Ontological Document Reading

An Experience Report

David W. Embley*[,a], Stephen W. Liddle[b], Deryle W. Lonsdale[b], Scott N. Woodfield[b]

[a] FamilySearch International, Lehi, Utah, USA
[b] Brigham Young University, Provo, Utah, USA

Abstract. *Ontological document reading is defined as automatically and appropriately populating a conceptual model representing an ontological conceptualization of some fragment of the real world. Appropriately populating the conceptualization involves not only extracting the information with respect to the declared object and relationship sets of the conceptual model but also involves checking the extracted information for real-world constraint violations, standardizing the data, and inferring the unwritten information that a document author intended convey. Appropriately populating an ontology may, in addition, require adjustments to the ontology itself. This approach to document reading is presented in terms of an effort to build a system to extract the genealogical information in family history books. The status of the reading system is reported. Also explained is how the generated results can be imported into and thus contribute to the construction of a large repository of world-wide family interrelationships. The reading system's potential use for constructing similar knowledge repositories in other domains is foreshadowed.*

Keywords. Document Reading • Information Extraction • Conceptual Modeling • Ontology Conceptualization • Extraction Ontology

1 Introduction

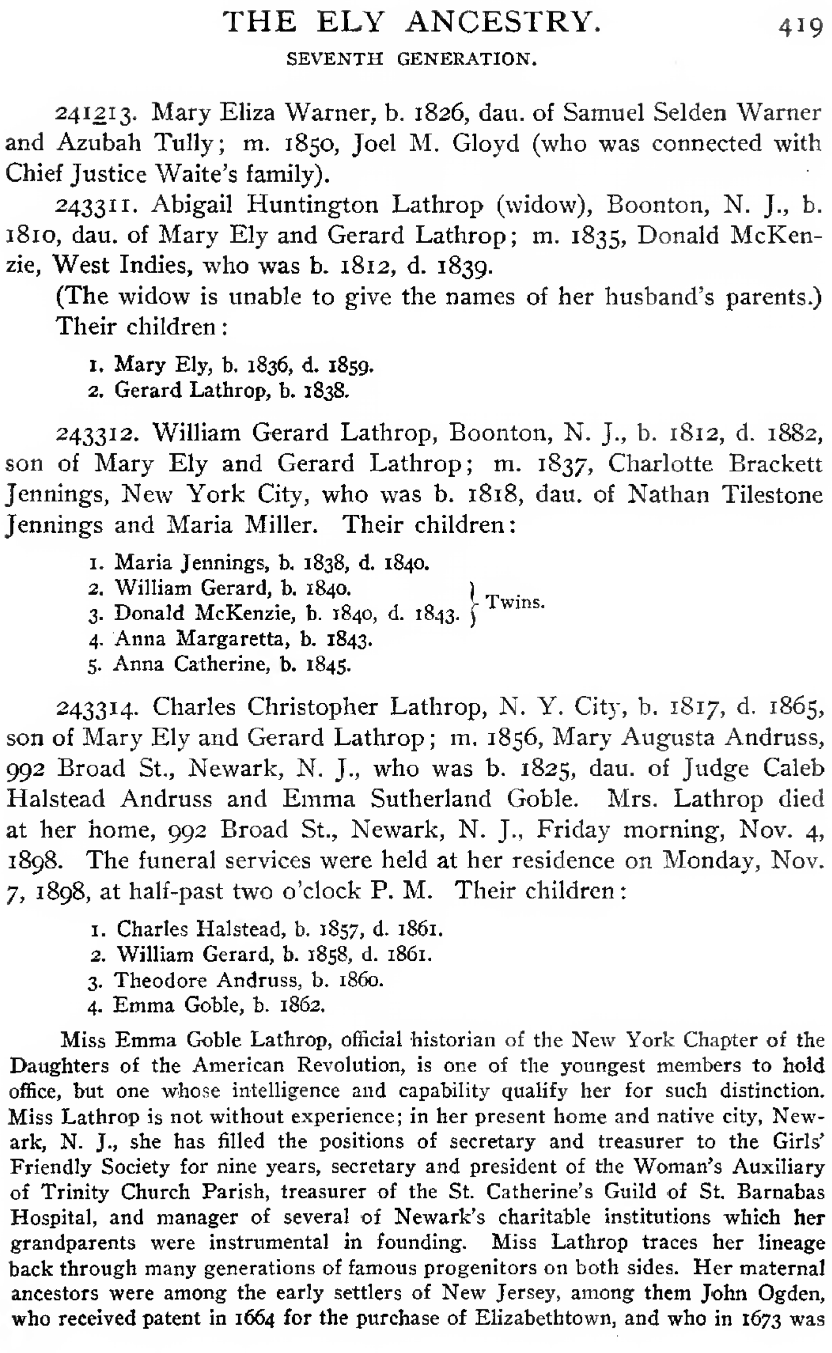
Ontology is the philosophical study of the nature of being, existence, or reality. *An ontology* is a shared, commonly agreed upon conceptualization of a domain of interest (Gruber 1993). An ontology can be represented as a conceptual model, which names and defines the types, properties, and interrelationships of the objects that exist in a particular domain (Dillon et al. 2008). (Also see (Guizzardi and Halpin 2008).)

Ontological document reading, in its simplest form, consists of automatically populating a conceptual model with object and relationship instance facts extracted from a document's text. More completely described, *ontological document reading* consists of four interrelated tasks:

1. populate multiple interrelated conceptual models with facts stated in a document and integrate the extracted information into a unified whole;
2. check the extracted facts with respect to ontological constraints supplied as part of the conceptual models;
3. infer facts called for in the conceptual models but not explicitly stated in the document; and
4. construct and augment conceptual models to capture document facts not conceptualized in a given collection of conceptual models.

As an example of ontological document reading, consider reading the document in Figure 1 with respect to the ontological conceptualization in Figure 2:

* Corresponding author.
E-mail. embley@cs.byu.edu



THE ELY ANCESTRY. 419

SEVENTH GENERATION.

241213. Mary Eliza Warner, b. 1826, dau. of Samuel Selden Warner and Azubah Tully; m. 1850, Joel M. Gloyd (who was connected with Chief Justice Waite's family).

243311. Abigail Huntington Lathrop (widow), Boonton, N. J., b. 1810, dau. of Mary Ely and Gerard Lathrop; m. 1835, Donald McKenzie, West Indies, who was b. 1812, d. 1839.

(The widow is unable to give the names of her husband's parents.)

Their children:

1. Mary Ely, b. 1836, d. 1859.
2. Gerard Lathrop, b. 1838.

243312. William Gerard Lathrop, Boonton, N. J., b. 1812, d. 1882, son of Mary Ely and Gerard Lathrop; m. 1837, Charlotte Brackett Jennings, New York City, who was b. 1818, dau. of Nathan Tilestone Jennings and Maria Miller. Their children:

1. Maria Jennings, b. 1838, d. 1840.
2. William Gerard, b. 1840. } Twins.
3. Donald McKenzie, b. 1840, d. 1843. }
4. Anna Margaretta, b. 1843.
5. Anna Catherine, b. 1845.

243314. Charles Christopher Lathrop, N. Y. City, b. 1817, d. 1865, son of Mary Ely and Gerard Lathrop; m. 1856, Mary Augusta Andruss, 992 Broad St., Newark, N. J., who was b. 1825, dau. of Judge Caleb Halstead Andruss and Emma Sutherland Goble. Mrs. Lathrop died at her home, 992 Broad St., Newark, N. J., Friday morning, Nov. 4, 1898. The funeral services were held at her residence on Monday, Nov. 7, 1898, at half-past two o'clock P. M. Their children:

1. Charles Halstead, b. 1857, d. 1861.
2. William Gerard, b. 1858, d. 1861.
3. Theodore Andruss, b. 1860.
4. Emma Goble, b. 1862.

Miss Emma Goble Lathrop, official historian of the New York Chapter of the Daughters of the American Revolution, is one of the youngest members to hold office, but one whose intelligence and capability qualify her for such distinction. Miss Lathrop is not without experience; in her present home and native city, Newark, N. J., she has filled the positions of secretary and treasurer to the Girls' Friendly Society for nine years, secretary and president of the Woman's Auxiliary of Trinity Church Parish, treasurer of the St. Catherine's Guild of St. Barnabas Hospital, and manager of several of Newark's charitable institutions which her grandparents were instrumental in founding. Miss Lathrop traces her lineage back through many generations of famous progenitors on both sides. Her maternal ancestors were among the early settlers of New Jersey, among them John Ogden, who received patent in 1664 for the purchase of Elizabethtown, and who in 1673 was

Figure 1: Page 419 of The Ely Ancestry (Vanderpoel 1902)

1. *Ontology Population*
   Extraction results should include the following facts: *Person(Mary Eliza Warner) was born on BirthDate(1826). Person(Mary Eliza Warner) married Spouse(Joel M. Gloyd) on MarriageDate(1850) at BirthPlace(?). Child(Maria Jennings) is a child of Person(William Gerard Lathrop). Child(Maria Jennings) is a child of Person(Charlotte Brackett Jennings).*
2. *Constraint Checking*
   Human readers should, and usually do, implicitly check extracted facts against ontological reality. Automated extraction tools, with no intuition of their own, often make extraction mistakes that are wildly unreasonable. For example, because the OCR system makes a mistake—interpreting "1" in Theodore Andruss' birth year as "i", which actually happens in the OCR of the document in Figure 1—an automated extractor can extract *Person(Theodore Andruss) was born on BirthDate(860).* By itself this extracted fact is reasonable, but not in the context of other extracted facts: *Child(Theodore Andruss) is a child of Person(Mary Augusta Andruss)* and *Person(Mary August Andruss) was born on BirthDate(1825).* These extracted facts along with the constraint that a child cannot be born before its mother imply that at least one of them is incorrect.
3. *Fact Inference*
   Document authors typically do not explicitly state all the facts they wish to convey. Readers of the page in Figure 1, for example, should infer (a) the gender of a person (e.g. *Person(Mary Eliza Warner) has Gender(Female))*; (b) roles such as father, mother, wife, husband (e.g. *Mother(Abigail Huntington Lathrop)* because she is female and has children); (c) full married names of female spouses based on cultural traditions (e.g. *Person(Mary Eliza Warner) has InferredMarriedName(Mary Eliza Warner Gloyd)*; and (d) full birth names (e.g. *Person(Maria Jennings) has Inferred-BirthName(Maria Jennings Lathrop))*. Readers should also be able to determine that some name instances refer to the same person. In the last family on the page in Figure 1, for example, *Person(Mrs. Lathrop)* is the same as *Person(Mary Augusta Andruss)*.
4. *Ontology Construction and Augmentation*
   Using a named-entity recognizer, we can create a simple ontology with two linked concepts, the entity *E* and its name *EName* (e.g. *Location has LocationName*). Note that this conceptualization is similar to the conceptualization *Person has Name* in Figure 2. Furthermore, just as the appearance of a person's name instantiates a *Person* object and links it to the name, the appearance of a location name instantiates a *Location* object and links it to the location's name. From the page in Figure 1 three of the extractable location facts are *Location(West Indies)*, *Location(Boonton, N. J.)*, and *Location(N. Y. City)*. A reader can infer from the text that there is an association respectively between these locations and *Person(Donald McKenzie)*, *Person(William Gerard Lathrop)*, and *Person(Charles Christopher Lathrop)*. Natural language processing systems can too. Hence, having determined that there is an implied association between *Person* and *Location*, an automated ontology construction system can connect the two ontologies, augmenting both, on its way to constructing an ever-growing collection of ontologies that can be populated by an ontological document reading system.

A tremendous amount of academic research has contributed to the grand challenge of document understanding. Ontology, the nature of reality, and epistemology, the theory of knowledge, have been the subject of research since the days of Aristotle (Aristotle about 350BC). In modern times, several disciplines within Computer Science and Linguistics have contributed to document understanding as indicated by the many conferences and workshops that have sprung up surrounding the topic (e.g. TREC, ICDAR, ICPR, DAS, AAAI, ER, SIGMOD, SIGIR, and SIGRAPH, among others). The grand challenge of document understanding involves much more than just document reading.

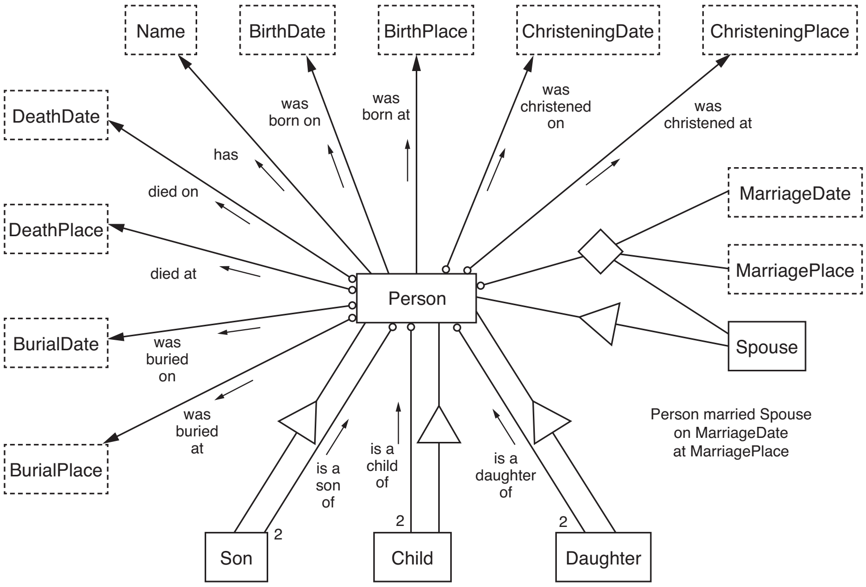


Figure 2: A Genealogy Ontology

It includes image processing, handwriting recognition, optical character recognition, identification of document components (e.g. figures, tables, text), determination of text reading order, and much more. Document reading, on which we focus in this experience report, is an essential component, but only a component, of the larger grand challenge.

The contributions of this report include:

1. a definition and description of ontological document reading;
2. a prototype implementation of an ontological document reading system; and
3. a real-world application of the implementation that extracts and organizes information for and contributes to the online and ever-growing *Family Tree* (FamilySearch n.d.).

We present the details of these contributions as follows. Section 2 explains how we populate conceptual models from text. Section 3 describes our information-extraction tools and how they map the facts they extract into ontological conceptualizations. Section 4 discusses the integration of the conceptual models our extraction tools use and of the information that populates these conceptual models. Section 5 describes how we reason about extracted and inferred facts with respect to given ontological constraints. It also describes how the system can sometimes resolve invalid extraction results and can always at least point specifically to what is likely wrong, so that system users can resolve them. Section 6 explains how we infer facts of interest implied by, but not stated in a document. It also discusses resolving object identity through coreference reasoning for multiple mentions of the same person. Section 7 explains how we can (semi)automatically construct and evolve ontological conceptualizations so that additional information can be gleaned from a document. Section 8 gives the status of our implementation and the results of some field experiments we have conducted to evaluate the effectiveness of our document-reading system. It also reports on some initial contributions to *Family Tree*, and it looks to future work and application-enhancement opportunities. Section 9 summarizes the work and makes concluding remarks.

2 Ontological Text Extraction

Research on automated text extraction began in the information retrieval community and can be dated back at least to Salton's groundwork (Salton 1968). Researchers in other computer science disciplines soon joined in the quest to extract information from text. NLP researchers have made significant progress on named entity recognition (NER) in free running text (Nadeau and Sekine 2007) and have more recently focused on recognizing relationships among named entities (e.g. (Schone and Gehring 2016)). Researchers within the database, library/information science, document analysis, and AI communities have sought to make documents more easily searchable and to extract specified items of information. Hundreds of research papers in these various disciplines have been published in many journals and conference proceedings. Surveys of this work include (Laender et al. 2002), (Turmo et al. 2006), (Sarawagi 2008), (Grishman 2015), and (Jiménez et al. 2016).

For ontological document reading, extraction results must be mappped to an ontology. Early extraction systems (e.g. (Lehnert et al. 1994), (Kushmerick et al. 1997), and those surveyed in (Eikvil 1999)) labeled extracted data with category names, often called "slots." But these slot conceptualizations lacked the structure and interconnectedness of larger ontologies and the richness of their constraints and inference capabilities. Later work on linguistically grounding ontologies begins to open the door to ontological document reading (Buitelaar et al. 2009).

In our extraction work, we create ontologies as the target for information extraction and directly populate these ontologies with our information extraction engines. We specify ontologies as conceptual models using the OSM conceptual modeling language (Embley et al. 1992), which can be rendered both textually and graphically. We augment these conceptual models with extraction rules that map document text to object and relationship instances in the conceptual model's object and relationship sets. An ontological conceptual model augmented with extraction rules is an *extraction ontology* that can read a document and populate its object and relationship sets with information gleaned from the document. We have developed an ensemble of tools for creating extraction rules and for otherwise generating mappings from text to ontological conceptualizations. These tools rely on techniques for developing rule-based expert systems and for doing natural-language processing (NLP), cognitive reasoning, and machine learning.

2.1 Ontological Conceptualizations

Figure 3 shows an OSM conceptual model. Rectangular boxes are *object sets*—dashed if lexical and solid if non-lexical. The objects stored in lexical object sets are strings. In Figure 3 *SpouseName* and *Year* are lexical and may have strings such as "Mary Augusta Andruss" and "1826" as members of their respective object sets. Non-lexical objects are represented as numbered object identifiers—(e.g. "osmx73", because our underlying representation for **OSM** conceptual models is **XML**).

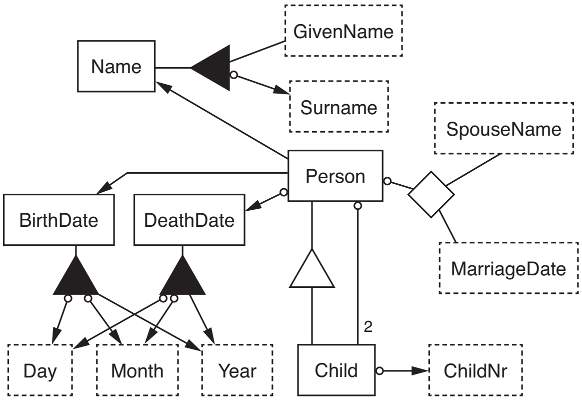


Figure 3: Basic Ely Ontology

Lines between object sets denote *relationship sets*; a diamond may appear in the middle of a line and usually does for three or more connecting object sets—e.g. the marriage relationship set in Figure 3. Object-set/relationship-set connections each have a *participation constraint*, a special kind of cardinality constraint (Liddle et al. 1993) that specifies the number of times an object in an object set can participate in a relationship set. The *2* in Figure 3, for example, specifies that a *Child* is related to exactly two *Persons*—the child's parents. Graphical decorations on object-set/relationship-set connections denote common participation constraints: a small "o" for **o**ptional participation and the absence of an "o" for mandatory participation; and an arrowhead for functional participation from tail object set to head object set, which limits objects in the tail-side object set to participate at most once. In Figure 3, for example, a *Person* mandatorily has one *BirthDate* and optionally has one *DeathDate*. Also, *Person*s must have a *Name*, but they need not be married nor have children. Relationship sets have names, either explicit, as are the relationship-set names in Figure 1, or implicit, as in Figure 3. Both explicit and implicit relationship-set names include the names of the connected object sets. Explicit relationship-set names for binary relationship sets have a reading direction arrow that specifies which of the object set names comes before and which comes after the connecting verbage in the full relationship-set name. In Figure 1, *Person was buried at BurialPlace* and *Son is a son of Person* are two of the relationship-set names. Implicit names are a space concatenation of the object-set names in any order—e.g. *Person SpouseName MarriageDate* for the marriage relationship set in Figure 3. By default, implicit names for functional binary relationship sets, may use *has* as the connecting verbage, read from tail to head—e.g. *Person has BirthDate* in Figure 3.

A white-filled triangle denotes a *generalization/specialization* with one or more specializations connected to the base and a generalization connected to the apex. The objects in a specialization are a subset of the objects in a generalization, and all the connecting relationship sets of a generalization are inherited by its specialization(s). In Figure 3, every *Child* object is also a *Person* object and has a *Name* and *BirthDate* and may be married and may have died. A *Child* may have a *ChildNr*, but a *Person* who is not a *Child* may not have a *ChildNr*.

A black-filled triangle denotes an *aggregation* with two or more component-part object sets connected to the base and the aggregate object set connected to the apex. Aggregate and component-part object sets can independently either be lexical or non-lexical. The black-filled triangle is a just a grouping of ordinary relationship sets with an implied composition whose implicit names are all *x is part of y* where *x* is a component-part object-set name and *y* is the aggregate object-set name. Decorations on the connections specify participation requirements in the aggregate. In Figure 3 *BirthDate* and *DeathDate* are each an aggregate, optionally, of a *Day* and *Month*, and, mandatorily, of a *Year*; and *Name* is an aggregate of one or more *GivenName*s and an optional *Surname*.

2.2 Extraction Ontologies

When writing the *The Ely Ancestry* (Vanderpoel 1902), the author may have conceptually had in mind the ontology in Figure 3 populated with information. The job of a document reader is to reverse the process—extract the information from the book and populate the ontology. We enable an ontology to populate itself by attaching enriched linguistic recognizers to every object set and relationship set and also to selected *ontology snippets*—coherent subcomponent views of an ontology. A reading-enabled ontology is an *extraction ontology* (Embley et al. 1999a; Embley and Zitzelberger 2010; Park 2015).

In an extraction ontology, every lexical object set has an associated *data frame* (Embley 1980). In essence, a data frame for a lexical object L describes the objects that may populate L including how to recognize object instances in a document and how these objects may behave and interact with other objects. More formally, a data frame is an abstract data type whose value set V has an instance recognizer that identifies lexical patterns denoting values in V and whose set of operations O includes an input operator to convert identified instances to the internal representation for V and an output operator to convert instances in V to strings, as well as applicable operations such as the Boolean operator "between" for two successive dates. Except for input and output, each operation o in O has an operator recognizer that identifies lexical patterns as indicators that *o* applies, e.g. "is between (Date) and (Date)".

Figure 4 shows an example of a *BirthDate* data frame. From the value expression and the left and right context expressions, we form a regular-expression extraction rule by concatenating the given regular expressions and placing capture-group parentheses around the value expression. The extraction rule generated from the *YearOnly* recognizer in Figure 4—`b[.,]?\s(\b\d{4}\b)`—extracts 17 of the 18 birth years in Figure 1 into the *BirthDate* object set in Figure 2. (Theodore Andruss' birth year has an OCR error, "i860" instead of "1860", causing his birth year to be missed.) The *MonthDayYear* expression in Figure 4 recognizes dates like "Nov. 4, 1898" in Figure 1, in which `{Month}` is a macro referring to a lexicon of month names and abbreviations. Keyword phrases like `born` and `birth` in Figure 4 are not part of the regular-expression extraction rule. Instead, if data-frame extraction rules of two or more object sets recognize the same string of characters as being possible instances of themselves, the keywords and keyword phrases help disambiguate the intended target object set for the extraction. For example, to which *Date* object set in Figure 2 does the date "Nov. 4, 1898" in Figure 1 belong? A keyword `died` found in the same sentence as the date indicates that it should be extracted into the *DeathDate* object set, which has an identical *MonthDayYear* recognizer.

```
internal representation: int // Julian date: yyyyddd
external representation:
    // YearOnly
        value expression: \b\d{4}\b
        left context expression: b[.,]?\s
        right context expression:
        keyword phrases: \bborn\ | \bbirth\
    // MonthDayYear
        value expression:
        \b{Month}\s(?:1\d|2\d|30|31|\d)[.,]?\s(?:\d{4})\b
    ...
methods:
    ageAtDeath(b:BirthDate, d:DeathDate) returns int {
        return d/1000 - b/1000
    }
    ...
```

Figure 4: Sample BirthDate Data Frame (partial)

In an extraction ontology, every non-lexical object set is populated by *ontological commitment*—a relation between a language and objects postulated to exist by that language. *Person* objects in Figure 2 are instantiated by the appearance of a person's name in a document. In a data frame for a non-lexical object set N we specify which related lexical object instantiations also instantiate an object of N. The instantiation of objects in N also instantiates relationships between non-lexical objects and their related lexical objects. For example, extracting "Mary Eliza Warner" in Figure 1 into the *Name* object set in Figure 2 causes a new object, say *osmx103*, to be placed in the *Person* object set and the relationship *Person(osmx103) has Name(Mary Eliza Warner)* to be placed in the *Person has Name* relationship set. Because of ontological commitment, we can and often do write this relationship as *Person(Mary Eliza Warner)*.

In general, ontological commitment may be declared directly by association with one or more lexical object sets or indirectly through one or more non-lexical object sets. In Figure 3, for example, the non-lexical *Name* object set is instantiated when either of the lexical object sets *GivenName* or *Surname* is instantiated, and the instantiation of a *Person* object happens when an object is instantiated in the non-lexical *Name* object set. Objects in the non-lexical object set *Child* in Figure 3 are instantiated by inheritance when a name is known to be a child name. The object existence rule `\b\d\d?[.]\s{Person}` identifies a child in Figure 1 by a person name following the appearance of a one- or two-digit number, a period, and a space. In this object existence rule `{Person}` is a macro which devolves to the macro `{Name}` which, in turn, devolves to the regular-expression rule for either *GivenName* or *Surname*.

Extraction rules for relationship sets build on data-frame-specified extraction rules for object sets. As an example, the rule `{Person}[\s\S]{1,30}?b\.\s{BirthDate}` correctly extracts from Figure 1 most of the *Person was born on BirthDate* relationships for the extraction ontology in Figure 2. (It incorrectly assign's Emma Goble's birth year to Theodore Andruss because of the OCR error, "i860", and it assigns Mary Augusta Andruss' birth year to what it thinks is a person name in the address text between Mary's name and her birth year.) The macro references to `{Person}` and `{BirthDate}` illustrate how relationship-set extraction rules build on object-set extraction rules. All combinations of value-expressions in the data frames for each referenced object set are plugged in to create extraction rules for the relationship set.

Extending relationship-set extraction rules to span across multiple relationship sets lets us define extraction rules for a coherent subcomponent of an ontology. We call these subcomponents *ontology snippets*. As an example, we can write the ontology snippet extraction rule

```
(\d)\.\s([A-Z]\w+)\s([A-Z]\w+),\sb[,.]
\s([\d|i]\d{3})(?:,\sd\.\s(\d{4}))?\.
```

which extracts all eleven numbered children along with their birth year, and, if stated, their death year. To complete the extraction rule we must match capture groups with object sets. For the ontology in Figure 3, Capture Group *1* associates with *ChildNr*, *2* and *3* associate with *GivenName*, *4* associates with *BirthDate Year*, and *5* associates with *DeathDate Year*. Note that we can associate an ordered sequence of capture groups with a single object set, as we do here for *GivenName*. This feature also lets us capture lexical items that do not appear together such as name and surname in a phone book listing where the surname is factored to the top of the list of all names with the same surname.

3 Learning to Read

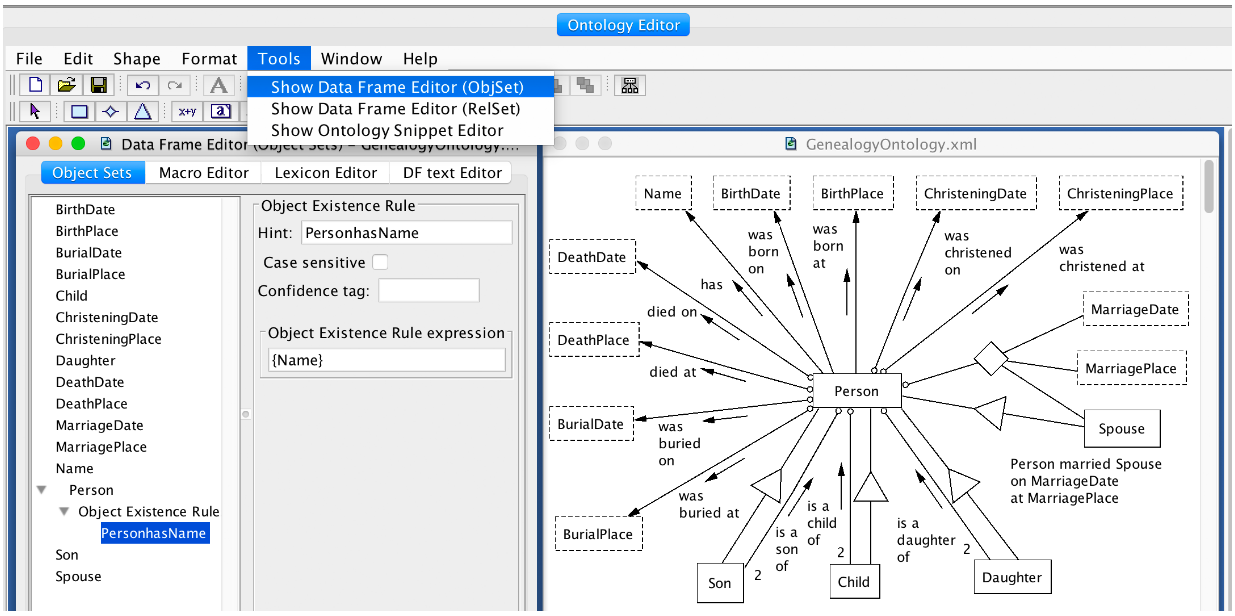
There are a number of ways a computer can be taught to read: (1) It can be told what the patterns in the text mean (Section 3.1). (2) It can be told how ontology-equivalent forms should be filled in (Section 3.2). (3) It can learn by example (Sections 3.3 and 3.4). (4) It can discover patterns and how they map to conceptualizations (Section 3.5). (5) It can learn from training data (Section 3.6). (6) It can learn by natural language processing (NLP) and cognitive reasoning (Section 3.7).

Before describing our extraction tools, we first observe that for historical documents we are given neither the words in the document nor the lines of text. Instead the OCR provides us only with the characters recognized and their bounding boxes. Reconstructing the text flow in the document can sometimes be non-trivial. Particular problems occur when the point size changes or when words are typeset with comparatively large spaces between letters to maintain right justification. The document in Figure 1 has neither of these problems, and the reconstructed text is relatively clean. Other than the OCR errors of "i" for "1" and "." for "," and vice versa which we have already mentioned or alluded to, the only error is the failure of the OCR engine to recognize "Twins" which is between the lines and the brace which spans multiple lines. Reconstructing tab stops is another matter. We usually left justify each line in the OCR, but have the option to add tab stops at the beginning of lines at what appears to be every level of indentation—four of them for the page in Figure 1.

3.1 FROntIER: Extraction Rule Creation with Data Frames

Figure 5 shows our FROntIER ontology workbench (Park 2015) with which we can create extraction rules by hand as explained in Section 2.2. The rectangle, diamond, and triangle icons at the top let users respectively create object sets, relationship sets and generalization/specializations of an ontology. The *Tools* menu lets users select which kind of extraction rule they wish to create. The data frame rule creation tool is currently open showing an object existence rule that has been created.

We note that extraction rule creation by machine learning is more popular in academic circles than in industry. When it comes to practical application, however, rule creation is the clear winner, especially for large vendors, with 67% of their implementations being pure rule-based systems



Hint: PersonhasName

Object Existence Rule expression
{Name}

Figure 5: Ontology Workbench

and another 17% being a hybrid of rule-based and machine-learned information extraction systems (Chiticariu et al. 2013). Five of seven of our extraction tools are based on rules. In three of the five, however, the rules are system generated.

3.2 OntoES: Form-based Extraction Rule Creation

Forms can be designed so that they have a one-to-one correspondence with an ontology. Creation of such a form induces an ontology (Tao et al. 2009). In Figure 6 the form being created corresponds to the ontology in Figure 7. The form title *Person* becomes a non-lexical object set, and the single-entry form fields nested under *Person* become lexical object sets functionally dependent on *Person*. The construction menu in Figure 6 is attached to the *BirthDate* field. By clicking on *Single* we could nest lexical *Day*, *Month*, and *Year* fields under what would then be a non-lexical *BirthDate* field. We can create the ontology in Figure 8 by making the form title be *Couple* and nesting under it a multiple-entry field, extended to have three form fields, *SpouseName*, *MarriageDate*, and *MarriagePlace*. Similarly, we can create the ontology in Figure 9 by making the form title be *Family* and nesting under it two single-entry form fields, *Parent1* and *Parent1*, and a multiple-entry form field *Child*.

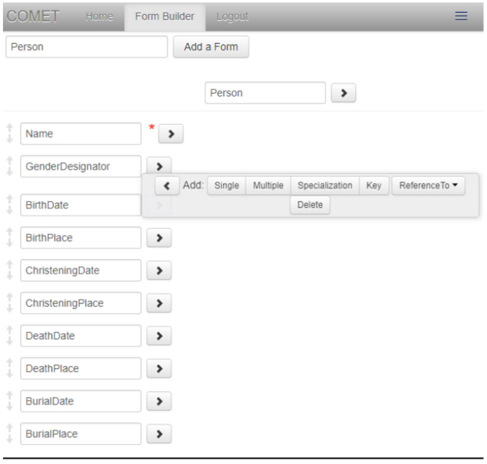


Figure 6: Form Creation and Ontology Induction

We can also create ontologies that have specializations. Clicking on *Specialization* in the construction menu for a field in Figure 6 nests a specialization under it. Then with the construction menu attached to the specialization, clicking on *Reference To* allows a user to select any of the

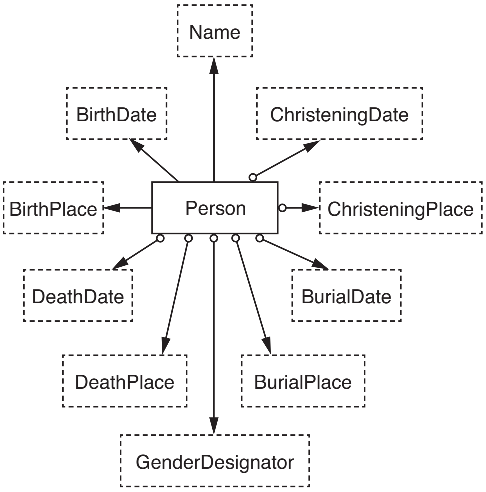


Figure 7: Person Ontology

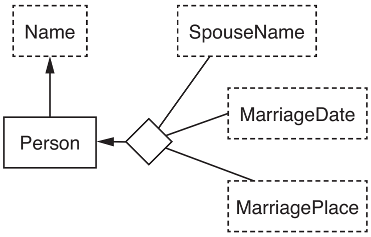


Figure 8: Couple Ontology

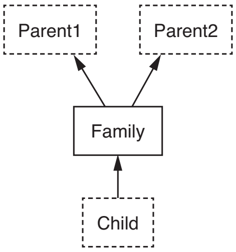


Figure 9: Family Ontology

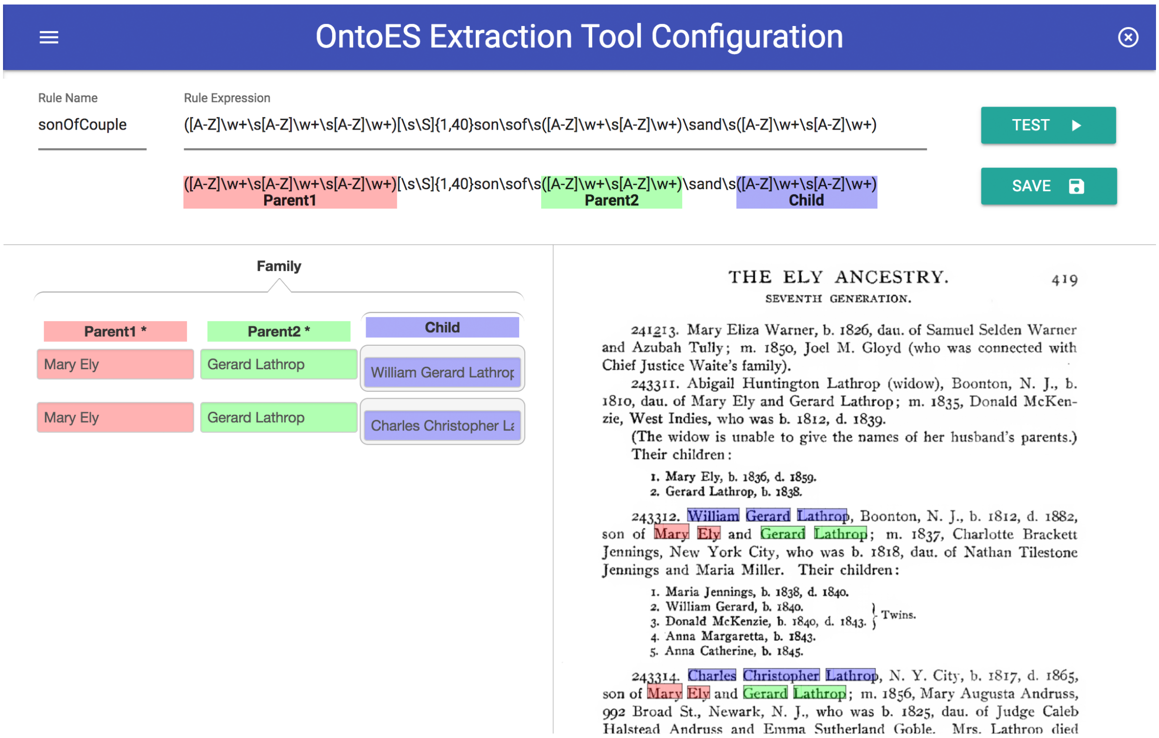
existing object sets to which a connecting relationship set is added. Thus, for example, we can create *Child* in Figure 3 as a specialization of *Person* and also create the relationship between *Child* and *Person*. *Key* in the construction menu is for specifying ontological commitment and designates the field to which the menu is attached as an instantiator of the non-lexical field under which it is nested. The star next to the *Name* field in Figure 6 marks *Name* as the instantiator field for the *Person* ontology in Figure 7.

Having built forms, a user *U* can "teach" the reading system how to fill them in. Clicking on the menu icon in the upper right of Figure 6, lets *U* choose a form and a page in a book and then bring up the interface in Figure 10, albeit initially with only the page and form header. *U* can then create an extraction rule by naming it and typing in a regular expression whose capture groups correspond to form fields. After specifying the match between capture groups and form fields, *U* can click on the *Test* button to test the extraction rule. Figure 10 shows the result: the form records are created and filled in, and the highlighting shows the correspondence among filled in form fields, regular-expression capture groups, and text extracted from the page. If satisfactory, *U* can save the rule along with others in an extraction ontology for the form.

3.3 GreenFIE: Extraction Rule Learning by Form Filling Examples

Rather than writing regular expressions to tell the computer how to fill in form records, a user can fill in a form manually. In the background, GreenFIE (Kim 2017) "watches" a user fill in a form record, generates a regular-expression extraction rule that would also have filled in the same record, generalizes the rule, executes it, and automatically fills in form records with information that matches the generalized extraction rule.

Figure 11 shows the interface of our form-filling tool, which we call COMET—**C**lick-**O**nly, or at least **M**ostly, **E**xtraction **T**ool (Embley et al. 2017). To fill in a field currently in focus (i.e. the one with its border highlighted), a user need only click



Rule Name: sonOfCouple

Rule Expression: ([A-Z]\w+\s[A-Z]\w+\s[A-Z]\w+)[\s\S]{1,40}son\sof\s([A-Z]\w+\s[A-Z]\w+)\sand\s([A-Z]\w+\s[A-Z]\w+)

Family

Parent1 *	Parent2 *	Child
Mary Ely	Gerard Lathrop	William Gerard Lathrop
Mary Ely	Gerard Lathrop	Charles Christopher La

THE ELY ANCESTRY. 419

SEVENTH GENERATION.

241213. Mary Eliza Warner, b. 1826, dau. of Samuel Selden Warner and Azubah Tully; m. 1850, Joel M. Gloyd (who was connected with Chief Justice Waite's family).

243311. Abigail Huntington Lathrop (widow), Boonton, N. J., b. 1810, dau. of Mary Ely and Gerard Lathrop; m. 1835, Donald McKenzie, West Indies, who was b. 1812, d. 1839.

(The widow is unable to give the names of her husband's parents.) Their children:

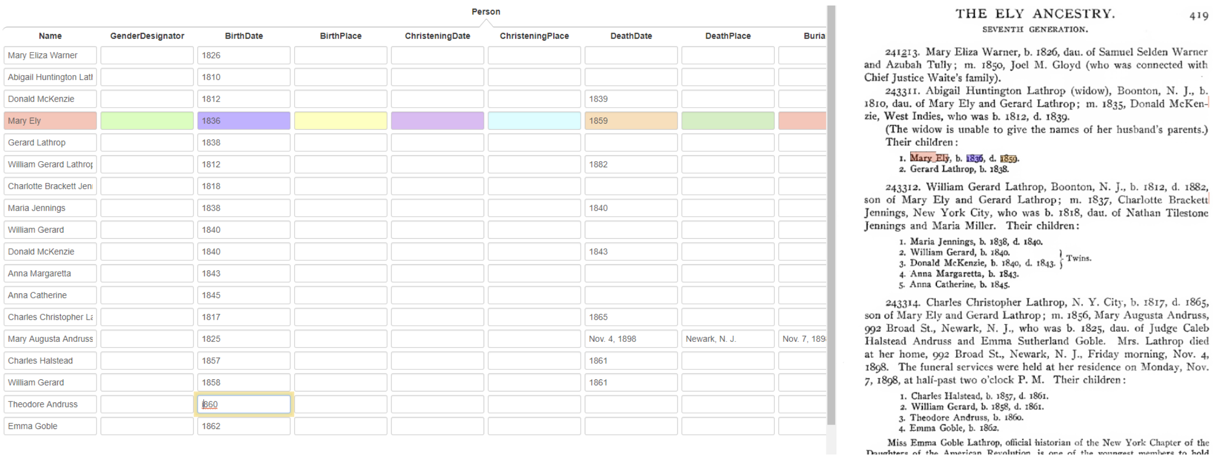
1. Mary Ely, b. 1836, d. 1859.
2. Gerard Lathrop, b. 1838.

243312. William Gerard Lathrop, Boonton, N. J., b. 1812, d. 1882, son of Mary Ely and Gerard Lathrop; m. 1837, Charlotte Brackett Jennings, New York City, who was b. 1818, dau. of Nathan Tilestone Jennings and Maria Miller. Their children:

1. Maria Jennings, b. 1838, d. 1840.
2. William Gerard, b. 1840. } Twins.
3. Donald McKenzie, b. 1840, d. 1843. }
4. Anna Margaretta, b. 1843.
5. Anna Catherine, b. 1845.

243314. Charles Christopher Lathrop, N. Y. City, b. 1817, d. 1865, son of Mary Ely and Gerard Lathrop; m. 1856, Mary Augusta Andruss, 992 Broad St., Newark, N. J., who was b. 1825, dau. of Judge Caleb Halstead Andruss and Emma Sutherland Goble. Mrs. Lathrop died

Figure 10: OntoES Rule Editor



Person

Name	GenderDesignator	BirthDate	BirthPlace	ChristeningDate	ChristeningPlace	DeathDate	DeathPlace	Buria
Mary Eliza Warner		1826						
Abigail Huntington Latl		1810						
Donald McKenzie		1812				1839		
Mary Ely		1836				1859		
Gerard Lathrop		1838						
William Gerard Lathrop		1812				1882		
Charlotte Brackett Jeni		1818						
Maria Jennings		1838				1840		
William Gerard		1840						
Donald McKenzie		1840				1843		
Anna Margaretta		1843						
Anna Catherine		1845						
Charles Christopher La		1817				1865		
Mary Augusta Andruss		1825				Nov. 4, 1898	Newark, N. J.	Nov. 7, 189
Charles Halstead		1857				1861		
William Gerard		1858				1861		
Theodore Andruss		1860						
Emma Goble		1862						

THE ELY ANCESTRY. 419

SEVENTH GENERATION.

241213. Mary Eliza Warner, b. 1826, dau. of Samuel Selden Warner and Azubah Tully; m. 1850, Joel M. Gloyd (who was connected with Chief Justice Waite's family).

243311. Abigail Huntington Lathrop (widow), Boonton, N. J., b. 1810, dau. of Mary Ely and Gerard Lathrop; m. 1835, Donald McKenzie, West Indies, who was b. 1812, d. 1839.

(The widow is unable to give the names of her husband's parents.) Their children:

1. Mary Ely, b. 1836, d. 1859.
2. Gerard Lathrop, b. 1838.

243312. William Gerard Lathrop, Boonton, N. J., b. 1812, d. 1882, son of Mary Ely and Gerard Lathrop; m. 1837, Charlotte Brackett Jennings, New York City, who was b. 1818, dau. of Nathan Tilestone Jennings and Maria Miller. Their children:

1. Maria Jennings, b. 1838, d. 1840.
2. William Gerard, b. 1840. } Twins.
3. Donald McKenzie, b. 1840, d. 1843. }
4. Anna Margaretta, b. 1843.
5. Anna Catherine, b. 1845.

243314. Charles Christopher Lathrop, N. Y. City, b. 1817, d. 1865, son of Mary Ely and Gerard Lathrop; m. 1856, Mary Augusta Andruss, 992 Broad St., Newark, N. J., who was b. 1825, dau. of Judge Caleb Halstead Andruss and Emma Sutherland Goble. Mrs. Lathrop died at her home, 992 Broad St., Newark, N. J., Friday morning, Nov. 4, 1898. The funeral services were held at her residence on Monday, Nov. 7, 1898, at half-past two o'clock P. M. Their children:

1. Charles Halstead, b. 1857, d. 1861.
2. William Gerard, b. 1858, d. 1861.
3. Theodore Andruss, b. 1860.
4. Emma Goble, b. 1862.

Miss Emma Goble Lathrop, official historian of the New York Chapter of the Daughters of the American Revolution, is one of the youngest members to hold

Figure 11: GreenFIE User Interface

on the text string or strings in the document to be filled into the form. For historical documents, the image of a document page is super imposed over hidden OCR'ed text, so that clicking on a word in the image extracts the OCR'ed text of the word and enters it into the focus field. In Figure 11, for example, the field in focus is the *BirthDate* field for Theodore Andruss, and when a user clicks on "1860" in the image, the underlying OCR'ed text "i860" is entered into the field. The user can correct the OCR error by double-clicking on the field and editing the text. To make it easy to see what text in the document has been extracted into which form record, a user can use a mouse to hover over a record, which then highlights each field in the record with a different color and also highlights in the document the text filled into a field with the same color. In Figure 11, the user is hovering over the record of Mary Ely who was born in 1836 and died in 1859.

The filled in form records in Figure 11 were generated with the help of GreenFIE as follows: Beginning with a single empty record, a user *U* extracted the highlighted Mary Ely information into the empty record. *U* then clicked on the *Regex* button at the end of the filled-in record. (In Figure 11, the *Regex* buttons are hidden behind the document page, but are accessible with a horizontal slider bar below the form records.) As a result, GreenFIE added the four other children in the page that also have both a birth and a death date. (After adding records, GreenFIE sorts them in page order, which is why Mary Ely's record eventually ends up as the fourth record in Figure 11.)

Next, *U* filled in a new empty record for Mary Ely's brother Gerard Lathrop, who has only a birth year, and clicked on the record's *Regex* button. For this case GreenFIE-generated the regular expression,

```
\n\d{1}[.,]\s([A-Z][a-z]+\s(?:(?:Mc
[A-Z][a-z]+)|(?:[A-Z][a-z]+))[.,]\s
b[.,]\s(\d{4})[.,]
```

which we use here for illustration since it is the shortest of all the expressions. GreenFIE generates and generalizes using several heuristics: (1) The left context for a capture group is the text back to the first white space character before the word preceding the capture group. (2) The right context of a capture group consists of all immediate punctuation characters following the capture group; or if none, then `\s` is added. (3) Between capture groups, if the right of the first overlaps or abuts against the left of the second, the text between is the literal context; otherwise the left and right context is kept and a skip, `[\s\S]{`m`,`n`}`, is added where m and n are set heuristically depending on the actual number of characters being skipped. (4) Some OCR errors are anticipated, here just "." for "," and vice versa. (5) Strings of n digits become `\d{n}`. (6) Person names, dates, and place names are selected from a library of regular expressions depending on which expressions match the person names, dates, and place names in the given example. The regular expression here matches all the numbered children with only a birth year except Theodore's record which has the mentioned OCR error. It also matches all the numbered children with both a birth year and a death year, but GreenFIE keeps only subsuming records and thus does not generate new form records for them.

Continuing, user *U* next filled in a record for Mary Eliza Warner and her birth date, but the generated regular-expression found no other matching text pattern. Similarly, generated records for both Abigail Huntington Lathrop and the West Indies Donald McKenzie yielded no other form records. The generated expression for William Gerard Lathrop of Boonton, N. J., however, does match the record for Charles Christopher Lathrop of N. Y. City, and the expression generated for Charlotte Brackett Jennings does match the record for Mary Augusta Andruss. The generated record for Mary, however, is incomplete, so *U* added the death date and place and also the funeral date as the burial date.

Extraction rules for the *Couple* and *Family* forms are generated similarly, except that the multiple entry fields in both forms call for an additional type of generalization for lists. For the Abigail and Donald family in Figure 1, for example, when Mary Ely and Gerard Lathrop are added as children, GreenFIE not only generalizes each entry as explained above, it also generalizes the list itself to possibly have up to a user-specified maximum number of children (12 in our implementation). GreenFIE thus generates 12 extraction rules, the first with a capture group for the first child, the second with a non-capture group for the first child and a capture group for the second child, and so forth to the 12th in which the first 11 children are recognized in non-capture groups and the 12th is recognized with a capture group. When executed, GreenFIE takes the results and stitches all the children recognized into a single record with their parents. Lists are often numbered as they are in Figure 1, and GreenFIE knows about numbering schemes (e.g. Arabic Numerals, Roman Numerals, Ordinal Numbers) and replaces the left context, when it includes the numbering scheme as it does in Figure 1, with the correct number for each list item. There are no examples of persons with multiple spouses in Figure 1. On other pages, when there are multiple spouses, they are numbered with ordinal numbers starting with the "2nd" spouse.

GreenFIE extraction rules are kept in a repository and executed in advance as a user moves from page to page. Eventually, the GreenFIE-generated extraction rules cover all but a very few exceptional cases. Thus, the extraction work of a user diminishes over time. The name "GreenFIE" stands for "**Green F**orm-based **I**nformation **E**xtraction, where "Green" is a designator for tools that improve themselves with use in real-world tasks (Nagy 2012).

3.4 GreenQQ: Extraction Rule Learning by Text Snippet Examples

Users interact with GreenQQ by means of sample snippets of the OCR'ed text (Embley and Nagy 2017). For each lexical object set S of a given ontology, users initially give GreenQQ a text-snippet example that contains a text instance t to be extracted into S. The text snippet should also contain some surrounding text tokens that help identify and classify t as being a member of S. From the text-snippet example, GreenQQ creates an extraction rule by generalizing and tagging the text tokens (e.g. identifying them as capitalized words, n-digit numbers, punctuation) and designating the position within the snippet of the textual instance to be extracted. It then sweeps this extraction-rule template across an entire document whose text has previously been generalized and tagged and returns all matching instances for the specified lexical object set. Once bootstrapped in this way, GreenQQ is also able to discover and propose new extraction rules for other instances that likely should be extracted by the current set of extraction rules but were not. Presenting these proposed rules in terms of examples, the user can accept, reject, or modify these candidate rules. GreenQQ then executes the expanded rule set and proposes yet more potential rules. This cycle continues until the user is satisfied with the results.

As an example, consider applying GreenQQ to populate the *Person* ontology in Figure 7 from *The Ely Ancestry*, one of whose pages is in Figure 1. In preparation, GreenQQ tokenizes the full 830-page book and generalizes and tags each token according to its token type. Also, in preparation, a user considers the document and designates a few keyword tokens that are likely to help classify text instances, e.g. "b.", "born", "d.", "died", "m." for *The Ely Ancestry*. The user then gives an example text snippet for each lexical object set in the ontology for which information is available and designates the text instance to be extracted, e.g. [DeathDate "d. 1859." "1859"], which specifies that *DeathDate* is the target object set for the extracted text, "1859", within the text snippet example, "d. 1859.". From this example, GreenQQ generates the extraction template

```
DeathDate  d. NUM4 .  [1,1]
```

where `[1,1]` designates that the offset of the token(s) to be extracted within the template "`d. NUM4 .`" is 1 (zero start count) and that the number of tokens to extract is 1. When GreenQQ sweeps this template pattern across the page in Figure 1, it extracts the death dates “1839”, “1859”, “1840”, “1843”, “1861”, and another “1861”, and it extracts hundreds more from the full book.

Continuing the example, GreenQQ next generates and classifies text snippet examples that it expects should be turned into extraction rules and presents a few of these for consideration. It identifies these candidate text snippets as frequent tagged text sequences surrounding user-given keywords and classified text tokens that *do not* include text tokens already labeled with the classification it is proposing. For example,

```
DeathDate: , b. 1812, d. 1882, son of
```

is one classified text snippet GreenQQ might return for consideration. From previous rules, GreenQQ has already associated the keyword `d.` with the `DeathDate` class, and the user-chosen window size of four tokens before and after the `d.` does not include a token sequence already labeled as a `DeathDate`. If the user now designates “1882” to be extracted with “d.” and “,” as the left and right context, GreenQQ generates the extraction rule

```
DeathDate  d. NUM4 ,  [1,1]
```

which when executed extracts “1882” and “1865” as additional *DeathDates* in Figure 1, and many more in the full book.

Figure 12 shows a set of GreenQQ-generated extraction rule templates that would extract most of the information of interest from Figure 1. Templates for Mrs. Lathrop’s death date and place and her burial date (implied from the funeral date) are not included. Rule templates for this information can be created but are likely unique within the full book. Because Theodore’s birth date was OCR’ed as “i860” it is also not included in the information that would be extracted by the rule set. GreenQQ would see this OCR error frequently enough in the context of “b.” that it might suggest

```
BirthDate: Theodore Andruss, b. i860. EOL 4 .
```

for consideration. In this case, the user could specify that `i860` should be extract with `b.` and `.` as its left and right context. If so GreenQQ would generate

```
BirthDate  b. ALPHANUM .  [1,1]
```

as a rule template.

Following the final rule create-and-execute cycle, GreenQQ groups the results into records from which it populates the ontology. GreenQQ only operates with ontologies that have a single non-lexical object set to which all lexical object sets are related, like the *Person*, *Couple*, and *Family* ontologies in Figures 7, 8, and 9. To form records, GreenFIE must group together all lexical objects related to a single non-lexical object and in the case when a connecting relationship set is n-ary ($n > 2$), as is the quaternary relationship set in Figure 8, must also properly group the connecting lexical objects. Since non-lexical objects are instantiated by ontological commitment, there are lexical objects in the text around which records are formed (e.g. person names for the ontologies in Figures 7 and 8 and a parent name for the ontology in Figure 9).

Depending on how book authors organize their presentation of information, grouping labeled text instances into records can be complex (Embley et al. 1999b). Sometimes record creation is straightforward, as it is for the Ely book in which the author groups all *Person*-ontology information immediately after the person’s name. In this case, GreenQQ can run through its initial output, which is a lexical-object-set-name-labeled list of tokens in book-text order, and form record groups from one *Name*-labeled object to the next. At other times, however, records are intertwined. In Figure 1, for example, the couple Mary Ely and Gerard Lathrop is inside the text snippet that tells us that Abigail Huntington Lathrop and Donald McKenzie constitute a couple. In these cases, and in general, if we can process each record type in a separate run over the GreenQQ output, we can properly group labeled objects into records. In Figure 12 we have separated the templates into

```
Name           NUM5+ . CAP CAP CAP ,      [2,3]
Name           NUM5+ . CAP CAP CAP (      [2,3]
Name           m. NUM4 , CAP CAP ,        [3,2]
Name           NUM1or2 . CAP CAP ,        [2,2]
Name           m. NUM4 , CAP CAP CAP ,    [3,3]
BirthDate      b. NUM4                    [1,1]
DeathDate      d. NUM4                    [1,1]
GenderDes...   dau.                       [0,1]
GenderDes...   son                        [0,1]
```

(a)

```
Name           NUM5+ . CAP CAP CAP ,      [2,3]
Name           NUM5+ . CAP CAP CAP (      [2,3]
MarriageDate   m. NUM4                    [1,1]
SpouseName     m. NUM4 , CAP CAP . CAP (  [3,4]
SpouseName     m. NUM4 , CAP CAP ,        [3,2]
SpouseName     m. NUM4 , CAP CAP CAP ,    [3,3]
SpouseName     m. NUM4 , CAP CAP CAP .    [3,3]

Name           of CAP CAP CAP and         [1,3]
Name           of CAP CAP and             [1,2]
Name           of CAP CAP CAP CAP and     [1,4]
SpouseName     and CAP CAP ;              [1,2]
```

(b)

```
Parent1        NUM5+ . CAP CAP CAP ,      [2,3]
Parent1        NUM5+ . CAP CAP CAP (      [2,3]
Parent2        m. NUM4 , CAP CAP . CAP (  [3,4]
Parent2        m. NUM4 , CAP CAP ,        [3,2]
Parent2        m. NUM4 , CAP CAP CAP ,    [3,3]
Parent2        m. NUM4 , CAP CAP CAP .    [3,3]
Child          NUM1or2 . CAP CAP ,        [2,2]

Child          NUM5+ . CAP CAP CAP ,      [2,3]
Child          m. NUM4 , CAP CAP . CAP (  [3,4]
Child          m. NUM4 , CAP CAP ,        [3,2]
Child          m. NUM4 , CAP CAP CAP ,    [3,3]
Child          m. NUM4 , CAP CAP CAP .    [3,3]
Parent1        of CAP CAP CAP and         [1,3]
Parent1        of CAP CAP and             [1,2]
Parent1        of CAP CAP CAP CAP and     [1,4]
Parent2        and CAP CAP ;              [1,2]
```

(c)

Figure 12: GreenQQ Templates for Information of Interest in Figure 1 for the (a) Person, (b) Couple and (c) Family Ontologies

record groups and within each group have ordered the fields according to the Ely author's presentation. For the first *Family* record-template group in Figure 12(c), for example, GreenQQ would process its initial output file by finding a `Parent1` followed immediately by a `Parent2` and then immediately by a `Child` followed zero or more `Child`-labeled text instances without any intervening other-than-`Child`-labeled text. GreenQQ knows to look for more than one `Child` because of the 1-many relationship set from *Family* to *Child*. Grouping and ordering templates automatically is likely to be non-trivial in general and may require user input.

As with GreenFIE, "Green" is a designator for tools that improve themselves with use in real-world tasks (Nagy 2012). The "QQ" in "GreenQQ" stands for "Quick" and "Quality." The process of executing a rule set on an entire book and simultaneously proposing new rules for user consideration to be used in the next iteration is "Quick" enough to allow for real-time, synergistic user interaction. As we indicate in Section 8.1.2, if a book has reasonably well structured patterns, GreenQQ can "Quickly" generate "Quality" results.

3.5 ListReader: Extraction Rule Learning by Text Pattern Discovery

Like GreenQQ, ListReader (Packer 2014) processes a full book as a unit. It discovers record patterns in the text and generates extraction rules for them in a sequence of steps:

1. *Abstract Text.* ListReader replaces various sequences of one or more characters by a more abstract version of the character sequence. The string "4. Emma Goble, b. 1862.", for example, becomes

   `[Dg].[Sp][UpLo+][Sp][UpLo+],[Sp][Lo].[Sp][Dg][Dg][Dg][Dg].`

   Each digit becomes the symbol `[Dg]`, each word that begins with an uppercase letter becomes `[UpLo+]`, punctuation characters remain as themselves, and so forth.

```
[Dg].[Sp][UpLo+][Sp][UpLo+],[Sp][Lo].[Sp][Dg][Dg][Dg][Dg].
------------------------------------------------------------
...
2. Gerard Lathrop, b. 1838.
2. William Gerard, b. 1840.
4. Anna Margaretta, b. 1843.
5. Anna Catherine, b. 1845.
4. Emma Goble, b. 1862.
...
```

Figure 13: ListReader Discovered Records

2. *Align Text.* ListReader aligns text by finding identical sequences of abstract symbols. Figure 13 shows the text strings in Figure 1 that align with the abstract symbol sequence for Emma Goble. The ellipses at the top and bottom of the list stand for the hundreds of additional text strings in the book that also have the same sequence of abstract symbols.

3. *Identify Record Templates.* Not all text patterns ListReader discovers make good record templates. For our application a record pattern must contain either numbers or capitalized words or both. (Numbers and proper nouns, which in English are capitalized words, typically denote items of interest.) A record should contain at least two of these items of interest. (A record relates at least two items.) Record patterns should not contain long sequences of lower-case words. (Lower-case words in English list records tend to be delimiters, which are usually limited to just a couple of words.) Record delimiters such as newline characters, `\n`, are good indicators of record beginnings and endings. (All the text strings in Figure 13 have newline characters immediately preceding and immediately following each string.)

4. *Process Record Templates.* To turn record templates into extraction rules, ListReader must know how the patterns map to ontological conceptualizations. Following the principles of active learning (Settles 2012), ListReader selects the record group, which when labeled, will likely provide the most benefit. Typically large groups with lengthy strings that have good record characteristics are best. Because ListReader also does cross-record labeling for fields with field identifiers such as `b.` and `d.` for birth- and death-date fields in Figure 1, it also takes into account how much cross-record labeling can occur for a chosen record group. Once ListReader selects a record group it chooses a prototypical list element, finds the page it is on, and brings up the page in COMET on the right and the form for the ListReader ontology on the left. When a user then fills in the form record from the highlighted text, ListReader has the mapping it needs to generate a regular-expression ontology snippet extraction rule. For example, if ListReader highlights `4. Emma Goble, b. 1862` in Figure 1 and displays its ontology (Figure 3) as a form, the user should use COMET to extract `4` into the *ChildNr* field, `Emma` and `Goble` in the multiple-entry *GivenName* field, and `1862` into the *Year* field nested under *BirthDate*. ListReader can now label, without having to ask for the user's help, every `b.` field with just a year in every record template group in the same way.

5. *Generate Extraction Rules.* Given the information obtained by processing record templates, ListReader can now generate the extraction rule for the template. For the record template in Figure 13 ListReader generates

```
\n(\d{1})\.\s([A-Z][a-z]+)
\s([A-Z][a-z]+),
\sb\.\s(\d{4})\.
```

where the first capture group maps to *ChildNr*, the second and third map to *GivenName*, and the fourth maps to *Year* under *BirthDate*. Observe that `[A-Z][a-z]+` exactly mirrors `UpLo+`, that the digit-sequence lengths match, that the punctuation represents itself, and that the end-of-line record identifier, `\n`, has been added as the initial character signifying that these records always start on a new line.

As the name implies, ListReader discovers lists of records. It works well for semi-structured text like the text in Figure 1 and countless other family history documents, as well as many other types of semi-structured documents. It should not be applied to free-running narrative text.

3.6 GreenML/GreenDDA: Machine Learning of Extraction Rules

We are exploring the applicability of different approaches involving machine learning (ML) in recognizing and extracting named entities and family relationships from various text types. While extractors do exist for these data types, their off-the-shelf (OTS) performance derives from models trained on other types of annotated texts, particularly newswire articles. Performance on family history books suffers, especially for finding relationships. We believe we can improve on OTS performance in three ways.

First, GreenML is an ML approach that builds minimal models during training, based on high-quality annotations collected for a specific book via user interaction. The user annotates a page of a document using COMET as described earlier. Next the GreenML system trains a model based on the results, which it then uses to annotate the next page. The user, through inspection and correction (where necessary) creates clean output, which is then added to the training set, triggering the creation of a new model for subsequent annotation. The cycle continues until the end of the book or until the user is satisfied that GreenML's accuracy is sufficient. The system is "Green" in the sense that it continuously improves its model as it receives user-checked and -corrected filled-in record forms page after page.

A second and related "Green" approach is called GreenDDA, for "**D**ecision **D**irected **A**daptation" (Nagy 2017). As with GreenML, a human supervises incremental, page-by-page training of models until some threshold point. In GreenDDA, though, the system proceeds from that point to annotate subsequent pages on its own, taking the results of each page and adding them (without human vetting) to the training set for model retraining. It then repeats this process on the remaining pages, extracting data from a full book.

Finally, instead of using an OTS ML model we could train our own model based on the totality of all COMET-user-verified data. This could be run without human intervention, or else with retraining via GreenML or GreenDDA as more data is collected.

Since most machine learning systems perform best with clean training data, we deem it advantageous to involve a human somewhere in the loop to check and correct generated data and to add missing data. Ideally, we could use active learning (Settles 2010) to target the most useful, informative interactions to present to the user for optimal annotation contribution, retraining models as needed to take advantage of this type of prioritization. If COMET is to be used to ensure that the data extracted from a book is complete and correct, some combination of green methods and active learning would behave like GreenFIE—one that would learn from a user's work and prepopulate subsequent pages with data in an attempt to reduce the user's workload.

3.7 OntoSoar: Extraction Rule Creation by Natural Language Processing and Cognitive Reasoning

OntoSoar (Lindes 2014; Lindes et al. 2015) extracts data using NLP techniques to discover and map extraction results from running text narrative. Its segmenter divides text like that in Figure 1 into sentences or subsentential fragments that are then pipelined to the Link Grammar parser (Temperley et al. n.d.). Figure 14 shows the parse of the fragment, "Mary Eliza Warner, b. 1826, dau. of Samuel Selden Warner and Azubah Tully;".

```
                                                 +---------------------------Xc--------------------------+
                        +-----------MX-----------+      +-----------------Js----------------+            |
    +---------Wf--------+--MX*p--+---Xca--+      |      +--------Js--------+                |            |
    |       +--G-+---G--+   +-Xd-+--IN-+  +--Xd--+--Mp--+    +---G--+---G--+          +--G--+            |
    |       |    |      |   |    |     |  |      |      |    |      |      |          |     |            |
LEFT-WALL Mary Eliza Warner , born.v 1826 , daughter.n of Samuel Selden Warner and Azubah Tully [;] RIGHT-WALL

Facts extracted:
  Reporting 7 objects:
    X2: Name(osmx85, "Mary Eliza Warner")
    X1: Daughter(osmx108)
    X1: Person(osmx108)
    X4: Name(osmx130, "Samuel Selden Warner")
    X3: Person(osmx137)
    X6: Name(osmx140, "Azubah Tully")
    X5: Person(osmx143)
    X7: BirthDate(osmx147, "1826")
  Reporting 6 relations:
    Y1(osmx150): Person(osmx108) has Name(osmx85)
    Y2(osmx153): Person(osmx137) has Name(osmx130)
    Y3(osmx156): Person(osmx143) has Name(osmx140)
    Y4(osmx159): Person(osmx108) was born on BirthDate(osmx147)
    Y7(osmx162): Daughter(osmx108) is a daughter of Person(osmx143)
    Y6(osmx165): Daughter(osmx108) is a daughter of Person(osmx137)
```

Figure 14: OntoSoar Syntax Analysis (top) and Semantic Analysis (bottom)

The Soar cognitive reasoner (Laird 2012) analyzes the parse, extracting salient semantic objects and relations from the relationships represented by the parse links. The Soar engine in our implementation has 240 production rules. These rules build meaning using ideas inspired by construction grammars, which (1) pair textual forms with meaning; (2) construct knowledge structures with inference rules; and (3) map knowledge structures to ontologies by comparing their common entities and relationships. The mapping provides a conduit for populating the ontological conceptualization in Figure 2 with data. Figure 14 shows the results of semantically analyzing the Mary Eliza Warner text chunk.

4 Ontology Integration

Given the results of applying our ensemble of extraction tools (Section 3), our next task is to integrate the results into a single populated ontology. This requires both schema integration and data integration. For our genealogy application, the ontology in Figure 15 is our target for schema integration. The data that populates the target ontology should be integrated so that the same objects and relationships extracted into the source ontologies are represented only once the target ontology.

4.1 Schema Integration

General schema integration is known to be a hard problem—denoted by some as "AI-complete" or essentially unsolvable (Marie and Gal 2007). The major bottleneck is automatically discovering matching schema components (Noy 2004; Rahm and Bernstein 2001). Schema matching usually requires algorithms to be multifaceted, meaning that several schema integration techniques are used together, and machine-learned, because of the complexity of successfully weighing the evidence gathered from the multiple facets (Embley et al. 2001; Xu 2003).

Despite these general difficulties, the typical schema matching needed for an ensemble of extraction engines, each with their own ontologies, can be much simpler. Matching algorithms for ontology-based reading systems have the advantage of being able to observe the extraction of the same objects and the same relationships into the various object and relationship sets in the ensemble's collection of ontologies. "Same" here means that the text being extracted comes from the same location in the document, and thus with the assurance that it actually denotes the same object.

As an example, consider integrating the *Family* ontology in Figure 9 into the integrated target

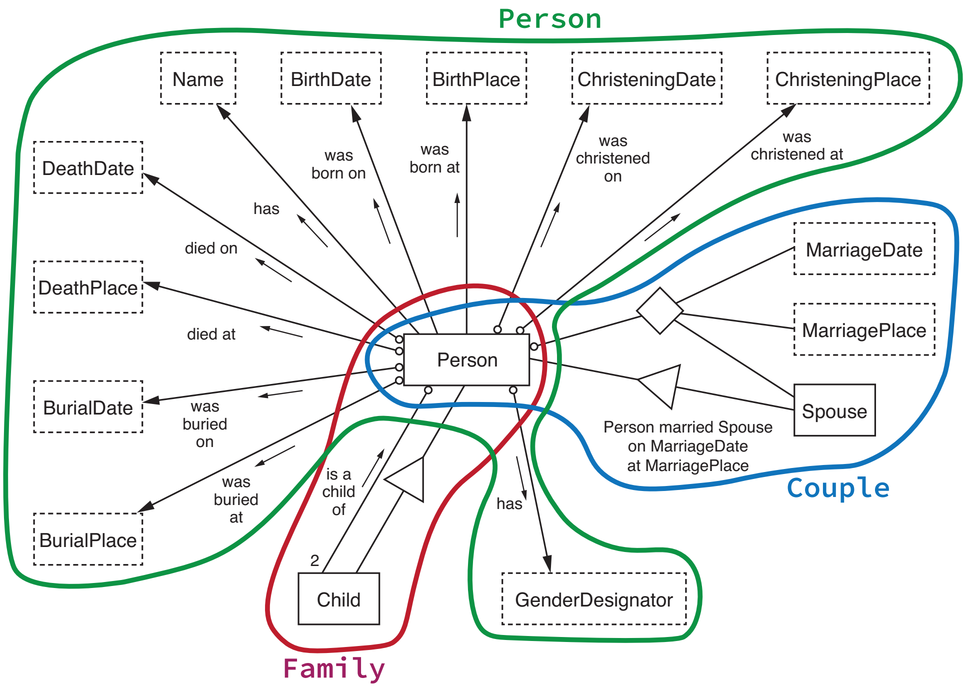


Figure 15: Integrated Target Extraction Ontology

extraction ontology in Figure 15. Note that except for *Child* none of the object set names match in the two ontologies. However, many of the same instances (e.g. "Charles Christopher Lathrop" and "Mary Augusta Andruss" in Figure 1) would be extracted into *Parent1* or *Parent2* in Figure 9 and into *Name* in Figure 2. Hence, *Parent1*, *Parent2*, and *Name* should all map to the same lexical object set in Figure 15, namely the *Name* object set. Similarly, *Child* should also map to the *Name* object set.

Schema integration need not be fully automatic, but we must know how to map the tool extraction ontologies to the common integrated target extraction ontology. Schema mappings for the *Family* ontology are outline above. For the *Couple* ontology *SpouseName* maps to *Name*. All other lexical object sets for our form ontologies (Figures 7, 8, and 9) map to lexical object sets with the same name. As the views superimposed on Figure 15 indicate, these three forms were designed to be complementary and to cover the integrated target extraction ontology. Except for the *Son* and *Daughter* components, the ontologies in Figures 2 and 5 map directly to the integrated extraction ontology in Figure 15. Sons and daughters are children, and in mapping them to *Child*, we also instantiate *GenderDesignator*s, "Son" and "Daughter" for them. For the ontology in Figure 3, we note that the integrated extraction ontology in Figure 15 does not have a breakdown of names and dates. Therefore, for each *BirthDate* and *DeathDate* object, we space-concatenate the strings in *Day*, *Month*, and *Year* as lexical *BirthDate* and *DeathDate* objects in Figure 15. Similarly, we space-concatenate *GivenName*s and the *Surname*. *ChildNr* is not in the integrated target extraction ontology, so we simply ignore it.

4.2 Data Integration

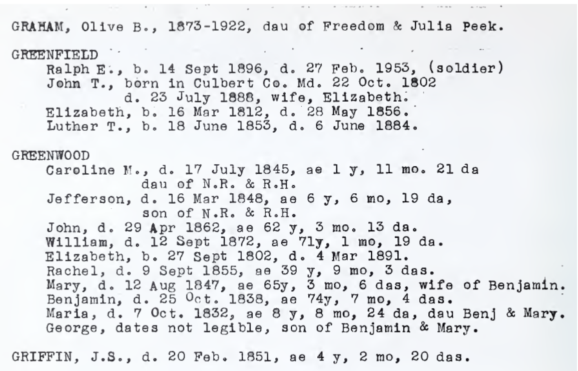
Lexical objects are the same if their text string is identical and is extracted from the same location in the document as determined by page and page-offset index values. Non-lexical objects are instantiated by ontological commitment with lexical objects. Thus, non-lexical objects are the same when their declared lexical instantiators are the same. Since the integrated target extraction ontology in Figure 15 has *Person* and specializations of *Person* as its only non-lexical object sets, and since we instantiate Person objects by an ontological commitment with *Name*, we discover duplicate non-lexical objects by determining whether they have the same name. Thus, the fundamental data-integration problem we must resolve is to determine whether two *Person* names denote the same object.

We declare names to be the same if and only if their text strings are substantially the same and are located at the substantially the same place in the document. Names that refer to the same object but are not located at the same place in the document are resolved later. These resolutions either require coreference reasoning to determine, for example, that "Mrs. Lathrop" refers to Mary Augusta Andruss in Figure 1, or they require object identity resolution to determine, for example, that three of the four mentions of the name "Mary Ely" in Figure 1 refer to Gerard Lathrop's wife while the fourth refers to Gerard Lathrop's granddaughter.

Although seemingly straightforward, determining "substantially the same" is often nontrivial. The extraction tools independently extract names; moreover, extraction rules within a single tool independently extract names. Often extracted names are identical, both offset and content, but there is no guarantee that a name in a document will be extracted in exactly the same way by all extraction rules within and across all tools.

Names can be extracted either as simple name strings or as complex name strings.

- Simple name strings comprise a single sequence of characters. Discrepencies can arise for several reasons: (a) One tool may extract name titles while another tool does not. In Figure 1, for example, one tool might extract "Judge Caleb Halstead Andruss" while another tool extracts "Caleb Halstead Andruss". (b) The tools are not consistent in the way they treat end-of-line hyphens. Some tools ignore them and thus would extract "Donald McKen" in Figure 1; some other tool may take the full name as given in the OCR text and extract "Donald McKen-\nzie"; and still some other tool may resolve the end-of-line hyphen and give the name as "Donald McKenzie". (c) Tools make mistakes and may extract "William Gerard Lathrop" in Figure 1, for example, as "William Gerard Lathrop, Boonton".
- Complex name strings comprise two or more name components that are not necessarily contiguous. Figure 16 shows some examples of names that do not consist of a single sequence of characters in the document: (a) "Freedom" "Peek" in the combined name "Freedom & Julia Peek"; (b) factored names such as "Ralph E." "GREENFIELD" and "John T." "GREENFIELD"; and (c) names such as "Benj" "GREENWOOD" and "Mary" "GREENWOOD", which are both factored and combined. Besides names, such as these, that are necessarily complex, tools may extract name strings that could be simple as complex name strings. Thus, for example, one tool may correctly extract the name that appears first in Figure 16 either as "GRAHAM, Olive B." or as "Olive" "B." "GRAHAM" or "Olive B." "GRAHAM". A tool may also incorrectly extract the name in several different ways, including as a partial name, "Olive" "GRAHAM", or an inferred name such as "Olive B." "Peek".



```
GRAHAM, Olive B., 1873-1922, dau of Freedom & Julia Peek.

GREENFIELD
    Ralph E., b. 14 Sept 1896, d. 27 Feb. 1953, (soldier)
    John T., born in Culbert Co. Md. 22 Oct. 1802
             d. 23 July 1888, wife, Elizabeth.
    Elizabeth, b. 16 Mar 1812, d. 28 May 1856.
    Luther T., b. 18 June 1853, d. 6 June 1884.

GREENWOOD
    Caroline M., d. 17 July 1845, ae 1 y, 11 mo. 21 da
             dau of N.R. & R.H.
    Jefferson, d. 16 Mar 1848, ae 6 y, 6 mo, 19 da,
             son of N.R. & R.H.
    John, d. 29 Apr 1862, ae 62 y, 3 mo. 13 da.
    William, d. 12 Sept 1872, ae 71y, 1 mo, 19 da.
    Elizabeth, b. 27 Sept 1802, d. 4 Mar 1891.
    Rachel, d. 9 Sept 1855, ae 39 y, 9 mo, 3 das.
    Mary, d. 12 Aug 1847, ae 65y, 3 mo, 6 das, wife of Benjamin.
    Benjamin, d. 25 Oct. 1838, ae 74y, 7 mo, 4 das.
    Maria, d. 7 Oct. 1832, ae 8 y, 8 mo, 24 da, dau Benj & Mary.
    George, dates not legible, son of Benjamin & Mary.

GRIFFIN, J.S., d. 20 Feb. 1851, ae 4 y, 2 mo, 20 das.
```

Figure 16: Part of a Page from Butler, County, Ohio, Cemetery Records (Stroup n.d.)

Determining whether two tool-extracted names are "substantially the same" is straightforward in the common case in which both the content and offset of all ordered name components match—and is otherwise anything but straightforward.

Heuristically, we determine whether two location-overlapping names match by first forming a single-string interpretations of the names. These single-string interpretations include a resolution of end-of-line hyphens, both those included in the extracted text and those not included, but only if they immediately follow any one of the extracted name components in the original document's text. Then, by checking both content and offset of each token of this single-string interpretation of the name, if one of these interpreted names subsumes the other, we declare a match. If the subsuming name has a recognized name form (currently, either a last-name-first form or a standard form with a sequence of names preceded optionally with titles and followed optionally with suffixes such as "Sr." and "Jr."), it becomes the interpreted name for the merged *Person* object. Otherwise, the subsumed name becomes the interpreted name, unless it also fails to have a standard name form, in which case we take the nonstandard subsuming name as the interpreted name.

This process iterates over all location-overlapping names by comparing each name with the current best name, and eventually forms an equivalence class of *Person* objects to be merged as a single object with a single best interpreted name. Examples: (a) The interpreted name for all extractions of "Donald McKen-\nzie", whether they include the hyphen or not, become "Donald McKenzie". (b) "Judge Caleb Halstead Andruss" is chosen to be the interpreted name over the subsumed "Caleb Halstead Andruss". Similarly, "Olive B. GRAHAM" is chosen over the subsumed "Olive GRAHAM". (c) The names "Olive B. GRAHAM" and "Olive B. Peek" do not match even though they presumably refer to the same person, and their respective person objects are not merged. (d) "William Gerard Lathrop, Boonton" subsumes "William Gerard Lathrop" but is not a proper name form and is therefore rejected in favor of the subsumed name "William Gerard Lathrop".

Once an equivalence class of *Person* objects has been formed, we next look for duplicate relationships connected to this equivalence class. Checking for duplicate relationships involves determining whether the text objects in corresponding lexical object sets are substantially the same. We determine "substantially the same" for these text objects in the same way we determine whether two names are "substantially the same" except that instead of checking for acceptable name forms, we check for acceptable textual forms for dates and place names. Duplicate relationships are then discarded and replaced with a single relationship whose lexical objects are the best among the possibilities.

5 Ontological Constraints

The OSM conceptual modeling language is grounded in a decidable restriction of first-order logic (Embley and Zitzelberger 2010). This formal grounding enables us to specify and check constraints.

5.1 Ontology Language Formalization

We formally define OSM-OL (OSM Ontology Language) as a triple (O, R, C) where O is a set of object sets, R is a set of relationship sets, and C is a set of constraints. Each object set in O is a one-place predicate, and each object-set predicate has either a *lexical* or a *non-lexical* designation. Instances of lexical object sets are strings (e.g. *DeathDate(Nov. 4, 1898)*, and instances of non-lexical object sets are object identifiers (e.g. *Person(osmx17)* and *Spouse(osmx17)*). Each relationship set in R is an n-place predicate ($n \geq 2$). We use a form of infix notation to specify instance relationships (e.g. *Person(osmx17) died on DeathDate(Nov. 4, 1898)*). C is a set of constraints:

- *Referential*: object instances referenced in relationship instances must exist in referenced object sets, e.g.

$$\begin{aligned}&\mathit{Person}(\mathit{osmx17})\ \mathit{died\ on}\\ &\quad \mathit{DeathDate}(\mathit{Nov.\ 4,\ 1898}) \Rightarrow\\ &\qquad \mathit{Person}(\mathit{osmx17})\\ &\qquad \wedge\ \mathit{DeathDate}(\mathit{Nov.\ 4,\ 1898})\end{aligned}$$

- *Participation*: instances in an object set S must participate in relationships in a relationship set R connected to S according to declared participation constraints, e.g.

  $$\forall x(\mathit{Child}(x) \Rightarrow \exists^2 y(\mathit{Child}(x)\ \mathit{is\ a\ child\ of}\ \mathit{Person}(y)))$$

  specifies that every child has exactly two parents.
- *Generalization/specialization*: instances in a specialization S of a generalization G must exist in G, e.g. $\forall x(\mathit{Spouse}(x) \Rightarrow \mathit{Person}(x))$.
- *General*: any predicate-calculus-specified constraint, e.g.

  $$\forall x \forall y \forall z(\mathit{Person}(x)\ \mathit{was\ born\ on}\ \mathit{BirthDate}(y) \wedge (\mathit{Person}(x)\ \mathit{died\ on}\ \mathit{DeathDate}(z)) \Rightarrow y \leq z$$

  specifies that a person's death date must not precede the person's birth date.

Note that all but general constraints can be specified in OSM's graphical notation. Nevertheless, all constraints are just predicate calculus statements. Note also that the graphical black-triangle aggregation symbol has no associated constraint because aggregation is merely a visual grouping of relationship sets devoid of other formal meaning.

The constraints mentioned so far are all crisp, yielding only a strict "yes" or "no" to determine violations. We call these constraints *hard*. Ontologies that seek to model reality, however, should also provide for *soft* constraints. By introducing probability distributions, we can allow constraints to return the probability that an assertion holds, and thus have soft as well as hard constraints. For participation constraints, instead of a crisp *min*:*max* designation, we can return the probability of the actual cardinality as asserted. Thus, for example, based on the probability distribution, we can determine how reasonable it is that a mother has 2 children (or 17 or 209). With extended general constraints, we can check the sensibility of many assertions such as whether the age of a mother is reasonable for having a child or whether the age difference between spouses is common for the time and place of their marriage.

Referential-integrity constraints and isa constraints in generalization/specialization hierarchies should not be extended to allow for uncertainty. The model itself would not make sense if objects referenced in relationships do not exist or if objects in specializations are not also in their generalizations (e.g. if an individual is a *Child* or a *Spouse* but not also a *Person*).

5.2 Constraint Checking

Given an ontological model of our world of interest, we can check extracted assertions against this model to see if they make sense. Unlike standard databases, which only allow updates if no constraints are violated, we allow our extraction tools to populate an ontology independent of whether they violate hard constraints or whether they are unreasonable with respect to soft constraints. We then determine whether the extracted assertions make sense with respect to the constraints of the ontological world of interest (Woodfield et al. 2016).

Figure 17 shows a soft constraint written in the OSM ontology language of the integrated target extraction ontology in Figure 15. It returns the probability of a child having been born to a mother at a particular age. The first three antecedent statements are predicates that come directly from the ontology in Figure 15. The fourth implicitly adds an object set *Gender* and a relationship set *Person has Gender* to the ontology. We populate the implicit object and relationship set with instances determined from information at hand: (1) If *Person*(x) has a *GenderDesignator*, we immediately know the *Gender*. (2) If not, but the *Person* is married and the spouse's gender is known, the person's *Gender* is known. (3) If still unknown, the first given name maps to the probability of the person being male or female in a large frequency table created by running over the billion or so name/gender pairs in the *Family Tree*. If the probability is above a specified threshold (currently set at 0.95), we can confidently set the gender. (4) Finally, if still unknown, we leave the gender unknown. The fifth antecedent statement makes use of the data-frame declared internal representation and operators for dates. The final antecedent statement references a probability distribution, which we are able to compute over the many millions of mother-child relationships in the *Family Tree*. The consequent statement yields a relationship for an implicit quaternary relationship set that connects the object sets *Person* and *Child*, which are already in the ontology, and *Age* and *Probability*, which implicitly belong to the ontology.

Child(x_1) *is a child of Person*(x_2)

Person(x_1) *was born on BirthDate*(x_3)

Person(x_2) *was born on BirthDate*(x_4)

Person(x_2) *has Gender*(*Female*)

Age(x_5) = *Age*(*YearOf*(x_4) − *YearOf*(x_3))

mother's Age(x_5) *at child's birth has Probability*(x_6)

⇒

Person(x_2)*'s Age*(x_5) *at Child*(x_1)*'s birth has Probability*(x_6)

Figure 17: Probability of a Mother's Age at Her Child's Birth being Reasonable

5.3 Constraint Violation Resolution

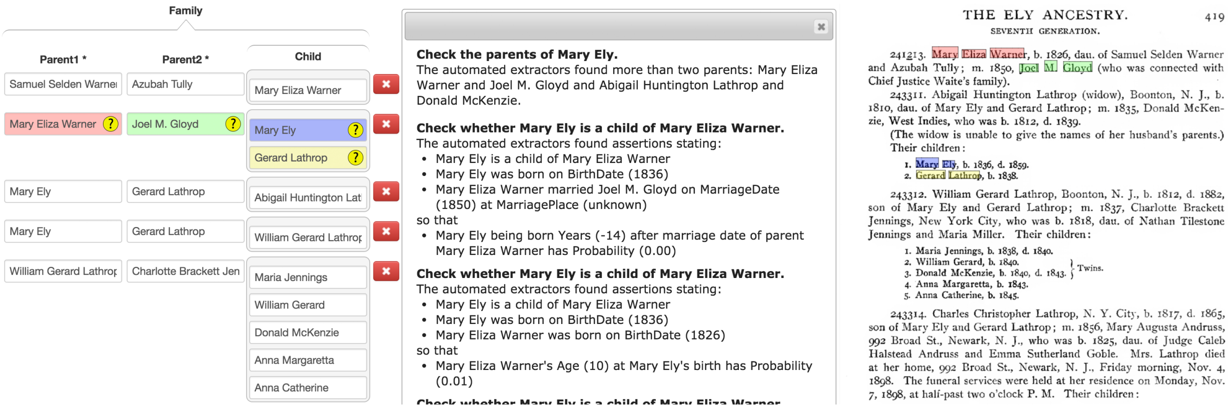
The process of detecting and correcting or removing inaccurate information in a data repository is known as data cleaning (Müller and Freytag 2003; Rahm and Do 2000). For our reading system, data cleaning consists of observing constraint violations and resolving them. Given the results of constraint checking, the reading system can sometimes correct itself. Most often, however, constraint violation resolution requires a human in the loop.

Figure 18 shows the interface a human uses to resolve constraint violations. Hovering over a record highlights its fields and the information in the text document filled into the various record fields. It also marks fields identified as possibly being in error with a red, yellow, or green warning icon, depending on the severity of the constraint violation—red when something is definitely wrong such as a death date preceding a birth date, yellow when something is likely wrong, and green when something is likely right, but should be double-checked. Often, the highlighting is sufficient to indicate the error. In Figure 18 a user can easily see that Mary and Gerard are not children of Joel and Mary Gloyd and should click on the corresponding red-x button to remove the highlighted record.

If a user clicks on a warning icon, an explanation box pops up. In Figure 18, the user has clicked on the warning icon in the field filled in with "Mary Ely". Three messages apply: (1) Mary Ely has too many parents, (2) Mary was born 14 years before her presumed parents Mary Eliza Warner and Joel M. Gloyd were married, and (3) Mary Ely's presumed mother Mary Eliza Warner would have been only 10 years old when Mary Ely was born.

All our soft general constraints are implication statements. When implication statements are violated, one or more of the antecedent statements must be incorrect. The suspect antecedent statements are those corresponding to assertions stored in the ontology. Explanations, such as those in Figure 18, list these statements as possibly being the cause of a constraint violation. Note that we can generate these human readable explanation statements from the rule itself by replacing references to non-lexical objects by their ontologically committed lexical counterparts (person names in our application) and by replacing references to lexical objects by placing the lexical values in parentheses following the object set name. In Figure 18, the asserted antecedent statement declaring that Mary Ely is a child of Mary Eliza Warner is wrong.

Automatic retraction of assertions is sometimes possible. When an implication rule has only one antecedent statement corresponding to an assertion stored in the ontology, the assertion can safely be retracted. Although not quite as safe,



Family

Parent1 *	Parent2 *	Child	
Samuel Selden Warner	Azubah Tully	Mary Eliza Warner	✖
Mary Eliza Warner ?	Joel M. Gloyd ?	Mary Ely ? Gerard Lathrop ?	✖
Mary Ely	Gerard Lathrop	Abigail Huntington Lat	✖
Mary Ely	Gerard Lathrop	William Gerard Lathrop	✖
William Gerard Lathrop	Charlotte Brackett Jen	Maria Jennings William Gerard Donald McKenzie Anna Margaretta Anna Catherine	✖

Check the parents of Mary Ely.
The automated extractors found more than two parents: Mary Eliza Warner and Joel M. Gloyd and Abigail Huntington Lathrop and Donald McKenzie.

Check whether Mary Ely is a child of Mary Eliza Warner.
The automated extractors found assertions stating:
- Mary Ely is a child of Mary Eliza Warner
- Mary Ely was born on BirthDate (1836)
- Mary Eliza Warner married Joel M. Gloyd on MarriageDate (1850) at MarriagePlace (unknown)

so that
- Mary Ely being born Years (-14) after marriage date of parent Mary Eliza Warner has Probability (0.00)

Check whether Mary Ely is a child of Mary Eliza Warner.
The automated extractors found assertions stating:
- Mary Ely is a child of Mary Eliza Warner
- Mary Ely was born on BirthDate (1836)
- Mary Eliza Warner was born on BirthDate (1826)

so that
- Mary Eliza Warner's Age (10) at Mary Ely's birth has Probability (0.01)

Check whether Mary Ely is a child of Mary Eliza Warner

THE ELY ANCESTRY. 419
SEVENTH GENERATION.

241213. Mary Eliza Warner, b. 1826, dau. of Samuel Selden Warner and Azubah Tully; m. 1850, Joel M. Gloyd (who was connected with Chief Justice Waite's family).

243311. Abigail Huntington Lathrop (widow), Boonton, N. J., b. 1810, dau. of Mary Ely and Gerard Lathrop; m. 1835, Donald McKenzie, West Indies, who was b. 1812, d. 1839.

(The widow is unable to give the names of her husband's parents.)
Their children:

1. Mary Ely, b. 1836, d. 1859.
2. Gerard Lathrop, b. 1838.

243312. William Gerard Lathrop, Boonton, N. J., b. 1812, d. 1882, son of Mary Ely and Gerard Lathrop; m. 1837, Charlotte Brackett Jennings, New York City, who was b. 1818, dau. of Nathan Tilestone Jennings and Maria Miller. Their children:

1. Maria Jennings, b. 1838, d. 1840.
2. William Gerard, b. 1840. } Twins.
3. Donald McKenzie, b. 1840, d. 1843. } Twins.
4. Anna Margaretta, b. 1843.
5. Anna Catherine, b. 1845.

243314. Charles Christopher Lathrop, N. Y. City, b. 1817, d. 1865, son of Mary Ely and Gerard Lathrop; m. 1856, Mary Augusta Andruss, 992 Broad St., Newark, N. J., who was b. 1825, dau. of Judge Caleb Halstead Andruss and Emma Sutherland Goble. Mrs. Lathrop died at her home, 992 Broad St., Newark, N. J., Friday morning, Nov. 4, 1898. The funeral services were held at her residence on Monday, Nov. 7, 1898, at half-past two o'clock P. M. Their children:

Figure 18: Error Warning Indicators and Explanations

a single antecedent assertion in the intersection of multiple rule violations is almost certain to be wrong and can be retracted. In Figure 18, for example, the antecedent assertion *Person(Mary Ely) is a child of Person(Mary Eliza Warner)* is in both the born-much-earlier-than-marriage violation and the mother-too-young-to-give birth violation and should be retracted. Sometimes, we can heuristically determine which assertion(s) should be retracted. We have observed, for example, that when a child has too many parents and all the parents precede the child in the document flow, the closest couple or single parent is correct, and parent-child assertions for all other parents can be retracted. In the document in Figure 18 the four mentioned parents all precede Mary Ely, and her correct parents, Donald and Abigail McKenzie, are closer to Mary than her incorrect parents, Joel and Mary Gloyd.

When a human is in the loop to check and correct extracted assertions, the system automatically retracts identifiably incorrect assertions before presenting results for an initial check-and-correct session. Warning icons are also initially omitted, which lets users to do unbiased checking and correcting and avoids overwhelming them with largely obvious and often extraneous statements about what might be wrong. However, if constraint violations remain after the initial human check-and-correct session, no retraction takes place, icons are added as a warning that something has likely been overlooked, and the document is returned to the user for further checking.

6 Ontological Inference

Authors of factual documents often convey information by implication and expect readers to infer these facts by what is explicitly stated. In Figure 1, for example, there are several implications of interest about person names. Maria Jennings' maiden name is Maria Jennings Lathrop, the surname being added by implication based on cultural norms. Abigail Huntington Lathrop would have been known in her married life as Abigail McKenzie since she was female (not stated, but determined by implication) and was married to Donald McKenzie. The author does not sort out the identity of the persons named "Mary Ely" in Figure 1, but leaves it to the reader to determine that there is one person named Mary Ely who is married to Donald Lathrop and another who is her granddaughter.

Reading systems, like human readers, need to be able to "read between the lines" and infer implied information (Embley et al. 2016). Reading systems should be able to infer new objects and relationships, placing them in potentially new ontological object and relationship sets. They should also be able to resolve object identity and determine which non-lexical object identifiers denote the same object.

6.1 Infer New Objects and Relationships

Because OSM extraction ontologies are formally grounded in first-order logic, it is natural to infer new objects and relationships as Datalog-like queries (Datalog User Manual 2004; Gallaire and Minker 1978). Although we have used Datalog directly to derive information (Embley et al. 2016; Park 2015), we currently program the equivalent of Datalog queries to infer the specific implied information we seek.

Figure 19 shows our target extraction ontology extended with four new object sets: *InferredGender*, *InferredBirthName*, *InferredMarriedName*, and *InferredFormalName*, which is an aggregate of one or more *GivenName*s and *Surname*s, and zero or more *Title*s and *Suffix*es. The reading system populates these new object sets and their connected relationship sets by inference.

In our application we first infer inverses of persons and their spouses. If *Person(x) married Spouse(y) on MarriageDate(z) at MarriagePlace(w)*, then *Person(y) married Spouse(x) on MarriageDate(z) at MarriagePlace(w)*. No new object or relationship sets are created but new facts not directly stated are added. From Figure 1 the extraction engines would have read that Mary Eliza Warner married Joel M. Gloyd, which implies also that Joel M. Gloyd married Mary Eliza Warner.

Next we obtain *InferredGender* as explained earlier. Extracted *GenderDesignator*s are converted into their appropriate gender values. For persons without *GenderDesignator*s, the system infers gender by first given names when the certainty is sufficiently high, relying also on spouse gender for married persons. Knowing the gender is particularly important for inferring names in English-speaking western cultures.

Before populating inferred name object sets, we first standardize all names. At this point in our processing pipeline, we have both the original text (as extracted) and the interpreted text (a single string of space-separated components in which components with end-of-line hyphens have been closed up). Standardized text is a third representation of all lexical objects. For names, we standardize each name component with upper- and lower-case letters and order the components respectively by titles (if any), given names, surname, and suffixes (if any). Thus, for example, the name “ALBRIGHT, ESTHER R.” in Figure 20 becomes “Esther R. Albright” in its standardized form.

With names in standard form, we can determine the surname of fathers and husbands and, using familial relationships, reason about birth names and married names. From the information in Figure 20, for example, we can determine that “Esther R. Albright” is a married name since she is female and married to Winfield S. Albright. We can also determine that Esther’s maiden name is Esther R. Morris since her father is Thomas Benton Morris. An *InferredFormalName* is a person’s birth name with titles and suffixes (if any) attached and with the surname extended with additional married surnames (if any). Thus, Esther’s *InferredFormalName* is “Esther R. Morris Albright” where both “Morris” and “Albright” are surnames.

6.2 Infer Object Identity

One way to infer object identity is by coreference resolution. Often several linguistic expressions (e.g. noun phrases, proper nouns and pronouns) in a text may refer to the same entity under discussion. In Figure 1, the expression “Mrs. Lathrop” corefers to the person previously mentioned as “Mary Augusta Andruss.” Finding and resolving instances of coreference is a difficult NLP problem. A wide range of knowledge sources, usually in combination, serve in systems that process text and posit corefence: morpholexical features such as person and number agreement; syntactic configurational relationships like apposition and binding; semantic properties like scope, modals and negation; pragmatic properties like animacy and gender; text-related properties like register and stylistics; and memory-related functions like recency, distance, and repetition. Rule-based implementations have existed for some time now (e.g. (Grosz et al. 1995)), but more current work

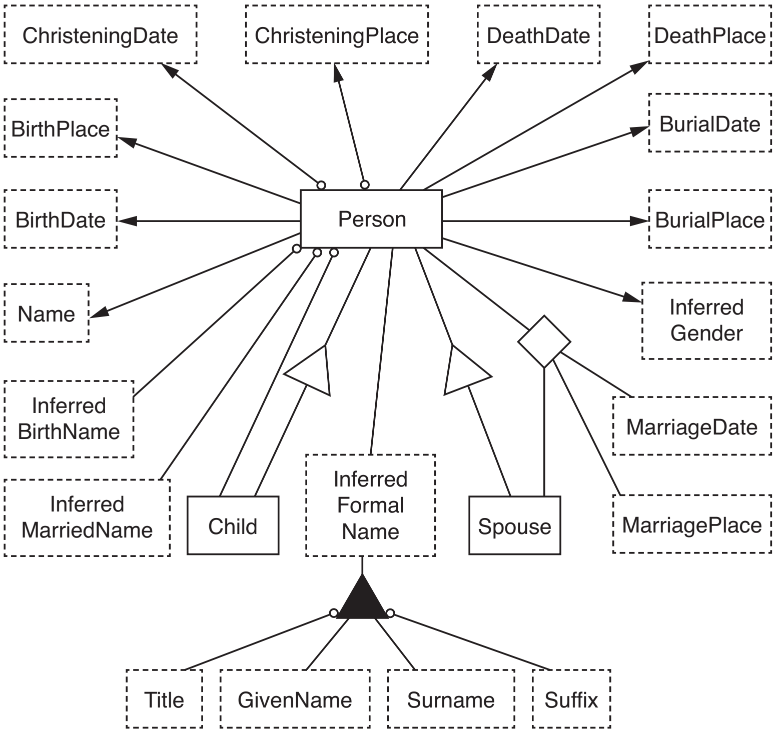


Figure 19: Ontology with Inferred Object and Relationship Sets

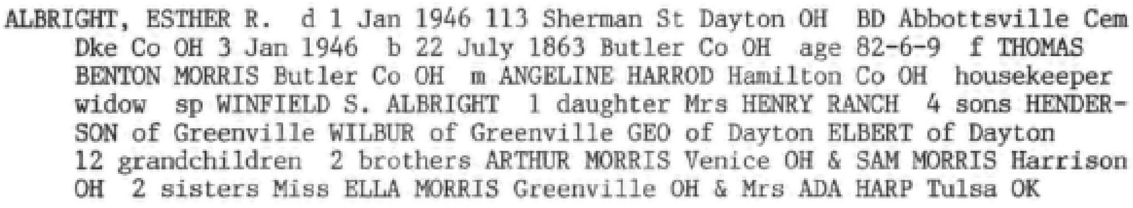

ALBRIGHT, ESTHER R. d 1 Jan 1946 113 Sherman St Dayton OH BD Abbottsville Cem
Dke Co OH 3 Jan 1946 b 22 July 1863 Butler Co OH age 82-6-9 f THOMAS
BENTON MORRIS Butler Co OH m ANGELINE HARROD Hamilton Co OH housekeeper
widow sp WINFIELD S. ALBRIGHT 1 daughter Mrs HENRY RANCH 4 sons HENDER-
SON of Greenville WILBUR of Greenville GEO of Dayton ELBERT of Dayton
12 grandchildren 2 brothers ARTHUR MORRIS Venice OH & SAM MORRIS Harrison
OH 2 sisters Miss ELLA MORRIS Greenville OH & Mrs ADA HARP Tulsa OK

Figure 20: Esther Albright Funeral Home Record (Miller Funeral Home Records, 1917 – 1950, Greenville, Ohio 1990)

also focuses on statistical, machine learning (Recasens and Hovy 2009), and deep learning solutions (Clark and Manning 2016).

Another way to infer object identity is by matching object properties and relationships with other objects (Benjelloun et al. 2009; Bhattacharya and Getoor 2007; Elmagarmid et al. 2007). In Figure 1, for example, our reading system will have extracted four different persons with the name "Mary Ely" and will have also extracted several related items of information for each of them. Three of the Mary Ely's have a spouse named Gerard Lathrop, albeit each with a different child: Abigail Huntington Lathrop, William Gerard Lathrop, and Charles Christopher Lathrop. The fourth Mary Ely has a birth year, 1836, a death year, 1859, a father, Donald McKenzie, and a mother, Abigail Huntington Lathrop (who by earlier data integration is known to be the same person as the first Mary Ely's daughter). This information is sufficient for Duke (Duke: Fast Deduplication Engine n.d.), an off-the-shelf entity-resolution engine, to conclude that the three Mary Ely's married to a Gerard Lathrop are all the same person and are not the same person as the Mary Ely who is the daughter of Abigail Huntington Lathrop.

In our application, we have yet one more way we can resolve object identity. At FamilySearch.org we are building a unified *Family Tree*, a public wiki-like, shared repository of interconnected family genealogies (FamilySearch n.d.). Sometimes the persons for whom we are extracting information are already on the *Family Tree*. In this case we can find potential matches in the tree, gather related information from both the *Family Tree* and the document being read, and present it for consideration for resolving object identity. Figure 21 shows an example in a D-Dupe-like view of the information (Kang et al. 2008). In the *Person* object set on the left are the four Mary Ely's under consideration for merging along with their attribute values (if any). In the *Person* object set on the right are two possible matches in the *Family Tree*. One-hop relationships for all these persons are are also found and displayed. When related persons are also found to be possible duplicates, they are grouped together. Those that have matches between the document being read and the *Family Tree* appear in the middle. The evidence in Figure 21 is even more persuasive for the merge of the first three Mary Ely's than the evidence in the document alone. Moreover, the evidence also argues for a merge within the *Family Tree* of Mary Ely (KFRL-WXZ) and Mary Eli (MGV1-9BJ).

7 Ontology Construction

Reading to construct ontologies is perhaps the most complex aspect of automated reading systems (Cimiano 2006; Cimiano et al. 2006; Wong et al. 2012). Nevertheless, for some special cases, we can augment or integrate ontologies in our collection, and with some restrictive types of input documents, we can sometimes construct ontologies from the text itself.

7.1 Connect and Augment Existing Conceptualizations

Named entity recognition (NER) systems, for example the Stanford CoreNLP machine learning annotator (Finkel et al. 2005), identify references to entities in text. They flag and categorize proper noun expressions as referring to such objects as persons, locations, organizations, and time expressions (Nadeau and Sekine 2007). Using the principle of ontological commitment, we use tagged entities from NER output to populate an ontology snippet consisting of a non-lexical object set named by the entity type connected to a lexical object set giving the entity's designating name. Figure 22(a) shows an example for *Location*.

Running over the text in Figure 1, a *Location* entity recognizer would populate the ontology in Figure 22(a) with entities for "West Indies", "Boonton N. J.", "N. Y. City", "Newark, N. J.", "New Jersey", and "Elizabethtown". Now, notice that an NLP system would be able to discover that the first three of these locations are related respectively to Donald McKenzie, William Gerard Lathrop, and Charles Christopher Lathrop. Hence, we can create and populate a relationship set connecting *Location* in Figure 22(a) and *Person*, the

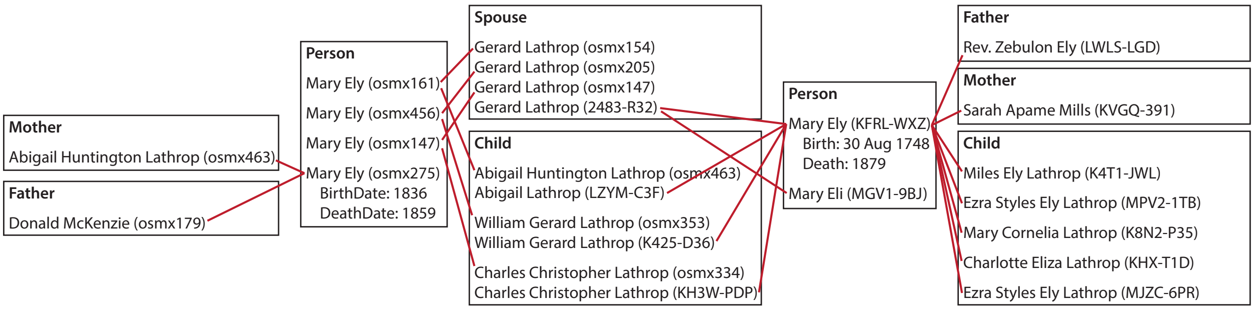

Mother
Abigail Huntington Lathrop (osmx463)
Father
Donald McKenzie (osmx179)
Person
Mary Ely (osmx161)
Mary Ely (osmx456)
Mary Ely (osmx147)
Mary Ely (osmx275)
BirthDate: 1836
DeathDate: 1859
Spouse
Gerard Lathrop (osmx154)
Gerard Lathrop (osmx205)
Gerard Lathrop (osmx147)
Gerard Lathrop (2483-R32)
Child
Abigail Huntington Lathrop (osmx463)
Abigail Lathrop (LZYM-C3F)
William Gerard Lathrop (osmx353)
William Gerard Lathrop (K425-D36)
Charles Christopher Lathrop (osmx334)
Charles Christopher Lathrop (KH3W-PDP)
Person
Mary Ely (KFRL-WXZ)
Birth: 30 Aug 1748
Death: 1879
Mary Eli (MGV1-9BJ)
Father
Rev. Zebulon Ely (LWLS-LGD)
Mother
Sarah Apame Mills (KVGQ-391)
Child
Miles Ely Lathrop (K4T1-JWL)
Ezra Styles Ely Lathrop (MPV2-1TB)
Mary Cornelia Lathrop (K8N2-P35)
Charlotte Eliza Lathrop (KHX-T1D)
Ezra Styles Ely Lathrop (MJZC-6PR)

Figure 21: Duplicate Detection and Resolution

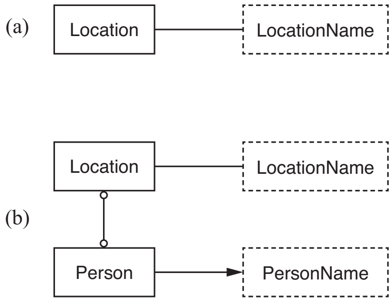


Figure 22: Location Ontology

primary object set in the ontologies we have been discussing. Not knowing what the relationship is, we set the constraints to be as loose as possible: many-many and optional on both connections. Figure 22(b) shows the result.

Similarly, we can connect the *Organization* ontology snippet in Figure 23(a) with the *Person* object set based on the sentences linking Emma Goble to organizations in the paragraph about her in Figure 1. We may even be able to do better: (1) we may already have a more extensive *Organization* ontology with object sets *Member*, *MemberName*, and *Officer* as a specialization of *Member* and instantiated by identifying the *Member* and *Office* in the *Organization*; or (2) perhaps by a more complex NLP analysis of the sentences in the paragraph about Emma Goble, we could construct such an ontology. Then, after populating both our *Person* ontology and the *Organization* ontology and by anaphoric reasoning discovering that all the references to Emma Goble refer to the same person, we can integrate the two ontologies as Figure 23(b) shows.

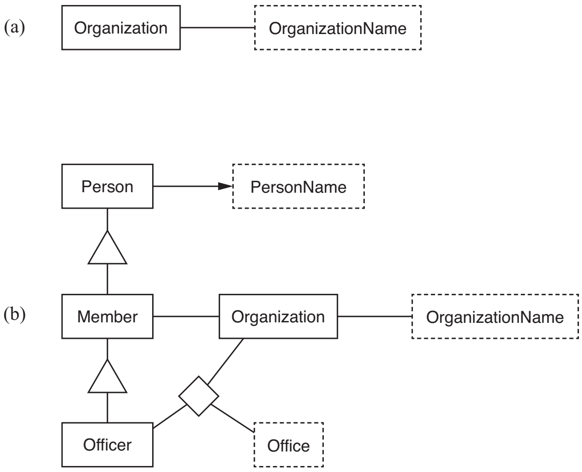


Figure 23: Organization Ontology

7.2 Read Semi-structured Text to Create Conceptualizations

TANGO (Table ANalysis for Generating Ontologies) is a methodology we have proposed as a means of automatically deriving an ontology from a correlated collection of standard tables (Tijerino et al. 2005). In essence, the idea is to find a correlated collection of ordinary tables, reverse-engineer them into conceptual models, and integrate them into a single conceptual model—an ontology that represents their union. The tables in a relational database, which are by definition correlated and furthermore have often been created from a conceptual model, are particularly amenable to TANGO processing.

As an example, we illustrate how we can use the TANGO methodology to derive an ontology starting from a document, which we assume is semi-structured like the document in Figure 1. As Figure 13 shows, ListReader (Packer 2014) can find text patterns in semi-structured text that have record-like structure. When using ListReader for information extraction, a user maps a selected text record to a given form to establish the connection between the components of the text record and the fields of a form, which thus also establishes the mapping of a text record to an ontology. When reading to create an ontology, however, there is no form and no ontology. In this case, a user can turn a ListReader-discovered list of text records into a relational table by associating user-chosen attribute names with ListReader-generated abstract text components. In Figure 13, for example, a user can associate *Name* with `[UpLo+][Sp][UpLo+]` and *BirthDate* with `[Dg][Dg][Dg][Dg]`. Stripping away the text in the records that does not associate with any of the attributes yields a relational database table. To name the objects the table represents, the user provides a name for the table—*Person* for our example. Figure 24(a) shows the result for Figure 13.

Reverse engineering the relational table in Figure 24(a) yields the snippet of the ontology in Figure 7 consisting of just the object sets *Person*, *Name*, and *BirthDate*. Other ListReader-discovered text record patterns in Figure 1 and elsewhere in *The Ely Ancestry* would lead to the relational tables *Person*(*Name, GenderDesignator*) and *Person*(*Name, DeathDate*). These relational tables along with other *Person*-property tables that include christenings, burials, and place information for births and deaths integrate trivially and can be reverse-engineered into the ontology in Figure 7. Obtaining the finer properties of the ontologies such as participation constraints can only be done heuristically. In practice, a knowledgeable user should check and, as necessary, adjust these finer properties of generated ontologies.

Although beyond the capabilities of ListReader's current implementation, an extension of ListReader's pattern discovery mechanism could potentially generate nested database relations like those in Figures 24(b) and 24(c). (Currently handwritten or GreenFIE-generated regular-expression rules can extract relations with these nested patterns.) Reverse engineering the nested relations in Figures 24(b) and 24(c) respectively yields the ontologies in Figure 8 and Figure 9. Integrating these ontologies and the ontology in Figure 7 yields the ontology in Figure 15.

7.3 Construct Conceptualizations from Text Specifications

Generating a natural language description of an OSM ontology is straightforward. We simply write down each relationship-set name and each generalization/specialization *is-a* connection in some arbitrary order. Figure 25(a) shows a sampling of sentences describing the ontology in Figure 2. Other researchers have also studied verbalization techniques for conceptual models with the goal of making it easier, for example, to validate models with domain experts. There are interesting challenges in creating high-quality verbalizations, but the approaches are generally relatively straightforward (Curland and Halpin 2012; Halpin 2004; Halpin and Curland 2006). Reversing the process, however, to generate an ontology from ordinary prose is non-trivial.

Although not quite ordinary natural language prose, we have developed a model-equivalent specification language for OSM conceptual models (Embley 1998; Liddle et al. 1995). Model-equivalent languages provide a way to specify ontologies using natural-language-like statements, but are written according to a restricted context-sensitive grammar. Figure 25(b) gives an example. Object set names are nouns and are capitalized. Verb phrases, which are written in lower-case words, connect nouns to form sentences. Sentences define relationship sets. Constraints appear in square brackets. Participation constraints for object-set/relationship-set connections appear in sentences in square brackets as Figure 25 shows. Generalization/specialization hierarchies are written as "is-a" sentences with the specialization as the subject and the generalization as the object in the sentence.

```
Person(Name,           BirthDate)
       -------------------------
       ...
       Gerard Lathrop  1838
       William Gerard  1840
       Anna Margaretta 1843
       Anna Catherine  1845
       Emma Goble      1862
       ...
```

(a) *Relational Table Obtained from the ListReader-Captured Data in Figure 13*

```
Couple(Person,               (SpouseName,                 MarriageDate,  MarriagePlace))
       -------------------------------------------------------------------------------
       Thomas Patterson       Jane Clark
       Ben Ezra Stiles Ely    Elizabeth Eudora McElroy    1848
                              Abbie Amelia Moore          1873
       Harriet Clarissima Ely Beale Steenberger Blackford
       Zebulon DeForest Ely   Clara Vanola Major          1874
                              Mamie Anna Souder           1878
```

(b) *Nested Relational Table of Couple Information from Page 421 of The Ely Ancestry*

```
Family(Parent1,                   Parent2,                   (Child                     ))
       ---------------------------------------------------------------------------------
       Samuel Selden Warner       Azubah Tully                Mary Eliza Warner
       Mary Ely                   Gerard Lathrop              Abigail Huntington Lathrop
       Abigail Huntington Lathrop Donald McKenzie             Mary Ely
                                                              GerardLathrop
       Mary Ely                   Gerard Lathrop              William Gerard Lathrop
       William Gerard Lathrop     Charlotte Brackett Jennings Maria Jennings
                                                              William Gerard
                                                              Donald McKenzie
                                                              Anna Margaretta
                                                              Anna Catherine
        Nathan Tilestone Jennings Maria Miller                Charlotte Bracket Jennings
        ...
```

(c) *Nested Relational Table of Family Information from Figure 1*

Figure 24: Relational Database Tables Created from Text Patterns Found on Page 419 and 421 of The Ely Ancestry

```
Person has Name.
Person was born on BirthDate.
Person was born at BirthPlace.
Child is a Person.
Child is a child of Person.
Spouse is a Person.
Person married Spouse on MarriageDate at MarriagePlace.
...
```

(a) *Natural Language Rendition of the Ontology in Figure 2 (partial).*

```
Person[1] has Name[1:*].
Person[1] was born on BirthDate[1:*].
Person[1] was born at BirthPlace[1:*].
Child is-a Person.
Child[2] is a child of Person[0:*].
Spouse is-a Person.
Person[0:*] married Spouse[1:*] on MarriageDate[1:*] at MarriagePlace[1:*].
...
```

(b) *Textual Specification of the Ontology in Figure 2 (partial).*

Figure 25: Ontologies Rendered and Specified in Natural Language

Many researchers have studied the challenge of moving from textual descriptions to corresponding conceptual models or ontologies. For example, Chen studied a number of the foundational issues associated with moving from English natural-language specifications to ER diagrams (Chen 1983). Cimiano et al. have proposed numerous techniques for constructing ontologies from text (Buitelaar et al. 2008; Cimiano 2006; Cimiano and Völker 2005; Cimiano et al. 2006). Notably, Heinrich Mayr and his colleagues have pursued a long line of research into how to better correlate natural-language statements of requirements specifications to conceptual models that can support information systems development (Fliedl et al. 2003, 2005, 2007, 2004; Kop et al. 2004; Mayr and Kop 1998). KCPM, the Klagenfurt Conceptual Pre-design Model, supports a "conceptual pre-design" step in the software development lifecycle in order to bridge the historical "impedance mismatch" between requirements analysis and conceptual design. Mayr et al. provide semi-automatic techniques for mapping from natural-language requirements specifications to pre-design schemas, and from pre-design schemas to conceptual schemas and domain ontologies.

We anticipate that researchers will continue to work on this interesting and complex challenge for decades to come.

8 Project Status

In the domain of family history and in cooperation with FamilySearch International (FamilySearch n.d.), we have implemented a genealogical document reading system. We give here the implementation status of this system including the status of the extraction tools and of the pipeline that integrates the information received from the extraction engines, checks it with respect to ontological constraints, standardizes it, and infers new additional information not directly extractable from the document. We also discuss our initial experience with preparing and importing the information obtained by the reading system into the *Family Tree*. Figure 26 shows the top-level interface to our project, from which an administrator can manage the reading system. The menu on the left allows the administrator to import a new book to be read, configure and test extraction tools, and manage the process of reading the book and generating GedcomX files (Gedcom X n.d.) for import into the *Family Tree*.

Besides our near-term objectives within FamilySearch, we also envision applying our reading system to construct, populate, and query a *Web of Knowledge* (*WoK*) for any domain of interest and thus also for a collection of overlapping and non-overlapping domains of interest. We report briefly on the status of this effort and particularly on the use of reading systems to "understand" queries and map them to formal queries over a WoK.

8.1 Extraction Tools

Figure 27 shows the extraction tools within the scope of our project categorized by methodology type and ordered by the amount of human effort required to configure the tools to read a document. Wanting to cover the space of document types from those that are highly structured (e.g. the document in Figure 28) to those with free-running unstructured text (e.g. Figure 29) to everything in between (e.g. Figure 1) prompted us to develop tools in a variety of methodology paradigms. Within these paradigms our research efforts are directed toward reducing human involvement—generating expert system rules from examples and discovered text patterns, learning how to map parsed sentences to ontologies, and reducing the amount of training data needed for machine learning.

In Figure 27, tools in red (FROntIER, OntoES, GreenFIE, ListReader, OntoSoar) have been developed and evaluated as academic prototypes. Below, we give evaluation results of these tools and assess their strengths and weaknesses. Tools in green (GreenQQ, GreenML, GreenDDA, OntoSoar2) are academic prototypes in development. Below, we briefly mention our expectations for these tools. In violet are language machine learning paradigms. We may consider adopting or adapting some of the tools developed in these areas for our reading system. We have yet to do a tech-transfer of any of these tools to our administrative management system (Figure 26).

8.1.1 Completed Academic Work

FROntIER

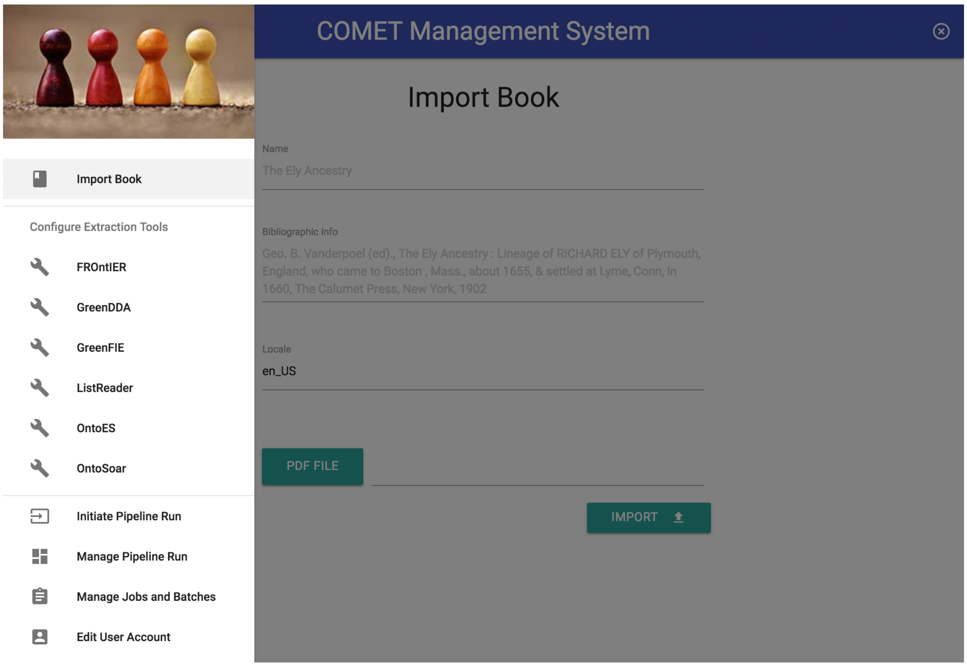
Nearly 20 years ago we began building ontology-based extraction systems (Embley et al. 1999a, 1998a,b). These systems extracted named entities from small record-like write-ups like car ads and obituaries. The extracted entities from these write-ups were assumed to all be related to the primary object, e.g. the car being advertised for car ads and the deceased person for obituaries. As explained in Sections 3.1 and 2.2, FROntIER extends this work to extract and populate relationships and ontology snippets (Park 2015).

In an evaluation of FROntIER extraction, we developed entity and relationship extraction rules sufficient to correctly extract the information from *The Ely Ancestry* Page 419 in Figure 1 with respect to the ontology in Figure 2. We then applied the extraction ontology to Page 479 and obtained the results in Table 1.

Table 1: Sample FROntIER Results

	Prec.	Rec.	F-score
Name	1.00	1.00	1.00
BirthDate	1.00	0.96	0.98
DeathDate	1.00	1.00	1.00
MarriageDate	1.00	1.00	1.00
born on	0.92	0.82	0.87
died on	0.75	0.75	0.75
son of	1.00	0.83	0.91
daughter of	0.67	0.33	0.44
child of	0.79	0.59	0.68
married	1.00	0.50	0.67

FROntIER's strengths are its rich facilities for expressing extraction rules for semi-structured text and for repetitive phrases in free-running text. Its weakness are that extraction rules must be hand-coded by experts at developing complex regular expressions and that no help is available for identifying patterns for which rules are needed (typically many dozen for a book like *The Ely Ancestry*). Painstakingly finding all the patterns, or at least sufficient to make the rule-set worthwhile, and developing a non-conflicting and reasonably minimal set of rules can be tedious and time-consuming.



Import Book

Name
The Ely Ancestry

Bibliographic Info
Geo. B. Vanderpoel (ed)., The Ely Ancestry : Lineage of RICHARD ELY of Plymouth, England, who came to Boston , Mass., about 1655, & settled at Lyme, Conn, in 1660, The Calumet Press, New York, 1902

Locale
en_US

Figure 26: Management System Interface

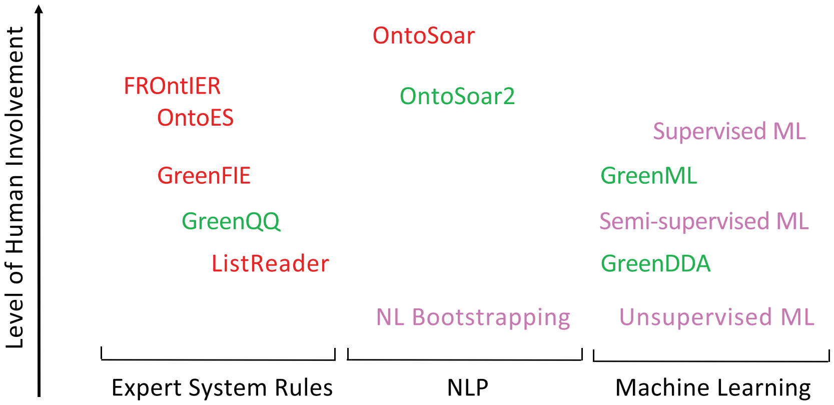
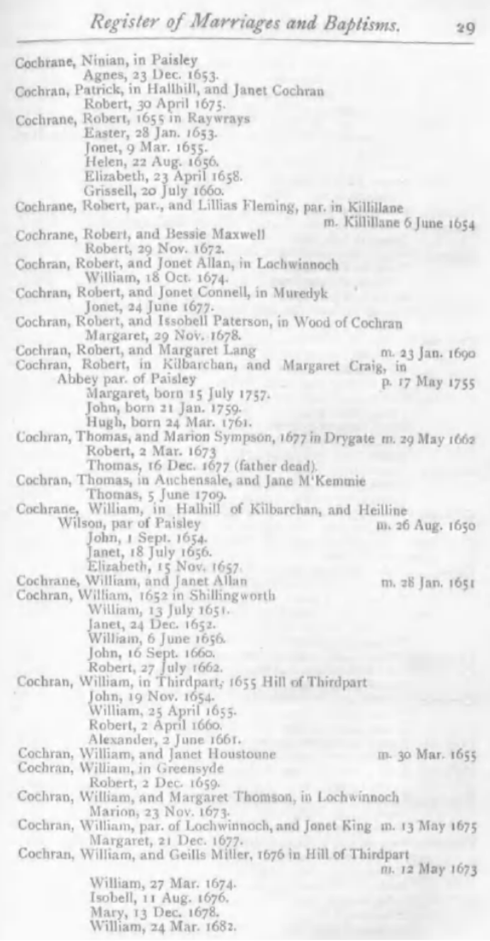


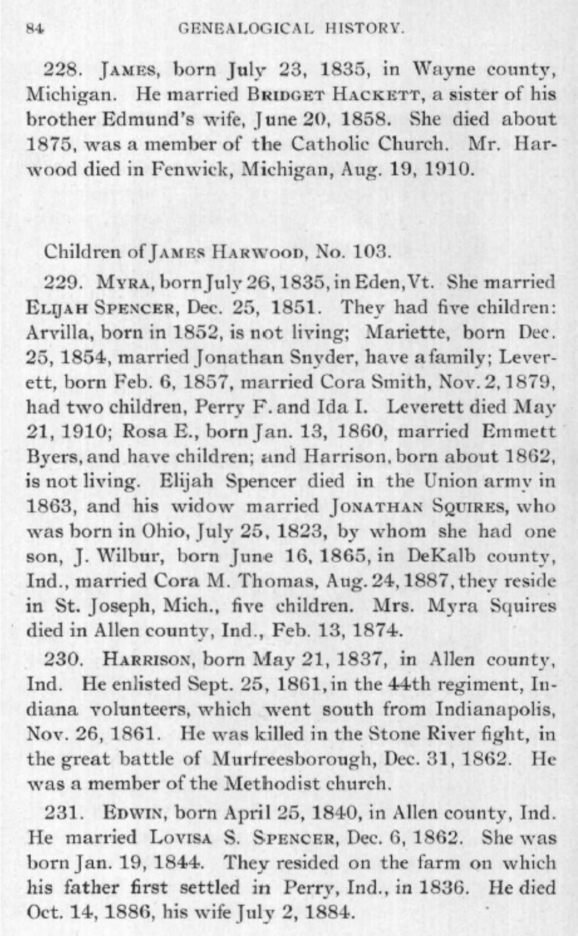
Figure 27: Extraction Tools: Methodology Type and Ease of Use



Register of Marriages and Baptisms. 29

Cochrane, Ninian, in Paisley
Agnes, 23 Dec. 1653.
Cochran, Patrick, in Hallhill, and Janet Cochran
Robert, 30 April 1675.
Cochrane, Robert, 1655 in Raywrays
Easter, 28 Jan. 1653.
Jonet, 9 Mar. 1655.
Helen, 22 Aug. 1656.
Elizabeth, 23 April 1658.
Grissell, 20 July 1660.
Cochrane, Robert, par., and Lillias Fleming, par. in Killillane
m. Killillane 6 June 1654
Cochrane, Robert, and Bessie Maxwell
Robert, 29 Nov. 1672.
Cochran, Robert, and Jonet Allan, in Lochwinnoch
William, 18 Oct. 1674.
Cochran, Robert, and Jonet Connell, in Muredyk
Jonet, 24 June 1677.
Cochran, Robert, and Issobell Paterson, in Wood of Cochran
Margaret, 29 Nov. 1678.
Cochran, Robert, and Margaret Lang m. 23 Jan. 1690
Cochran, Robert, in Kilbarchan, and Margaret Craig, in
Abbey par. of Paisley p. 17 May 1755
Margaret, born 15 July 1757.
John, born 21 Jan. 1759.
Hugh, born 24 Mar. 1761.
Cochran, Thomas, and Marion Sympson, 1677 in Drygate m. 29 May 1662
Robert, 2 Mar. 1673
Thomas, 16 Dec. 1677 (father dead).
Cochran, Thomas, in Auchensale, and Jane M'Kemmie
Thomas, 5 June 1709.
Cochrane, William, in Halhill of Kilbarchan, and Heilline
Wilson, par of Paisley m. 26 Aug. 1650
John, 1 Sept. 1654.
Janet, 18 July 1656.
Elizabeth, 15 Nov. 1657.
Cochrane, William, and Janet Allan m. 28 Jan. 1651
Cochran, William, 1652 in Shillingworth
William, 13 July 1651.
Janet, 24 Dec. 1652.
William, 6 June 1656.
John, 16 Sept. 1660.
Robert, 27 July 1662.
Cochran, William, in Thirdpart, 1655 Hill of Thirdpart
John, 19 Nov. 1654.
William, 25 April 1655.
Robert, 2 April 1660.
Alexander, 2 June 1661.
Cochran, William, and Janet Houstoune m. 30 Mar. 1655
Cochran, William, in Greensyde
Robert, 2 Dec. 1659.
Cochran, William, and Margaret Thomson, in Lochwinnoch
Marion, 23 Nov. 1673.
Cochran, William, par. of Lochwinnoch, and Jonet King m. 13 May 1675
Margaret, 21 Dec. 1677.
Cochran, William, and Geills Miller, 1676 in Hill of Thirdpart
m. 12 May 1673
William, 27 Mar. 1674.
Isobell, 11 Aug. 1676.
Mary, 13 Dec. 1678.
William, 24 Mar. 1682.

Figure 28: Page 29 of the Kilbarchan Parish Record (Grant 1912)



84 GENEALOGICAL HISTORY.

228. James, born July 23, 1835, in Wayne county, Michigan. He married Bridget Hackett, a sister of his brother Edmund's wife, June 20, 1858. She died about 1875, was a member of the Catholic Church. Mr. Harwood died in Fenwick, Michigan, Aug. 19, 1910.

Children of James Harwood, No. 103.

229. Myra, born July 26, 1835, in Eden, Vt. She married Elijah Spencer, Dec. 25, 1851. They had five children: Arvilla, born in 1852, is not living; Mariette, born Dec. 25, 1854, married Jonathan Snyder, have a family; Leverett, born Feb. 6, 1857, married Cora Smith, Nov. 2, 1879, had two children, Perry F. and Ida I. Leverett died May 21, 1910; Rosa E., born Jan. 13, 1860, married Emmett Byers, and have children; and Harrison, born about 1862, is not living. Elijah Spencer died in the Union army in 1863, and his widow married Jonathan Squires, who was born in Ohio, July 25, 1823, by whom she had one son, J. Wilbur, born June 16, 1865, in DeKalb county, Ind., married Cora M. Thomas, Aug. 24, 1887, they reside in St. Joseph, Mich., five children. Mrs. Myra Squires died in Allen county, Ind., Feb. 13, 1874.

230. Harrison, born May 21, 1837, in Allen county, Ind. He enlisted Sept. 25, 1861, in the 44th regiment, Indiana volunteers, which went south from Indianapolis, Nov. 26, 1861. He was killed in the Stone River fight, in the great battle of Murfreesborough, Dec. 31, 1862. He was a member of the Methodist church.

231. Edwin, born April 25, 1840, in Allen county, Ind. He married Lovisa S. Spencer, Dec. 6, 1862. She was born Jan. 19, 1844. They resided on the farm on which his father first settled in Perry, Ind., in 1836. He died Oct. 14, 1886, his wife July 2, 1884.

Figure 29: Page 84 of the History of the Harwood Families (Harwood 1911)

OntoES

As explained in Section 3.2, OntoES adopts FROntIER's ontology-snippet extraction capability and extends it for complex annotations. Further, for our genealogy application, OntoES specifically targets its extraction to the three ontologies in Figures 7, 8, and 9, and it is supported by the extraction rule creation and testing interface illustrated in Figure 10.

In an evaluation of OntoES, we created 25 ontology-snippet extraction rules—7 for the Person ontology, 4 for Couple, and 14 for Family—which together were sufficient to capture all the information of interest from the Kilbarchan page in Figure 28 and also the previous and subsequent page. Then, in a fully automatic extraction run over the 143 pages of *The Kilbarchan Parish Record*, these 25 rules extracted information for 8,539 individuals. Based on a check of several randomly chosen pages, the automatic extraction's F-score was judged to be near 95%.

Strengths of OntoES include its form-filling paradigm and its support for ontology-snippet extraction rule creation. Its weakness, like FROntIER's, is its requirement for hand-coded extraction rules. However, an interface like the one in Figure 10 nicely supports rule creation and testing, and goes a long way to mitigate the problems associated with hand-coding extraction rules.

GreenFIE

As explained in Section 3.3, GreenFIE synergistically works with users who are filling in forms. Once a record's fields are filled in, GreenFIE can generate a regular-expression extraction rule that would extract the same information, generalize the rule, execute it, and populate subsequent records.

Table 2 gives the results of an experiment we conducted. For both *The Ely Ancestry* and *The Kilbarchan Parish Record* we selected a sequence of three pages and filled in records until we achieved 100% recall. GreenFIE generated an extraction rule for every record filled in by the user and executed it to fill in subsequent records for the user to check and in some cases complete if GreenFIE filled in only part of the record. Although incorrectly filled-in records could have been deleted, for the experiment none were deleted so that we could measure the precision of the GreenFIE-generated extraction rules.

The rate of task completion is the total number of records correctly extracted divided by the total number of user actions (new records filled-in and partially filled-in records completed). The maximum rate, for example, is 19.60, which tells us that filling in just five records is sufficient to extract all the information in the 98 records for the Person form on the three Kilbarchan pages. Table 2 also tells us the total number of records extracted, 333, and the total number of user actions, 121. We estimated the number of data values extracted in these 333 records to be about 1,000 so that approximately 8.2 data values were obtained for each user-extracted or user-edited record.

As strengths, we note that GreenFIE users never need to write, or even edit, extraction rules. We also note that GreenFIE's rules are highly precise (0.99 for the overall precision in Table 2), which when coupled with COMET's highlighting of extracted text for records greatly facilitates check and correct. Considering weaknesses, we were disappointed with the overall rate of task completion. Although quite precise, GreenFIE's heuristic generalization of regular-expression rules will likely never rival a human expert. Human experts may, however, benefit from having GreenFIE generate regular expressions for them to adjust rather than having to write rules from scratch.

ListReader

As explained in Section 3.5, ListReader discovers record patterns in the text and generates extraction rules to recognize these records. To extract the information from recognized records, a human provides labels for the component parts of the recognized text to map each part to an ontology.

Table 3 shows the results of an experiment we conducted. We developed ListReader using *The Ely Ancestry* and tested it on a similar book, *Shaver-Dougherty* (Shaffer n.d.), and on *The Kilbarchan Parish Record*. For the ground truth, we labeled a few dozen pages in these books. The Saver-Dougherty results in Table 3 were essentially obtained after labeling about 25 patterns, and the Kilbarchan after only about a half dozen—“essentially” because both books have a long tail of less frequent patterns, many of which were labeled.

As strengths, ListReader processes an entire book at once, discovering patterns without human intervention, and it efficiently assists users with the labeling task by ordering the patterns by greatest impact first and by cross-labeling, which sometimes obviates the need to label a pattern at all. ListReader's weaknesses include its inability to recognize large patterns such as parents-with-child lists and its apparent low recall as seen in the experimental results.

OntoSoar

As explained in Section 3.7 OntoSoar parses sentential chunks of text and then applies a cognitive reasoner to map the parse to an ontology. As reported in (Lindes et al. 2015), the results of running OntoSoar over a text snippet from Figure 1 are in Table 4 and over a text snippet from Figure 29 are in Table 5. Recall errors are mostly due to information expressed in linguistic patterns—especially for listing children—that have not yet been encoded into the system.

As strengths, OntoSoar works on free running text, and it can process semi-structured text as well. The LG-parser it uses does not depend on the chunks of text it processes being complete

Table 2: GreenFIE Results

	User Action	Correct (=Total)	Correct per User Action	Incorrect	Precision	Recall	F-score
Ely	86	157	1.83	2	0.99	1.00	0.99
Person	28	76	2.71	2	0.97	1.00	0.99
Couple	26	43	1.65	0	1.00	1.00	1.00
Family	32	38	1.19	0	1.00	1.00	1.00
Kilbarchan	35	176	5.03	3	0.98	1.00	0.99
Person	5	98	19.60	0	1.00	1.00	1.00
Couple	15	30	2.00	2	0.94	1.00	0.97
Family	15	48	3.20	1	0.98	1.00	0.99
Overall	121	333	2.75	5	0.99	1.00	0.99

Table 3: Sample ListReader Results

	Prec.	Rec.	F-score
Shaver-Dougherty	0.94	0.39	0.55
Kilbarchan	0.94	0.54	0.68

sentences nor on the phrases being grammatically correct. OntoSoar's main weakness is its dependence on hand-coded production rules for Soar (Laird 2012). Obtaining the results in Tables 4 and 5 required 240 hand-coded production rules.

8.1.2 Academic Work in Progress

Preliminary results for the extraction tools we present in this section look promising, but much more academic work is needed to bring these tools to fruition and ready for tech-transfer to a production environment.

GreenQQ

Developed in an independent effort, GreenQQ is currently being investigated for adaption and use in our FamilySearch reading system. As explained in Section 3.3, it interacts with users via text-snippet examples. Beginning with the most frequently occurring token-sequence patterns that contain information of interest, GreenQQ proposes extraction rules for a user to adjust and accept for execution. After execution, GreenQQ again proposes rules and continues in this synergistic interaction cycle until the results are satisfactory.

In an initial trial run, we applied GreenQQ to the 119 pages of *The Kilbarchan Parish Record* that are similar to the page in Figure 28. Interacting with a user for less than 50 minutes, GreenQQ created 40 templates that classified sequences of tokens into 5 classes (HEAD of household, WIFE, BABY, GEO location, and DATE of birth, christening, marriage, or proclamation of marriage). In the final cycle, GreenQQ processed 9,464 lines of text with 89,391 tokens and found 17,206 matches in 5 seconds of runtime. Table 6 shows the accuracy results of the extraction for three randomly chosen pages (Embley and Nagy 2017).

A strength of GreenQQ is its ease of use. Classified text snippets presented to users as candidate extraction rules require only that a user is knowledgeable enough to identify and classify the part of the text snippet to be extracted. A weakness of GreenQQ is that it only does named entity recognition (NER). Thus, we must restrict GreenQQ's usage to ontologies with one central non-lexical object set that is directly connected to all of its lexical object sets through relationship sets—like the ontologies in Figures 7, 8, and 9. Furthermore, in its currently implemented state, an additional weakness is that it cannot be used when ontology records are interleaved as *Couple* and *Family* records are in Figure 1. However, as explained in Section 3.3, we expect to be able to resolve this weakness. Resolving it may require human input, which could lessen GreenQQ's ease of use.

GreenML/GreenDDA

We have carried out experiments with NER using the different levels of machine learning discussed in Section 3.6. We used the Stanford CoreNLP system in OTS (off-the-shelf), GreenML, and GreenDDA settings to perform

Table 4: OntoSoar Results for the Charles Christopher Lathrop family and the first three lines of the commentary about Miss Emma Goble Lathrop in Figure 1

Category	Exists	Found	Correct	P Errors	R Errors	P	R	F
Persons	12	11	11	0	1	100.0%	91.7%	95.7%
Births	6	6	6	0	0	100.0%	100.0%	100.0%
Deaths	4	4	4	0	0	100.0%	100.0%	100.0%
Marriages	1	1	1	0	0	100.0%	100.0%	100.0%
Sons & Daughters	7	2	2	0	5	100.0%	28.6%	44.4%
Totals/Average	30	24	24	0	6	100.0%	80.0%	88.9%

Table 5: OntoSoar Results for paragraph 229 plus the header preceding the paragraph in Figure 29

Category	Exists	Found	Correct	P Errors	R Errors	P	R	F
Persons	19	15	14	1	4	93.3%	73.7%	82.4%
Births	8	8	7	1	0	87.5%	87.5%	87.5%
Deaths	5	3	3	0	2	100.0%	60.0%	75.0%
Marriages	6	6	4	2	0	66.7%	66.7%	66.7%
Sons & Daughters	9	0	0	0	9	N/A	0.0%	0.0%
Totals/Average	47	32	28	4	15	87.5%	59.6%	70.9%

Table 6: GreenQQ Results. An extract was judged to be partially correct if the sequence of labeled tokens for the extract was a proper subsequence of the ground-truth token sequence for the extract. Soft scoring counted partially correct extracts as being correct, while Hard scoring counted them as being incorrect.

					Soft:Correct=Correct+Partial			Hard:Incorrect=Partial+Incorrect		
Class	Total	Correct	Partial	Incorrect	Recall	Precision	F-score	Recall	Precision	F-score
HEAD	72	71	0	0	0.99	1.00	0.99	0.99	1.00	0.99
WIFE	56	53	1	0	0.96	1.00	0.98	0.95	0.98	0.96
BABY	92	91	0	1	0.99	0.99	0.99	0.99	0.99	0.99
DATE	123	116	0	0	0.94	1.00	0.97	0.94	1.00	0.97
GEO	65	63	0	4	0.97	0.94	0.95	0.97	0.94	0.95
Overall	408	394	1	5	0.97	0.99	0.98	0.97	0.99	0.98

human-supervised annotation of a typical 10-page range of *The Ely Ancestry*.

Figure 30 displays the results of evaluated annotation performance against a gold standard. We ran OTS on each page as is—without any new training data. For both green techniques we annotated the first page according to the gold standard and used it as the only training data to train up a model to run on the second page. Thus, in Figure 30 the GreenML and GreenDDA results for the first and second page are identical. For GreenML, we then corrected the results for the second page, making the annotations identical to the gold standard, and then trained up a new model using the gold standard for both the first and second page. Continuing, we generated results for subsequent pages one at a time, always training up models using the gold standard for all previous pages. For GreenDDA, however, we provided no additional gold-standard training data after the first page. Instead, GreenDDA generated its own training data. Thus, for the third page, we trained a model using the gold standard for the first page and the GreenDDA-generated annotations for the second page. Continuing, we generated results for subsequent pages one at a time, always training up models using the gold standard for the first page and all GreenDDA-generated annotations for all other previous pages.

OTS annotation performance varies substantially by page, given the variability of the data across pages, and it will continue to vary in this way since each page is run with the same model. GreenML initially performs poorly as compared with OTS, but gradually increases until it compares favorably with OTS. Interestingly, after initially performing poorly as compared with OTS, GreenDDA results gradually trend upward and at the end in this experiment have the best results. Whether these NER results will hold for other page sequences in the book or in other books and whether results for relationship recognition will be similar remains to be seen.

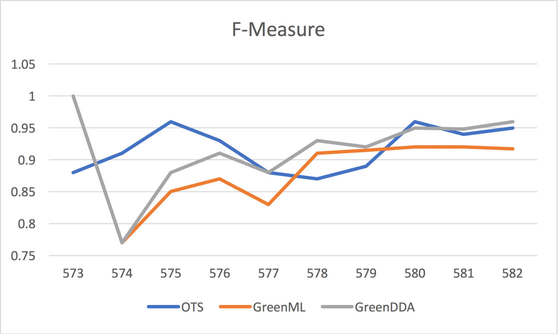


Figure 30: F-measure annotation results for a 10-page range with OTS versus green methods

OntoSoar2

Writing Soar production rules by hand is a complex and intensive process, calling into question the scalability of the current OntoSoar approach. Each new type of linguistic construction that contains information that could map to the semantic representation and ultimately the ontology needs to be treated in this manner. A future instantiation of the system, OntoSoar2, could include a rule compiler which would allow for rules to be coded in a metalinguistic grammar and compiled directly into Soar code, as is being done elsewhere (Lindes and Laird 2016).

Another potential feature for OntoSoar2 would involve an improved syntactic parsing process. The current LG parser, while robust and flexible, also requires hand-coding of custom rules, and does not include any machine learning functionality. More recently, dependency parsers (Mel'čuk 1988) have been implemented for many languages beyond English in frameworks such as CoreNLP, which support machine learning and incremental training.

Furthermore, it is unclear how flexible the current OntoSoar semantic representation framework (Embodied Construction Grammar) is for supporting large-scale NLP applications like ours (Lindes et al. 2017). While it has proven effective in relatively narrow domains such as robotics (Lindes and Laird 2017), OntoSoar2 work would need to explore the appropriateness of this or other alternative deep semantic representations.

Finally, OntoSoar2 has the potential to further implement of some of the language processing capabilities inherent in our current cognitive architecture framework. This could include such "human" aspects of processing such as being incremental (i.e. word-by-word rather than a sentence as a whole); eclectic (i.e. leveraging more pragmatic and real-world knowledge); repair-based (i.e. reformulating hypotheses that prove untenable in the presence of further context); and grounded (i.e. integrating perceptual input such as page layouts).

8.2 Pipeline Processing

Given a book that has been scanned and OCR'ed, the pipeline processes the book from import (a single PDF document of the full book) to export (a GedcomX document for each page that contains genealogical information). The pipeline runs in several steps:

1. *Prepare Pages*. Split the input PDF document into pages. For each PDF page generate four additional files: an xml file containing the OCR information for the page; a text file of the OCR'ed characters assembled into words and lines; and a PNG image and HTML document that together let a COMET user view and work with the page as an image superimposed over hidden OCR'ed text aligned with the text in the image.
2. *Extract Information*. Apply the extraction tools to the text files and, for each tool and page, populate the ontology in Figure 15.
3. *Merge Information*. For each page merge the information in the tool-populated ontologies and create a single populated ontology.
4. *Check and Correct Information*. Run a constraint checker over the extracted and merged information, discover anomalies, and fix those identified as being automatically rectifiable. If patrons are to check and correct the information, split the information into *Person*, *Couple*, and *Family* form data and for each form/page combination display the filled in form in COMET for a user to check and correct.
5. *Enhance Information*. To the extent possible, standardize person names, place names, and dates, and infer gender, birth names, married names, and formal names.
6. *Generate GedcomX*. Generate a GedcomX file for each page containing genealogical information. In addition, for each person having associated genealogical information, generate a person information document detailing associated names, event dates and places, and marriage and parent-child relationships (e.g. see Figure 31). Generate also an HTML document with all the extracted information for a person highlighted on an image of the page (or pages, if the information spans more than one page).

```
***********************************
Person osmx190: ALBRIGHT, ESTHER R.
***********************************
Name:
   Conclusion: Esther R. Albright
   Interpreted Document Text: ALBRIGHT, ESTHER R.
   Original Document Text: ALBRIGHT, ESTHER R.
   Inferred Formal Name: Esther R. Morris Albright
      Title:
      First Names: Esther R.
      Last Names: Morris Albright
      Suffix:
   Inferred Birth Name: Esther R. Morris
   Inferred Married Name: Esther R. Albright
   Gender: Female
Facts:
   BirthDate:
      Conclusion: 22 July 1863
      Interpreted Document Text: 22 July 1863
      Original Document Text: 22 July 1863
   BirthPlace:
      Interpreted Document Text: Butler Co OH
   DeathDate:
      Conclusion: 1 January 1946
      Interpreted Document Text: 1 Jan 1946
      Original Document Text: 1 Jan 1946
   DeathPlace:
      Interpreted Document Text: 113 Sherman St Dayton OH
   BurialDate:
      Conclusion: 3 January 1946
      Interpreted Document Text: 3 Jan 1946
      Original Document Text: 3 Jan 1946
   BurialPlace:
      Interpreted Document Text: Abbottsville Cem Dke Co OH
Marriage Relationships:
   Spouse: osmx334 (l-linfield S. Albright)
ParentOf Relationships
ChildOf Relationships:
   osmx169 (Thomas Benton Morris)
   osmx480 (Angeline Harrod)
```

Figure 31: Esther Albright Information as a Result of Full Pipeline Processing

A prototype of the pipeline runs from beginning to end, and the code is being improved as we gain experience and encounter new edge cases. (Pipeline processing for complex annotations, for example, is currently being added.) The extraction engines, whose academic prototypes are complete, all run, but considerable work is still required to convert them into tools usable by anyone besides ourselves. Academic research is continuing for extraction tools still under development. COMET has been used by subjects in some experimental evaluations; they generally find it usable after a few minutes of training (Woodfield et al. 2016). We have only begun to build a management system that will control the processing of books through the pipeline.

8.3 Ingest into the *Family Tree*

We have conducted several field tests to determine how our extraction results can contribute to the *Family Tree* (Embley et al. 2017).

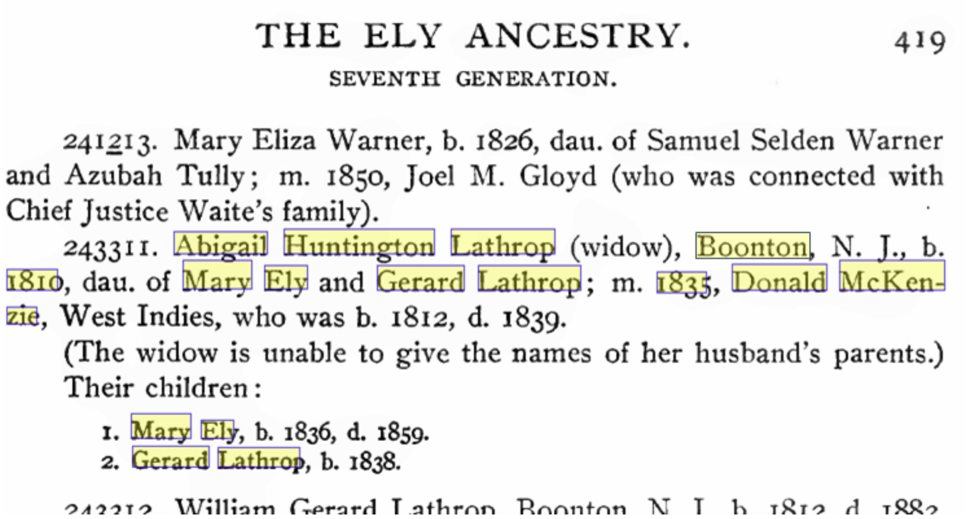
Ely We processed the page in Figure 1 through the pipeline—extracted information using FROntIER, Ontos, and OntoSoar; merged it; checked and corrected it with COMET; standardized the data; inferred gender and extended name information; and generated person information documents for each person mention on the page having associated genealogical information. To compare the effort to ingest information manually with a proposed automatic ingest, we updated the *Family Tree* by hand according to the generated person information documents. We filled in search forms, identified matching *Family Tree* records, merged duplicates (if any), checked the matching records, and added to them source documentation and missing information. From 31 unique generated person information documents, we found that 28 matched exactly one *Family Tree* person record. For the Mary Ely married to Gerard Lathrop we found two, as Figure 21 shows, and we merged them. Donald McKenzie's and Abigail Huntington Lathrop's person information documents both matched three records that were themselves duplicates, and in both cases we merged the three records. We added highlighted source documents like the one in Figure 32 for all 31 matched tree records. Overall, we (1) replaced two primary names with more complete names (e.g. "Emma Sutherland Goble" in place of "Emma S. Goble"); (2) replaced six uncertain BMD (Birth/Marriage/Death) facts (e.g. "about 1831" or merely "deceased") with certain facts; (3) added two missing BMD facts, and (4) added eight supplementary facts such as married names or alternate spellings of names. All of this work, which could have been done fully automatically within seconds of compute time, took more than five hours of tedious typing, checking, clicking, and waiting for responses from the FamilySearch web site.

Kilbarchan In a fully automatic extraction run over the 143-pages of the Kilbarchan parish record (Grant 1912), the pipeline, running without COMET-user intervention, created 8,539 person information documents like the one in Figure 31. The F-score for the automatic extraction was judged to be near 95%. In a sample of 150 of these 8,539 person information documents, a prototype matching algorithm was 100% correct when the tools extracted eight or more distinct items of information for the person. Of the correctly matched person records, 20% had information that could be added to the *Family Tree*, including adding or fixing first and last names, event dates, and parent-child relationships.

Miller Similar to our Kilbarchan field test, in a fully automatic extraction run over the 396-page Miller book (Miller Funeral Home Records, 1917 – 1950, Greenville, Ohio 1990), information for 12,226 individuals was extracted. Of the 1,280 individuals the matching algorithm found in the *Family Tree* with certainty, the Miller records provided information that could be added to 57% of them.

8.4 Web of Knowledge

The grand challenge of collecting and representing the world knowledge of any domain (scientific, geo-political, or any other) in a structured, searchable, online repository has been and is the dream of many visionaries. Projects with this vision in mind include: open information extraction systems that have extracted billions of assertions as the basis for both common-sense knowledge and novel question-answering systems (Banko et al.



THE ELY ANCESTRY. 419

SEVENTH GENERATION.

241213. Mary Eliza Warner, b. 1826, dau. of Samuel Selden Warner and Azubah Tully; m. 1850, Joel M. Gloyd (who was connected with Chief Justice Waite's family).

243311. Abigail Huntington Lathrop (widow), Boonton, N. J., b. 1810, dau. of Mary Ely and Gerard Lathrop; m. 1835, Donald McKenzie, West Indies, who was b. 1812, d. 1839.

(The widow is unable to give the names of her husband's parents.) Their children:

1. Mary Ely, b. 1836, d. 1859.
2. Gerard Lathrop, b. 1838.

243312. William Gerard Lathrop, Boonton, N. J., b. 1812, d. 1882.

Figure 32: Abigail's Information Highlighted

2007; Etzioni 2011; Fader et al. 2011; Mausam et al. 2012); Yahoo!'s Web of Concepts (Dalvi et al. 2009); and large projects such as Cyc (Lenat and Guha 1989), Freebase (Bollacker et al. 2008), DBpedia (Auer et al. 2007), YAGO (Suchanek et al. 2007), YAGO3 (Mahdisoltani et al. 2015), and NELL (Mitchell et al. 2015).

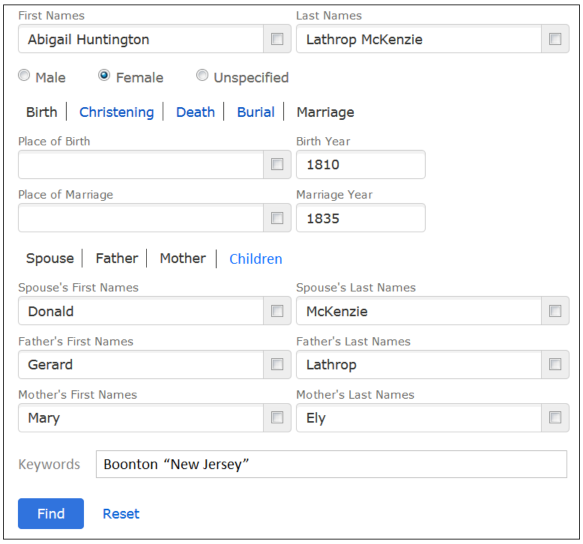
A Web of Knowledge (WoK), as we envision it (Embley et al. 2011), aims at a specific domain of interest such as family history. The backbone of each WoK is an ontology that describes the domain and is to be populated with information. As an example, FamilySearch's conceptualization of family history is populated with basic genealogical information for about 5.8 billion persons, 1.2 billion of which are on the public *Family Tree*. In addition, it stores over a billion auxiliary data items including more than 830 million images and transcripts of documents used as source documentation of genealogical facts; more than 22 million stories and pictures submitted by individuals as memories of their ancestors; nearly 350,000 family history books with pages like those in Figures 1, 28, and 29; and more than 2,000 historical record collections (e.g. census records, burial records, military records) consisting of more than a billion historical documents, many millions of which have been indexed for semantic search.

The backbone of a WoK should be an ontology in its true sense—domain knowledge, agreed upon and shared by a community (Dillon et al. 2008). A WoK ontology is itself worthy of study and understanding and may evolve over time as its community comes to a greater understanding of itself. When populated, the ontology may more rightly be seen as a *knowledge base*—a particular state of the domain whose information can be queried and updated (Dillon et al. 2008). The extraction ontologies we have been discussing in this paper may more accurately be seen as *operational views* of the larger WoK ontology—*views* in the traditional database sense and thus themselves ontologies albeit for a much smaller domain and *operational* in the sense that they enable document reading and query search along with computational operations over stored data.

We focus our remaining comments on WoK query formulation and execution and show its connection to ontological document reading. The correspondence among form, ontology, and semi-structured text provides insight into how to query the WoK: (1) create a form for the results wanted and have the system learn how to correlate it with the WoK ontology and fill it in (Embley 1989); (2) explore the graphical conceptual model view of the WoK ontology and filter desired information directly from the graphical view (Czejdo et al. 1990); (3) generate forms from WoK ontology snippet views and allow constraints to be applied to form fields (Zitzelberger et al. 2015); and (4) from a free-form query "read" and "understand" it to obtain what is wanted (Zitzelberger et al. 2015).

Most interesting from the standpoint of document reading are HyKSS free-form queries (Zitzelberger et al. 2015). "HyKSS" which stands for "Hybrid Keyword and Semantic Search" allows users to issue free-form queries such as "Find Abigail McKenzie who lived to be nearly 100 years old and whose husband was from the West Indies" HyKSS applies its collection of extraction ontologies to the query. Here, the *Name* recognizer in the ontology in Figure 2 recognizes "Abigail McKenzie" as a name, and "West Indies" is recognized as a *LocationName* in the ontology in Figure 22. The phrase "lived to be nearly 100 years old" associates with an *Age* operator in a *Date* data frame, and "husband" associates with the marriage relationship. The HyKSS keyword recognizer ignores stop words ("Find", "who", "and", "whose", "was", "from", "the"), words recognized as denoting operators ("lived to be", "years old"), and non-equality operands ("nearly 100") and treats the remaining words and any quoted phrases as keywords ("Abigail", "McKenzie", "West", "Indies"). As is the case for all search engines, keywords in source documents have been previously indexed. For HyKSS, semantics have also been indexed by applying extraction ontologies to documents, in particular for this example the page in Figure 1. Once the query has been "read" and "understood" (i.e. once the information and keywords have been extracted from the query and mapped to an ontology), a formal query can be generated and executed (Zitzelberger et al. 2015). The results are returned in search-engine fashion—a ranked clickable list of URLs with a snippet showing what was matched. In our example, the first link would likely lead to the page in Figure 1. Clicking would bring up the page with the identified keywords and semantic data highlighted.

Although processing free-form queries is interesting from a document-reading point of view, in applications like genealogy in which the ontological conceptualizations are well known, starting with a form search is likely better. Figure 33 shows one of FamilySearch's forms mocked-up with an additional search field for *Children* so that it better matches the underlying ontology and also an additional search field for *Keywords* to allow for HyKSS-like search over a hybrid of keywords and semantics. The form is filled in for the information extracted from Figure 1 for Abigail Huntington Lathrop with "Boonton" and "New Jersey" as keywords. The results should come back in a search-engine-like list of links to documents, which when clicked should yield a highlighted document like the one in Figure 32.



First Names: Abigail Huntington
Last Names: Lathrop McKenzie

○ Male ● Female ○ Unspecified

Birth | Christening | Death | Burial | Marriage

Place of Birth:
Birth Year: 1810

Place of Marriage:
Marriage Year: 1835

Spouse | Father | Mother | Children

Spouse's First Names: Donald
Spouse's Last Names: McKenzie

Father's First Names: Gerard
Father's Last Names: Lathrop

Mother's First Names: Mary
Mother's Last Names: Ely

Keywords: Boonton "New Jersey"

Figure 33: Query for Abigail Huntington Lathrop McKenzie

Finally, we mention that for a world-wide application like FamilySearch, query processing should be multilingual. We show in (Embley et al. 2014) how extraction ontologies can form the backbone of a query processing system that allows queries to be expressed in a user's native language, then processed in the language of the document, with results translated back into the user's native language. Clicking on resulting document links would bring up highlighted original-language documents.

9 Conclusion

We have defined ontology-based document reading and have expounded upon our experience in implementing an ontological document reading system. The reading system transforms asserted facts stated in a document into objects and relationships and populates the object and relationship sets of a conceptual model. It thus populates the ontology represented by the conceptual model, which gives meaning to the extracted text. Lexical object sets are populated directly with text tokens found in the document. Non-lexical object sets are populated by ontological commitment.

Besides directly populating lexical object sets with text found in a document, a reading system should also be able to "read between the lines" and infer author-implied facts such as a female's married name given her spouse's name or a birth name given a child's father's name. The populating objects and relationships must also make sense with respect to declared ontological constraints or otherwise be rejected, fixed, or at least questioned. An ideal reading system should also be able to extend or adjust its ontological conceptualizations and make connections among them.

In this experience report, we have described an implemented ontology-based document reading system for contributing to FamilySearch's online wiki-like *Family Tree*. The reading system reads an input document, usually a book, and populates the target extraction ontology in Figure 2 with information extracted from the document's text by an ensemble of extraction engines. It then merges and checks the information against ontological constraints and corrects constraint violations when possible. Optionally, it allows a human to check and and correct extraction results. Given the extraction results, it standardizes names and dates and infers gender and various name forms. Finally, it generates both a GedcomX document for each book page and an individual report for each person mentioned in the book who has genealogical information. Either the GedcomX documents or the individual reports can be used for automatic or user-checked semi-automatic import into the *Family Tree*.

The document reading system we have elucidated works particularly well for semi-structured document reading and for populating ontologies with facts within rich but narrow domains of interest like family history. We nevertheless foresee the possibility of using ontology-based document reading as a means to contribute to the construction of a general Web of Knowledge, possibly involving many interconnected domains of interest.

References

Aristotle (about 350BC) Metaphysics. (1993 translation). Oxford University Press, New York

Auer S., Bizer C., Kobilarov G., Lehmann J., Cyganiak R., Ives Z. (2007) DBpedia: A Nucleus for a Web of Open Data. In: Proceedings of the 6th International The Semantic Web and 2Nd Asian Conference on Asian Semantic Web Conference. Busan, Korea, pp. 722–735

Banko M., Cafarella M., Soderland S., Broadhead M., Etzioni O. (2007) Open Information Extraction from the Web. In: Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI 2007). Hyderabad, India, pp. 2670–2676

Benjelloun O., Garcia-Molina H., Menestrina D., Su Q., Whang S. E., Widom J. (2009) Swoosh: A Generic Approach to Entity Resolution. In: The VLDB Journal—The International Journal on Very Large Data Bases 18(1), pp. 255–276

Bhattacharya I., Getoor L. (2007) Collective Entity Resolution in Relational Data. In: ACM Transactions on Knowledge Discovery from Data 1(1)

Bollacker K., Evans C., Paritosh P., Sturge T., Taylor J. (2008) Freebase: A Collaboratively Created Graph Database for Structuring Human Knowledge. In: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data. Vancouver, Canada, pp. 1247–1250

Buitelaar P., Cimiano P., Frank A., Hartung M., Racioppa S. (2008) Ontology-based information extraction and integration from heterogeneous data sources. In: Int. J. Hum.-Comput. Stud. 66(11), pp. 759–788 https://doi.org/10.1016/j.ijhcs.2008.07.007

Buitelaar P., Cimiano P., Haase P., Sintek M. (2009) Towards Linguistically Grounded Ontologies. In: Proceedings of the 6th European Semantic Web Conference (ESWC'09). Heraklion, Greece, pp. 111–125

Chen P. P. (1983) English Sentence Structure and Entity-Relationship Diagrams. In: Inf. Sci. 29(2-3), pp. 127–149 https://doi.org/10.1016/0020-0255(83)90014-2

Chiticariu L., Li Y., Reiss F. (2013) Rule-based Information Extraction is Dead! Long Live Rule-based Information Extraction Systems! In: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing. Seattle, Washington, USA, pp. 827–832

Cimiano P. (2006) Ontology Learning and Population from Text: Algorithms, Evaluation and Applications. Springer, New York, New York

Cimiano P., Völker J. (2005) Text2Onto—A Framework for Ontology Learning and Data-driven Change Discovery. In: Proceedings of the 10th International Conference on Applications of Natural Language to Information Systems (NLDB'05). Alicante, Spain, pp. 227–238

Cimiano P., Völker J., Studer R. (2006) Ontologies on Demand? - A Description of the State-of-the-Art, Applications, Challenges and Trends for Ontology Learning from Text. In: 57, pp. 315–320

Clark K., Manning C. D. (2016) Improving Coreference Resolution by Learning Entity-level Distributed Representations. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016), pp. 643–653

Curland M., Halpin T. A. (2012) Enhanced Verbalization of ORM Models. In: On the Move to Meaningful Internet Systems: OTM 2012 Workshops, Confederated International Workshops: OTM Academy, Industry Case Studies Program, EI2N, INBAST, META4eS, OnToContent, ORM, SeDeS, SINCOM, and SOMOCO 2012, Rome, Italy, September 10-14, 2012. Proceedings, pp. 399–408 https://doi.org/10.1007/978-3-642-33618-8_54

Czejdo B., Elmasri R., Embley D., Rusinkiewicz M. (1990) A Graphical Data Manipulation Language for an Extended Entity-Relationship Model. In: Computer 23(3), pp. 26–36

Dalvi N., Kumar R., Pang B., Ramakrishnan R., Tomkins A., Bohannon P., Keerthi S., Merugu S. (2009) A Web of Concepts. In: Proceedings of PODS'09. Providence, Rhode Island, USA

Datalog User Manual. http://www.ccs.neu.edu/home/ramsdell/tools/datalog/datalog.html

Dillon T., Chang E., Hadzic M., Wongthongtham P. (2008) Differentiating Conceptual Modelling from Data Modelling, Knowledge Modelling and Ontology Modelling and a Notation for Ontology Modelling. In: Proceedings of the Fifth Asia-Pacific Conference on Conceptual Modelling. Wollongong, NSW, Australia, pp. 7–17

Duke: Fast Deduplication Engine. https://code.google.com/p/duke/

Eikvil L. (1999) Information Extraction from World Wide Web: A Survey. 945. Norweigan Computing Center

Elmagarmid A., Ipeirotis P., Verykios V. (2007) Duplicate Record Detection: A Survey. In: IEEE Transactions on Knowledge and Data Engineering 18(1), pp. 1–16

Embley D. (1980) Programming With Data Frames for Everyday Data Items. In: Proceedings of the 1980 National Computer Conference. Anaheim, California, pp. 301–305

Embley D. (1989) NFQL: The Natural Forms Query Language. In: ACM Transactions on Database Systems 14(2), pp. 168–211

Embley D. (1998) Object Database Development: Concepts and Principles. Addison-Wesley, Reading, Massachusetts

Embley D., Campbell D., Jiang Y., Liddle S., Lonsdale D., Ng Y.-K., Smith R. (1999a) Conceptual-Model-Based Data Extraction from Multiple-Record Web Pages. In: Data & Knowledge Engineering 31(3), pp. 227–251

Embley D., Campbell D., Jiang Y., Ng Y.-K., Smith R., Liddle S., Quass D. (1998a) A Conceptual-Modeling Approach to Extracting Data from the Web. In: Proceedings of the 17th International Conference on Conceptual Modeling (ER'98). Singapore, pp. 78–91

Embley D., Campbell D., Liddle S., Smith R. (1998b) Ontology-Based Extraction and Structuring of Information from Data-Rich Unstructured Documents. In: Proceedings of the 7th International Conference on Information and Knowledge Management (CIKM'98). Washington D.C., pp. 52–59

Embley D., Jackman D., Xu L. (2001) Multifaceted Exploitation of Metadata for Attribute Match Discovery in Information Integration. In: Proceedings of the International Workshop on Information Integration on the Web (WIIW'01). Rio de Janeiro, Brazil, pp. 110–117

Embley D., Jiang Y., Ng Y.-K. (1999b) Record-Boundary Discovery in Web Documents. In: Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data (SIGMOD'99). Philadelphia, Pennsylvania, pp. 467–478

Embley D., Kurtz B., Woodfield S. (1992) Object-oriented Systems Analysis: A Model-Driven Approach. Prentice Hall, Englewood Cliffs, New Jersey

Embley D., Liddle S., Eastmond T., Lonsdale D., Price J., Woodfield S. (2017) Conceptual Modeling in Accelerating Information Ingest into *Family Tree* In: Conceptual Modeling Perspectives Cabot J., Gómez C., Pastor O., Sancho M. (eds.) Springer, Cham, Switzerland

Embley D., Liddle S., Lonsdale D. (2011) Conceptual Modeling Foundations for a Web of Knowledge. In: Embley D., Thalheim B. (eds.) Handbook of Conceptual Modeling: Theory, Practice, and Research Challenges. Springer, Heidelberg, Germany, chap. 15, pp. 477–516

Embley D., Liddle S., Lonsdale D., Tijerino Y. (2014) Multilingual Extraction Ontologies In: Towards the Multilingual Semantic Web Buitellar P., Cimiano P. (eds.) Springer, pp. 155–173

Embley D., Liddle S., Park J. (2016) Increasing the Quality of Extracted Information by Reading between the Lines In: Ingénierie et management des systèmes d'information—Mélanges en l'honneur de Jacky Akoka Comyn-Wattiau I., du Mouza C., Prat N. (eds.)

Embley D., Nagy G. (2017) Green Interaction for Extracting Family Information from OCR'd Books (submitted for publication)

Embley D., Zitzelberger A. (2010) Theoretical Foundations for Enabling a Web of Knowledge. In: Proceedings of the Sixth International Symposium on Foundations of Information and Knowledge Systems (FoIKS'10). Sophia, Bulgaria, pp. 211–229

Etzioni O. (2011) Open information extraction: The second generation. In: Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI'11). Barcelona, Catalonia, Spain, pp. 3–10

Fader A., Soderland S., Etzioni O. (2011) Identifying Relations for Open Information Extraction. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing. Edinburgh, United Kingdom, pp. 1535–1545

FamilySearch. http://familysearch.org

Finkel J., Grenager T., Manning C. (2005) Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling. In: Proceedings of the 43nd Annual Meeting of the Association for Computational Linguistics (ACL 2005), pp. 363–370

Fliedl G., Kop C., Mayr H. C. (2003) From Scenarios to KCPM Dynamic Schemas: Aspects of Automatic Mapping. In: Natural Language Processing and Information Systems, 8th International Conference on Applications of Natural Language to Information Systems, June 2003, Burg (Spreewald), Germany, pp. 91–105 http://subs.emis.de/LNI/Proceedings/Proceedings29/article818.html

Fliedl G., Kop C., Mayr H. C. (2005) From textual scenarios to a conceptual schema. In: Data Knowl. Eng. 55(1), pp. 20–37 https://doi.org/10.1016/j.datak.2004.11.007

Fliedl G., Kop C., Mayr H. C., Salbrechter A., Vöhringer J., Weber G., Winkler C. (2007) Deriving static and dynamic concepts from software requirements using sophisticated tagging. In: Data Knowl. Eng. 61(3), pp. 433–448 https://doi.org/10.1016/j.datak.2006.06.012

Fliedl G., Kop C., Mayr H. C., Winkler C., Weber G., Salbrechter A. (2004) Semantic Tagging and Chunk-Parsing in Dynamic Modeling. In: Natural Language Processing and Information Systems, 9th International Conference on Applications of Natural Languages to Information Systems, NLDB 2004, Salford, UK, June 23-25, 2004, Proceedings, pp. 421–426 https://doi.org/10.1007/978-3-540-27779-8_41

Gallaire H., Minker J. (eds.) Logic and Data Bases, Symposium on Logic and Data Bases. Advances in Data Base Theory. Plenum Press, New York

Gedcom X. http://www.gedcomx.org/

Grant F. (ed.) Index to The Register of Marriages and Baptisms in the PARISH OF KILBARCHAN, 1649–1772. J. Skinner & Company, LTD, Edinburgh, Scotland

Grishman R. (2015) Information Extraction. In: IEEE Intelligent Systems 30, pp. 8–15

Grosz B., Joshi A., Weinstein S. (1995) Centering: A Framework for Modeling the Local Coherence of Discourse. In: Computational Linguistics 21(2), pp. 203–225

Gruber T. (1993) A Translation Approach to Portable Ontology Specifications. In: Knowledge Acquisition 5(2), pp. 199–220

Guizzardi G., Halpin T. A. (2008) Ontological foundations for conceptual modelling. In: Applied Ontology 3(1-2), pp. 1–12 https://doi.org/10.3233/AO-2008-0046

Halpin T. (2004) Business Rule Verbalization. In: Proceedings of the 3rd International Conference on Information Systems Technology and its Applications. Salt Lake City, Utah, pp. 39–52

Halpin T. A., Curland M. (2006) Automated Verbalization for ORM 2. In: On the Move to Meaningful Internet Systems 2006: OTM 2006 Workshops, OTM Confederated International Workshops and Posters, AWeSOMe, CAMS, COMINF, IS, KSinBIT, MIOS-CIAO, MONET, OnToContent, ORM, PerSys, OTM Academy Doctoral Consortium, RDDS, SWWS, and SeBGIS 2006, Montpellier, France, October 29 - November 3, 2006. Proceedings, Part II, pp. 1181–1190 https://doi.org/10.1007/11915072_21

Harwood W. (1911) A Genealogical History of the Harwood Families, Descended from Andrew Harwood, Whose English home was in Dartmouth, Devonshire, England, And who emigrated to America, and was living in Boston, Mass., in 1643, Third. Published by Watson H. Harwood, M.D., Chasm Falls, New York

Jiménez P., Corchuelo R., Sleiman H. (2016) ARIEX: Automated Ranking of Information Extractors. In: Knowledge-Based Systems 93, pp. 84–108

Kang H., Getoor L., Shneiderman B., Bilgic M., Licamele L. (2008) Interactive Entity Resolution in Relational Data: A Visual Analytic Tool and Its Evaluation. In: IEEE Transactions on Visualization and Computer Graphics 14(5)

Kim T. (2017) A Green Form-Based Information Extraction System for Historical Documents. MA thesis, Brigham Young University, Provo, Utah

Kop C., Mayr H. C., Zavinska T. (2004) Using KCPM for Defining and Integrating Domain Ontologies. In: Web Information Systems - WISE 2004 Workshops: WISE 2004 International Workshops, Brisbane, Australia, November 22-24, 2004. Proceedings, pp. 190–200 https://doi.org/10.1007/978-3-540-30481-4_19

Kushmerick N., Weld D., Doorenbos R. (1997) Wrapper Induction for Information Extraction. In: Proceedings of the 1997 International Joint Conference on Artificial Intelligence, pp. 729–735

Laender A., Ribeiro-Neto B., da Silva A., Teixeira J. (2002) A Brief Survey of Web Data Extraction Tools. In: SIGMOD Record 31(2), pp. 84–93

Laird J. (2012) The Soar Cognitive Architecture. The MIT Press, Cambridge, Massachusetts

Lehnert W., Cardie C., Fisher D., McCarthy J., Riloff E., Soderland S. (1994) Evaluating an Information Extraction System. In: Journal of Integrated Computer-Aided Engineering 1(6)

Lenat D., Guha R. (1989) Building Large Knowledge-Based Systems; Representation and Inference in the Cyc Project. Addison-Wesley Longman Publishing Co., Inc., Boston, Massachusetts, USA

Liddle S., Embley D., Woodfield S. (1993) Cardinality Constraints in Semantic Data Models. In: Data & Knowledge Engineering 11(3), pp. 235–270

Liddle S., Embley D., Woodfield S. (1995) Unifying Modeling and Programming Through an Active, Object-Oriented, Model-Equivalent Programming Language. In: Proceedings of the Fourteenth International Conference on Object-Oriented and Entity-Relationship Modeling (OOER'95). Gold Coast, Queensland, Australia, pp. 55–64

Lindes P. (2014) OntoSoar: Using Language to Find Genealogy Facts. MA thesis, Brigham Young University, Provo, Utah

Lindes P., Laird J. (2016) Toward Integrating Cognitive Linguistics and Cognitive Language Processing. In: Proceedings of the International Conference on Cognitive Modeling (ICCM'16). The Pennsylvania State University, University Park, Pennsylvania, USA

Lindes P., Laird J. (2017) Cognitive Modeling Approaches to Language Comprehension using Construction Grammar. In: Proceedings of the AAAI 2017 Spring Symposium on Computational Construction Grammar and Natural Language Understanding. Technical Report SS-17-02, pp. 213–221

Lindes P., Lonsdale D., Embley D. (2015) Ontology-based Information Extraction with a Cognitive Agent. In: Proceedings of the AAAI-15 Special Track on Cognitive Systems. Austin, Texas

Lindes P., Mininger A., Kirk J., Laird J. (2017) Grounding Langauge for Interactive Task Learning. In: Proceedings of the First Workshop on Lnaguage Grounding for Robotics. Vancouver, Canada, pp. 1–9

Mahdisoltani F., Biega J., Suchanek F. (2015) YAGO3: A Knowledge Base from Multilingual Wikipedias. In: Proceedings of the 7th Biennial Conference on Innovative Data Systems Research (CIDR 2015). Asilomar, California, USA

Marie A., Gal A. (2007) Managing Uncertainty in Schema Matcher Ensembles. In: Scalable Uncertainty Management, First International Conference, SUM 2007, Washington, DC, USA, October 10-12, 2007, Proceedings, pp. 60–73 https://doi.org/10.1007/978-3-540-75410-7_5

Mausam, Schmitz M., Bart R., Soderland S., Etzioni O. (2012) Open Language Learning for Information Extraction. In: Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning. Jeju Island, Korea, pp. 523–534

Mayr H. C., Kop C. (1998) Conceptual Predesign - Bridging the Gap between Requirements and Conceptual Design. In: 3rd International Conference on Requirements Engineering (ICRE '98), Putting Requirements Engineering to Practice, April 6-10, 1998, Colorado Springs, CO, USA, Proceedings, p. 90 https://doi.org/10.1109/ICRE.1998.667813

Mel'čuk I. (1988) Dependency Syntax: Theory and Practice. State University of New York Press, Albany, State University Plaza, Albany, New York

Miller Funeral Home Records, 1917 – 1950, Greenville, Ohio

Mitchell T., Cohen W., Hruschka E., Talukdar P., Betteridge J., Carlson A., Dalvi B., Gardner M., Kisiel B., Krishnamurthy J., Lao N., Mazaitis K., Mohamed T., Nakashole N., Platanios E., Ritter A., Samadi M., Settles B., Wang R., Wijaya D., Gupta A., Chen X., Saparov A., Greaves M., Welling J. (2015) Never-Ending Learning. In: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI-15). Austin, Texas, pp. 2302–2310

Müller H., Freytag J.-C. (2003) Problems, Methods, and Challenges in Comprehensive Data Cleansing. HUB-IB-164. Humboldt University Berlin

Nadeau D., Sekine S. (2007) A Survey of Named Entity Recognition and Classification. In: Lingvisticae Investigationes: International Journal of Linguistics and Language Resources 30(1), pp. 3–26

Nagy G. (2012) Estimation, Learning, and Adaptation: Systems that Improve with Use. In: Proceedings of the Joint IAPR International Workshop on Structural, Syntactic, and Statistical Pattern Recognition. Hiroshima, Japan

Nagy G. (2017) G. Nagy, DDA: Decision Directed Adaptation. personal communication

Noy N. (2004) Semantic Integration: A Survey of Ontology-Based Approaches. In: SIGMOD Record 33(4), pp. 65–70

Packer T. (2014) Scalable Detection and Extraction of Data in Lists in OCRed Text for Ontology Population Using Semi-Supervised and Unsupervised Active Wrapper Induction. PhD thesis, Brigham Young University

Park J. (2015) FROntIER: A Framework for Extracting and Organizing Biographical Facts in Historical Documents. MA thesis, Brigham Young University, Provo, Utah

Rahm E., Bernstein P. (2001) A Survey of Approaches to Automatic Schema Matching. In: The VLDB Journal 10, pp. 334–350

Rahm E., Do H. (2000) Data Cleaning: Problems and Current Approaches. In: IEEE Data Engineering Bulletin 23(4), pp. 3–13

Recasens M., Hovy E. (2009) A Deeper Look into Features for Coreference Resolution. In: Lalitha Devi S., Branco A., Mitkov R. (eds.) Anaphora Processing and Applications, DAARC 2009. Lecture Notes in Computer Science Vol. 5847. Springer, Berlin, Germany

Salton G. (1968) Automatic Information Organization and Retrieval. McGrawHill

Sarawagi S. (2008) Information Extraction. In: Foundations and Trends in Databases 1(3), pp. 261–377

Schone P., Gehring J. (2016) Genealogical Indexing of Obituaries Using Automatic Processes. In: Proceedings of the Family History Technology Workshop (FHTW'16). https://fhtw.byu.edu/archive/2016. Provo, Utah, USA

Settles B. (2010) Active Learning Literature Survey. Computer Sciences Technical Report 1648. University of Wisconsin–Madison

Settles B. (2012) Active Learning. In: Synthesis Lectures on Artificial Intelligence and Machine Learning 6(1), pp. 1–114

Shaffer H. Shaver/Shafer and Dougherty/Daughery Families also Kiser, Snider and Cottrell, Ferrell, Hively and Lowe Families. Gateway Press, Inc., Baltimore, Maryland, USA

Stroup H. Butler County, Ohio, Cemetery Records Vol. VIII. Hazel Stroup, Hamilton, Ohio

Suchanek F., Kasneci G., Weikum G. (2007) Yago: A Core of Semantic Knowledge. In: Proceedings of the 16th international World Wide Web Conference (WWW 2007). Banff, Alberta, Canada, pp. 697–706

Tao C., Embley D., Liddle S. (2009) FOCIH: Form-based Ontology Creation and Information Harvesting. In: Proceedings of the 28th International Conference on Conceptual Modeling (ER2009). Gramado, Brazil, pp. 346–359

Temperley D., Sleator D., Lafferty J. Link Grammar. http://www.link.cs.cmu.edu/link/

Tijerino Y., Embley D., Lonsdale D., Ding Y., Nagy G. (2005) Toward Ontology Generation from Tables. In: World Wide Web: Internet and Web Information Systems 8(3), pp. 261–285

Turmo J., Ageno A., Català N. (2006) Adaptive Information Extraction. In: ACM Computing Surveys 38(2)

Vanderpoel G. (ed.) The Ely Ancestry: Lineage of RICHARD ELY of Plymouth, England, who came to Boston, Mass., about 1655 & settled at Lyme, Conn., in 1660. The Calumet Press, New York, New York

Wong W., Liu W., Bennamoun M. (2012) Ontology Learning from Text: A Look Back and into the Future. In: ACM Computing Surveys 44(4), 20:1–20:36

Woodfield S., Lonsdale D., Liddle S., Kim T., Embley D., Almquist C. (2016) Pragmatic Quality Assessment for Automatically Extracted Data. In: Proceedings of ER 2016 Vol. LNCS 9974. Gifu, Japan, pp. 212–220

Xu L. (2003) Source Discovery and Schema Mapping for Data Integration. PhD dissertation, Brigham Young University

Zitzelberger A., Embley D., Liddle S., Scott D. (2015) HyKSS: Hybrid Keyword and Semantic Search. In: Journal on Data Semantics 4(4), pp. 213–299