus inheriting all attributes defined at c), and specialized by all attributes defined in P. The extension of the schema S with new classes may then also extend the set A of attribute names and the set Δ of attribute definitions. When a new class c' is a creation, then it is defined with the expression $c' \Leftarrow \text{Specialization}_P()$. A_P is the set of attribute names mentioned in the definitions of P. The evaluation of the Specialization expression has the following consequence on the schema.

	$c' \Leftarrow \text{Specialization}_P(c)$	$c \Leftarrow \text{Specialization}_P(c)$	$c' \Leftarrow \text{Specialization}_P()$
C'	$C' = C \cup \{c'\}$	$C' = C$	$C' = C \cup \{c'\}$
A'	$A' = A \cup A(c')$	$A' = A$	$A' = A \cup A(c')$
Δ'	$\Delta' = \Delta \cup D_A(c')$	$\Delta' = \Delta$	$\Delta' = \Delta \cup D_A(c')$
$\prec'$	$c' \prec c$	$\prec' = \prec$	$\prec' = \prec$

The impact of a specialization on the instance of the schema is the following. For each specialization c' of a class c, $\pi'(c') = \{\langle o, c'\rangle \mid \langle o, c\rangle \in \pi(c)\}$. When c' is a created class, then it is populated with referents of the

form $\langle o', c' \rangle$ where o' is a new oid. In all cases, only attributes mentioned in $P \cup K(c')$ are assigned with a method init invoking extraction tools.

The expression $c' \Leftarrow \text{Generalization}_P(c_1, \ldots, c_n)$, where P is a set of attribute definitions at c', expresses that c' is a common super-class of c_1, ..., and c_n. The semantics of the Generalization is the following.

	$c' \Leftarrow \text{Generalization}_P(c_1, \ldots, c_n)$, for $n \geq 1$
C'	$C' = C \cup \{c'\}$
A'	$A' = A \cup A(c')$
Δ'	$\Delta' = \Delta \cup D_A(c')$
$\prec'$	$c_i \prec c$, for $i \in [1, n]$

For each generalization, $\pi'(c') = \{\langle o, c' \rangle \mid \exists i \ \langle o, c_i \rangle \in \pi(c_i)\}$. As for the specialization, only attributes in $P \cup K(c')$ are assigned with extracted information.

4.3 Projection

Intuitively, the projection step consists in hiding classes. A schema S' is the result of the projection of schema S. A projection expression is of the form Projection(c), where $c \in C$. All definitions of the form $(a, c \rightarrow \gamma(c''))$ or $(a, c'' \rightarrow \gamma(c))$ are (virtually) removed from Δ. The semantics of the expression Projection(c) is the following.

	Projection(c)
C'	$C' = C - \{c\}$
A'	$A' = A$
Δ'	$\Delta' = \Delta - [\{(a, c'' \rightarrow \gamma(c)) \mid (a, c'' \rightarrow \gamma(c)) \in \Delta\}$ $\cup \{(a, c \rightarrow \gamma(c'')) \mid (a, c \rightarrow \gamma(c'')) \in \Delta\}]$
$\prec'$	if $c' \prec c$ and $c \prec c''$ then $c' \prec' c''$

The values of each attribute of removed definition $(a, c \rightarrow \gamma(c''))$ are assigned to $(a, c' \rightarrow \gamma(c''))$ for each $c' \prec c$ the following way. For each $\langle o, c' \rangle \in \pi(c')$ such that $\langle o, c \rangle \in \pi(c)$, then $\langle o, c' \rangle.a = \langle o, c \rangle.a$.

4.4 Inferring a view definition from the user's query

The evaluation of query $\mathbf{Q}_{i+1}$ as a refinement of query $\mathbf{Q}_i$ is done in three steps.

1. determination of S_t, the schema inferred by $\mathbf{Q}_{i+1}$;
2. extension expression e_{i+1} to define the new current schema S_{i+1};
3. projection expression p_{i+1} such that $S_t = p_{i+1}(S_{i+1}) = p_{i+1}(e_{i+1}(S_i))$.

The first step characterizes the virtual schema corresponding to the user's view and understanding when asking a query. The second step characterizes the new current schema S_{i+1} as a materialized extension of S_i. The third step consists in expressing the virtual projection to apply to S_{i+1} in order to obtain S_t.

4.4.1 Evaluation of Q1

The schema S_t inferred by **Q1** is given in Figure 1 when the current schema S_0 is empty. The extension step only consists in defining S_1 and the no projection is necessary. The view definition is the extension expression e_1 defined by:

$$e_1 = \begin{array}{l} \texttt{Conference} \Leftarrow \text{Specialization}_{\{(\texttt{name},\texttt{Conference}\rightarrow string),(\texttt{submission\_deadline},\texttt{Conference}\rightarrow Date)\}}\ (), \\ \texttt{Date} \Leftarrow \text{Specialization}_{\{(\texttt{month},\texttt{Date}\rightarrow integer),(\texttt{day},\texttt{Date}\rightarrow integer),(\texttt{year},\texttt{Date}\rightarrow integer)\}}\ () \end{array}$$

resulting from the extension algorithm.

1. $e_i := id$
2. for each class c in C_t which is not in C_i, such that $S(c) = \emptyset$ or $S(c) = c' \notin C_i$, let P be the set of $\{d = (a, c \rightarrow \gamma(c'')) \mid d \in \Delta_t\} \cup K(c)$, and $e_i := c \Leftarrow \text{Specialization}_P()$, e_i;
3. for each class c in $C_t \cap C_i$, let P be the set of $\{d = (a, c \rightarrow \gamma(c'')) \mid d \in \Delta_t - \Delta_i\}$, and $e_i := c \Leftarrow \text{Specialization}_P(\text{c})$, e_i;
4. for each class c in C_t which is not in C_i, such that $S(c) = c' \in C_i$, let P be the set of $\{d = (a, c \rightarrow \gamma(c'')) \mid d \in \Delta_t\} \cup K(c)$, and $e_i := c \Leftarrow \text{Specialization}_P(\text{c'})$, e_i.

4.4.2 Evaluation of Q2

The schema S_t inferred by **Q2** results from the combination of class `Conference` with attribute `location` of signature `Conference` $\rightarrow$ `Location`, as illustrated in Figure 3. S_t is defined by C_t={`Conference`, `Location`}, A_t={`name`, `city`, `state`, `country`, `location`} and $\Delta_t = D_K(\texttt{Conference}) \cup D_K(\texttt{Location}) \cup \{(\texttt{location}, \texttt{Conference} \rightarrow \texttt{Location})\}$. The extension expression e_2 is defined by:

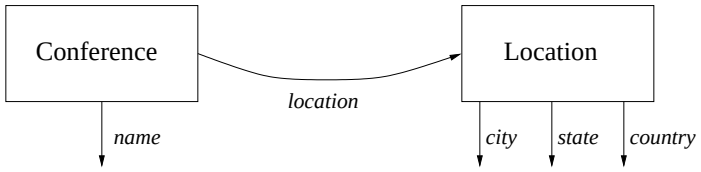

Figure 3: Conceptual schema inferred from query **Q2**

$$e_2 = \texttt{Conference} \Leftarrow \text{Specialization}_{\{(\texttt{location},\texttt{Conference}\rightarrow\texttt{Location})\}}(\texttt{Conference}),$$
$$\texttt{Location} \Leftarrow \text{Specialization}_{\{(\texttt{city},\texttt{Location}\rightarrow string),(\texttt{state},\texttt{Location}\rightarrow string),(\texttt{country},\texttt{Location}\rightarrow string)\}}()$$

The projection step consists in hiding class `Date` and is expressed by $p_2 = \text{Projection}(\texttt{Date})$. It follows that the view is defined by $v_2(S_1) = p_2(e_2(S_1))$.

5 Conclusion

In this paper, we have described an alternative approach to combine information extraction with database technology to provide a flexible and user-oriented interface to query raw documents. The choice a on object-oriented database presents several advantages. The object data nodel provides a notion of identity which naturally represents the essential notion of *co-reference* in IE tools. Its syntax matches the conceptual representation of extraction capabilities as well as the user's representation and understanding. The evaluation of a query expressed in an OQL-like language defines the object database (schema and population) and returns the output against it. The resulting database is object view of source(s) that can be seen as a structured cache built on demand. Our approach allows integration of heterogeneous data that may be local or external (and thus has to be retrieved) in a flexible and transparent way.

The view mechanism described in the paper is one of the component of the AKIRA (Agentive Knowledge-based Information Retrieval Architecture) system [LSC98b], currently under development at the Institute for Research in Cognitive Science in collaboration with the Database group of the University of Pennsylvania.

References

[AHV95] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases.* Addison–Wesley, 1995.

[AK89] S. Abiteboul and P. Kanellakis. Object Identity As A Query Language Primitive. In *ACM SIGMOD Symposium on the management of Data*, pages 159–173, Portland Oregon USA, June 1989.

[ART95] I. Androutsopoulos, G.D. Ritchie, and P. Thanisch. Natural language interfaces to databases - an introduction. *Journal of Natural Language Engineering*, 1(1):29–81, 1995. Cambridge University Press. http://www.mri.mq.edu.au/ ion/nldbsurv.ps.gz.

[Ba97] D. Bartels and al. *The Object Database Standard: ODMG 2.0*. Morgan Kaufmann, San Francisco, 1997.

[BDR+97] B. Baldwin, C. Doran, J.C. Reynar, B. Srinivas, M. Niv, and M. Wasson. EAGLE: An Extensible Architecture for General Linguistic Engineering. In *In Proceedings of RIAO'97*, Montreal, June 1997.

[Coh96] W. Cohen. Learning rules that classify e-mail. In *AAAI Spring Symposium on Machine Learning in Information Access*, 1996.

[dS95] C. Souza dos Santos. *Un Mécanisme de Vues pour les systèmes de Gestion de Bases de Données Objet*. PhD thesis, Université de Paris Sud - Centre d'Orsay, Paris, France, November 1995.

[dSDA94] C. Souza dos Santos, C. Delobel, and S. Abiteboul. Virtual Schemas and Bases. In *Proceedings of the International Conference on Extending Database Technology*, March 1994.

[JS94] A.K. Joshi and B. Srinivas. Disambiguation of Super Parts of Speech (or Supertags): Almost Parsing. In *Proceedings of the* 17^{th} *International Conference on Computational Linguistics (COLING '94)*, Kyoto, Japan, August 1994.

[LDB97] Z. Lacroix, C. Delobel, and Ph. Brèche. Object Views and Database Restructuring. In *Proc. of Intl. Workshop on Database Programming Languages*, August 1997.

[LSC98a] Z. Lacroix, A. Sahuguet, and R. Chandrasekar. Information Extraction & Database techniques: a user-oriented approach to querying the Web. In 10^{th} *Conference on Advanced Information Systems Engineering (CAiSE'98)*, Pisa, Italy, June 1998.

[LSC98b] Z. Lacroix, A. Sahuguet, and R. Chandrasekar. User-oriented smart-cache for the Web: What You Seek is What You Get! In *ACM SIGMOD – Research prototype demonstration*, Seattle, Washington, USA, June 1998. http://www.cis.upenn.edu/∼AKIRA.

[Paz97] M. Pazienza, editor. *Information Extraction*, volume 1299 of *Lecture Notes in Artificial Intelligence (LNAI)*. Springer, 1997.

[Woo97] W.A. Woods. Conceptual indexing: A better way to organize knowledge. Technical Report TR-97-61, Sun Microsystems Laboratories, April 1997.