Increasing the Quality of Extracted Information by Reading between the Lines

Joseph Park[1] David W. Embley[1] and Stephen W. Liddle[2]

[1] Department of Computer Science
[2] Information Systems Department
Brigham Young University, Provo, Utah 84602, USA
jspark2012@gmail.com, embley@cs.byu.edu, liddle@byu.edu

Abstract. Much of the information in factual documents is implied, and thus a reader can therefore only obtain such information by inference. Automated information-extraction tools cannot directly extract implied information. FROntIER, the information-extraction tool we present here, not only extracts stated information but also "reads between the lines" to obtain implied information. Being based on a formal foundation of predicate calculus, FROntIER uses inference rules to obtain implied information. As a result FROntIER can improve the quality of extracted information by enhancing it with information the document author intended to convey by implication. In a field study, we show that our implementation of FROntIER can extract valuable information obtainable only by "reading between the lines."

Keywords: implied fact assertions, inference, information extraction, object identity resolution, "reading between the lines".

1 Introduction

Authors of factual documents often convey information by implication and expect readers to understand implied facts by what is explicitly stated. When extracting asserted facts automatically from documents, information-extraction engines typically only extract stated facts and entirely overlook implied facts. The quality of automated extraction, and thus the quality of automatically populated ontologies extracted from factual documents, suffers unless the tools are able to extract the information that authors intend to convey by implication.

Consider, for example, a page from *The Ely Ancestry* [1] in Figure 1. The page contains many dozens of stated facts about families (e.g. basic information about individuals such as birth and death dates, who was born to which parents, and who married whom). Suppose now that a reader is interested in querying for a list of birth names of children whose mother is Abigail McKenzie. There are only two: Mary Ely McKenzie and Gerard Lathrop McKenzie. As none of these names even appear on the page, it takes a fair amount of reasoning to answer the query. We can see that in 1835 Abigail Huntington Lathrop (a female, implied by the name Abigail) married Donald McKenzie (a male, implied by the name

Donald) and therefore would be known as Abigail McKenzie (implied by the social norms of their homeland in the 1800's). Their children, Mary Ely and Gerard Lathrop, would likewise have the surname "McKenzie." For the query, we also want to be careful that we get the right Mary Ely and Gerard Lathrop—not the grandmother of Mary Ely McKenzie, whose name is Mary Ely, and not the grandfather of Gerard Lathrop McKenzie, whose name is Gerard Lathrop. Neither the gender implying mother and father nor the surnames implied by cultural conventions are stated and thus must be inferred.

To automate inference, we extract information into a high-quality conceptual model whose underlying formalism is predicate calculus. We then use standard reasoning to augment the predicate-based conceptualization with an augmented high-quality conceptual model that includes reasoned facts based on stated facts. The inference rules are based on cultural conventions that document authors expect the reader to understand. Some of these conventions are probabilistic, such as (1) gender, based on name and (2) same-person-as, based on known information like parent-child relationships, birth dates, and death dates that tend to uniquely identify duplicate mentions of a person.

A decade of research has produced many information-extraction engines [2, 3], including our own [4], and several text-reading systems [5–7]. None we know of, however, provide for inference grounded in reasoning over formal, predicate-calculus-based conceptual models. We call our proposed extraction and reasoning engine FROntIER (**F**act **R**ecognizer for **Ont**ologies with **I**nference and **E**ntity **R**esolution). FROntIER augments ontology-based extraction engines with

1. standard predicate-calculus-based inference,
2. reasoning based on probabilistic logic, and
3. capabilities to add inferred predicate types to ontologies and populate them.

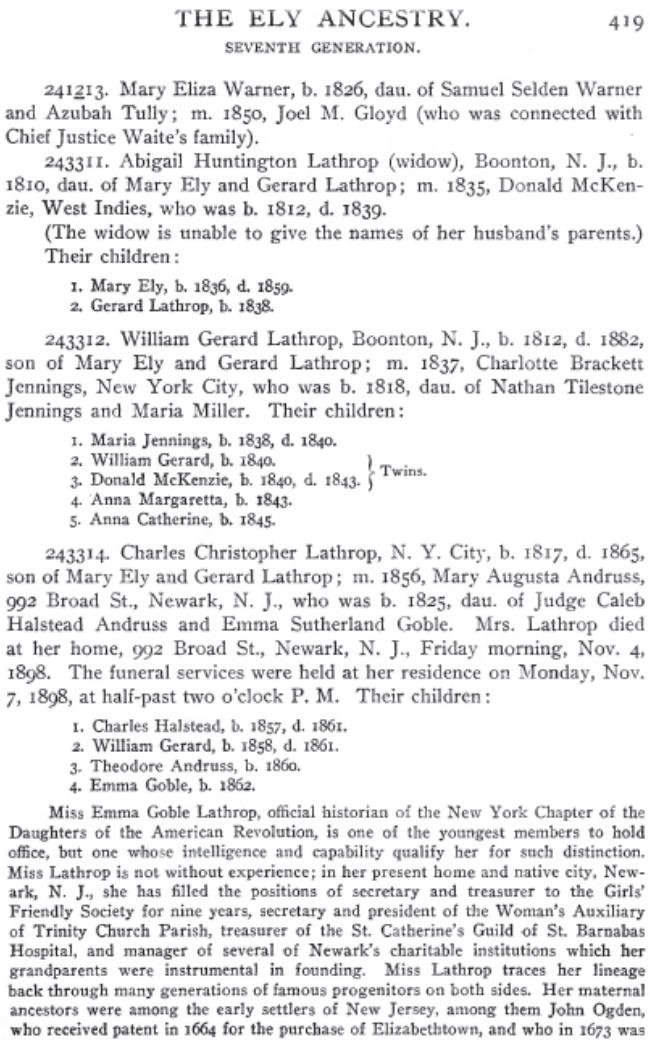
We present the details of these contributions as follows. In Section 2 we briefly describe the formal predicate-calculus foundation of linguistically grounded extraction ontologies. Section 3 shows how we add inference rules, and Section 4 shows how we add object-identity resolution based on extracted and inferred information. Section 5 describes some experimentation with our implementation of FROntIER. We make concluding remarks in Section 6 and mention future research opportunities.

2 Extraction Ontologies

An *extraction ontology* is a linguistically grounded conceptual model. Figure 2 shows the GUI of our Ontology Workbench with the Ontology Editor open, displaying the conceptual model diagram of an extraction ontology. The Tools tab is also open, showing access to the tools for linguistically grounding an extraction ontology.

Formally, an extraction ontology is a 5-tuple (O, R, C, I, L):

O : Object sets—one-place predicates each of whose instance values are either all *lexical*, denoted by named dashed-border rectangles in Figure 2, or all



THE ELY ANCESTRY. 419

SEVENTH GENERATION.

241313. 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

Fig. 1. Page 419 of *The Ely Ancestry.*

nonlexical, denoted by solid-border rectangles (e.g. *DeathDate* is lexical with values such as "Nov. 4, 1898" and *Person* is nonlexical with object-identifier values).

R : Relationship sets—n-place predicates, $n \geq 2$, represented by lines connecting two object-set rectangles (e.g. *Person–Name* in Figure 2) or by diamonds with connecting lines to three or more object sets (e.g. the quaternary re-

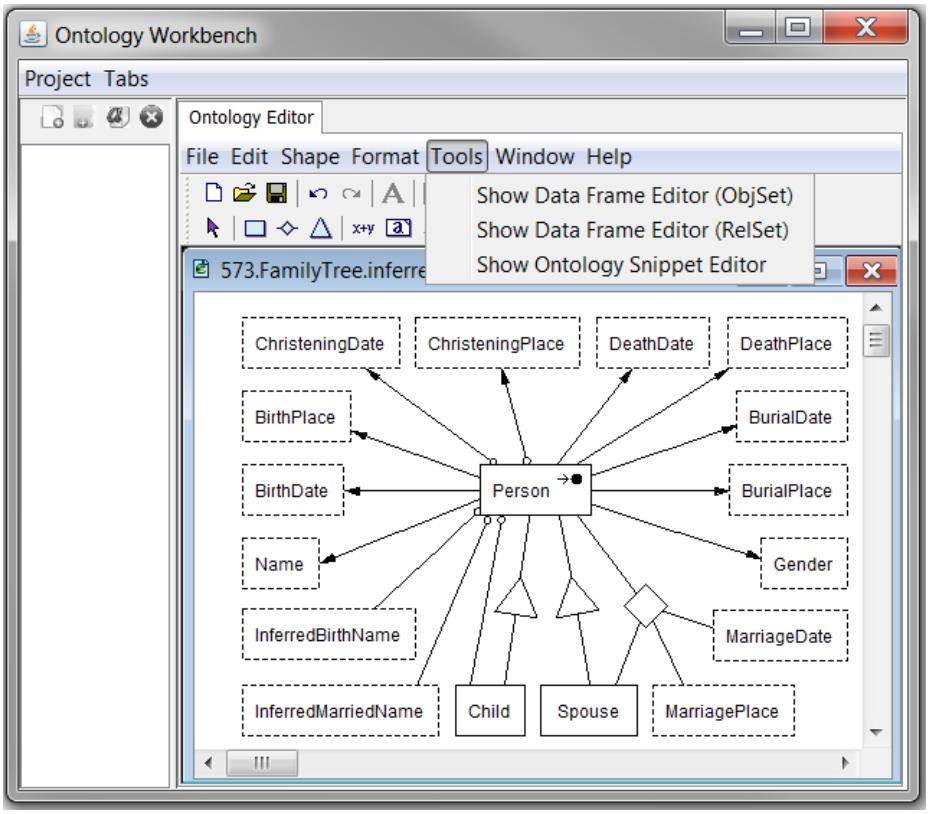


Fig. 2. The Ontology Editor.

lationship set connecting *Person*, *Spouse*, *MarriagePlace*, and *MarriageDate* in Figure 2).

C : Constraints—closed formulas, as implied by the notation (e.g. $\forall x(Person(x) \Rightarrow \exists! y(Person\text{-}BirthDate(x, y)))$—one of the many functional constraints denoted by the arrowhead on the range side of the *Person-BirthDate* relationship set and by a mandatory constraint denoted by the absence of on optional "o" on the domain side; $\forall x(Child(x) \Rightarrow Person(x))$—a generalization/specialization constraint denoted by a triangle, which may optionally also specify mutual exclusion among its specialization sets by a "+" symbol (e.g. mutual exclusion of *Son* and *Daughter* object sets in a specialization of *Person*), or specify that the generalization set is a union of its specialization sets ("$\cup$") or both ("$\uplus$") to form a partition among its specializations).

I : Inference rules—logic rules specified over predicates (e.g. *Daughter* –> *Person-Gender*(x, '*Female*')).

L : Linguistic groundings—text recognizers for populating object and relationship sets and collections of interrelated object and relationship sets as ontology snippets as indicated in the tools menu in Figure 2.

The conceptual foundation for an extraction ontology is a restricted fragment of first-order logic, but its most distinguishing feature is its linguistic grounding [8], which turns an ontological specification into an extraction ontology. Each object set has a *data frame* [9], which is an abstract data type augmented with linguistic recognizers that specify textual patterns for recognizing instance values. Relationship sets may also have data-frame recognizers. Recognizers for larger ontological components are also possible.

We are building an ensemble consisting of a collection of tools to automate the construction of rules and to directly extract information into the ontological structure [10, 11]. FROntIER provides the inference engine and the entity resolver in the ensemble.

3 Inference

FROntIER uses the Jena reasoner[3] to organize facts in conformance to a target ontology (e.g. the ontology in Figure 2). To use the Jena reasoner, we must convert ontology object and relationship instances into RDF triples. To conform with RDF syntactic requirements, we normalize our ontologies as we convert them. We convert lexical object sets into nonlexicals (RDF classes) with a *Value* property and convert n-ary relationship sets ($n > 2$) into binary relationships connected to a nonlexical (RDF class) that represents the n-ary relationship set. As a result, all relationship sets are binary between two RDF classes, and each lexical object set has a property value associated with its RDF class. Consider for example, the quaternary relationship set in Figure 2. The lexical object sets *MarriageDate* and *MarriagePlace* become RDF classes, respectively with a *MarriageDateValue* property and a *MarriagePlaceValue* property. We then create an RDF class for *PersonSpouseMarriageDateMarriagePlace* and form binary relationships between it and each of the four RDF classes *Person*, *Spouse*, *MarriageDate*, and *MarriagePlace*.

Figure 3 gives a sampling of FROntIER inference rules. The rules specify schema mappings between a source ontology s and a target ontology t, as specified in the prefix statements in Figure 3. In our example the *FamilyTree* source ontology is the same as the target ontology in Figure 2 except that it does not include the object sets *InferredBirthName*, *InferredMarriedName*, and *Gender*, which are the items of interest we wish to infer in our application.

Some rules are simply direct transfers of information. The first three rules in Figure 3 are examples. They transfer *Person* objects, *Name* objects, and *Person-Name* relationships from source to target. The Ontology Editor in Figure 2 allows a user to right click on an object set or relationship set and display its content. The left-most pop-up window in Figure 4 shows the result of displaying the content of the *Person-Name* relationship set after executing the first three rules in Figure 3. Note that the nonlexical *Person* objects are all identified by surrogate identifiers (e.g. *osmx411*, the *Person* identifier, for *Emma Southerland Goble*, the *Name* value).

[3] http://jena.apache.org/documentation/inference/index.html

```
@prefix s: <http://dithers.cs.byu.edu/owl/ontologies/FamilyTree#>.
@prefix t: <http://dithers.cs.byu.edu/owl/ontologies/TargetOntology#>.
@prefix ann: <http://dithers.cs.byu.edu/owl/ontologies/annotation#>.

[(?x rdf:type s:Person) -> (?x rdf:type t:Person)]
[(?x rdf:type s:Name), (?x s:NameValue ?nv)
->
 (?x rdf:type t:Name), (?x t:NameValue ?nv)]
[(?x s:Person-Name ?y) -> (?x t:Person-Name ?y)]
...
//Gender based on name
[(?x s:Person-Name ?n), (?n rdf:type s:Name), (?n s:NameValue ?nv),
 isMale(?nv), makeSkolem(?gender, ?x)
->
(?x t:Person-Gender ?gender), (?gender rdf:type t:Gender),
 (?gender t:GenderValue 'Male')]
...
//Birth name for child: surname of father added
[(?p s:Person-Child ?c), (?p s:Person-Name ?n), (?n s:NameValue ?nv),
 (?p t:Person-Gender ?g), (?g t:GenderValue 'Male'), (?c s:Person-Name ?cn),
 (?cn t:NameValue ?cnv), getsurname(?nv, '^(([A-Z][A-Za-z]+)[- ]*)+', ?x),
 getsurname(?cnv, '^(([A-Z][A-Za-z]+)[- ]*)+', ?y), notEqual(?x, ?y),
 strConcat(?cnv, ' ', ?x, ?nx),
 makeSkolem(?bn, ?c, ?p, ?n, ?nv, ?g, ?cn, ?cnv, ?nx)
->
 (?bn rdf:type t:InferredBirthName), (?bn t:InferredBirthNameValue ?nx),
 (?c t:Person-InferredBirthName ?bn)]
...
//Inferred married name for female spouse: married surname
[(?p rdf:type t:Person), (?p t:Person-Name ?n), (?n t:NameValue ?nv),
 (?marriageRecord t:PersonSpouseMarriageDateMarriagePlace-Person ?p),
 (?p t:Person-Gender ?gf), (?gf t:GenderValue 'Female'),
 (?q rdf:type t:Spouse),
 (?marriageRecord t:PersonSpouseMarriageDateMarriagePlace-Spouse ?q),
 (?q t:Person-Gender ?gm), (?gm t:GenderValue 'Male'), (?q t:Person-Name ?mn),
 (?mn t:NameValue ?mnv), getsurname(?mnv, '^(([A-Z][A-Za-z]+)[- ]*)+', ?x),
 strConcat(?nv, ' ', ?x, ?nx),
 makeSkolem(?fsn, ?p, ?n, ?nv, ?marriageRecord, ?gf, ?q, ?gm, ?mn, ?mnv, ?nx)
->
 (?fsn rdf:type t:InferredMarriedName), (?fsn t:InferredMarriedNameValue ?nx),
 (?fsn t:InferredMarriedName-Person ?p)]
```

Fig. 3. Sampling of Inference Rules.

The *Gender* rule in Figure 3 determines the gender of male names. To determine gender, we use a user-defined built-in predicate *isMale*, which accesses a predefined statistical table giving the probability of a name being male or female as computed from the billions of name instances in the FamilySearch.org data store [12]. Using a threshold of 0.75, if the *isMale* predicate holds for an instance of the *NameValue* variable *?nv*, the rule generates a gender object with its *makeSkolem* function and sets its lexical *GenderValue* to be 'Male'. As the second pop-up window in Figure 4 shows, all names in Figure 1 turned out to be highly indicative of gender.

Once we have the gender, then along with the extracted parent-child relationships we can determine the father of a child. Then if the child's name, as extracted, does not have the same surname as the father, we can add the father's

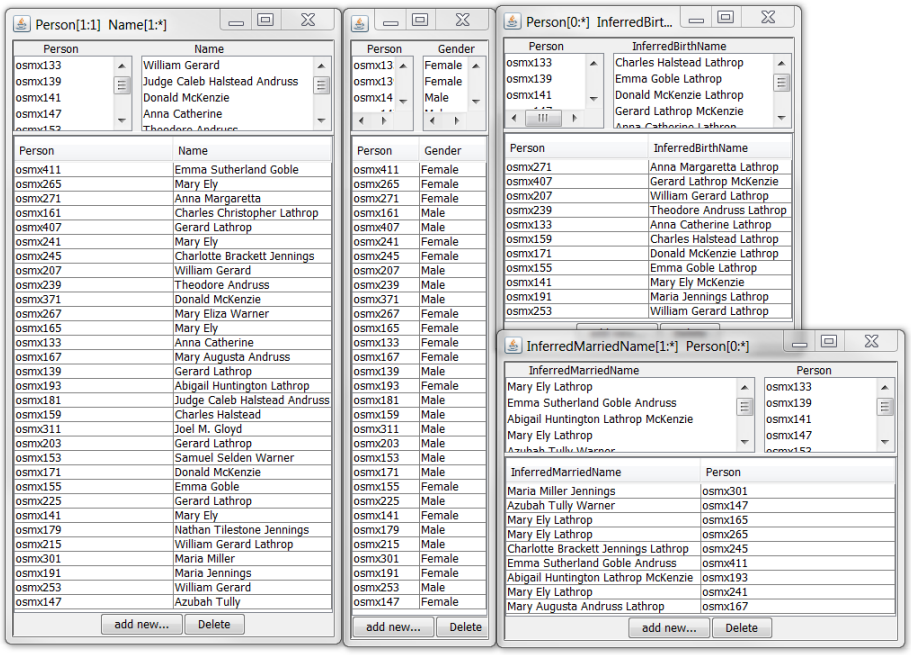


Fig. 4. Inferred Results

surname to the child's name and thus obtain the child's full birth name. The pop-up window in the upper-right of Figure 4 shows that every child in Figure 1 without a surname has a surname added.

Inferring married surname is similar to inferring birth names of children, but is based on the marriage relationship.[4] Before inferring married names of female spouses, we infer the inverse relationships for *Person* and *Spouse* in the *PersonSpouseMarriageDateMarriagePlace* relationship set. Thus, for example, we have *Person*(*Maria Miller*) and *Spouse*(*Nathan Tilestone Jennings*) as a couple as well as the extracted *Person*(*Nathan Tilestone Jennings*) and *Spouse*(*Maria Miller*). Then, in the marriage rule in Figure 3, we only need to add surnames for married persons whose gender is 'Female'. The first entry in the lower-right pop-up window in Figure 4 shows that Maria Miller became Maria Miller Jennings by (assumed) marriage. Indeed, every female spouse in Figure 1 has a correctly inferred married name.

[4] This is the case for our running example. Of course other cultures require different inference rules.

4 Object Identity Resolution

Object identity resolution in FROntIER is about determining whether any two object identifiers in the *Person* object set designate the same person. Object-existence rules make a new surrogate identifier for every extracted *Name*. In Figure 1, for example, the three Mary Elys who are spouses of Gerard Lathrop are the same person, but their generated surrogate identifiers are different: *osmx265*, *osmx241*, and *osmx165* as Figure 4 shows. On the other hand, Mary Ely *osmx141* whose surname is McKenzie is a different Mary Ely.

FROntIER's object identity resolution uses facts for entities in populated target ontologies as input and generates *owl:sameAs* relationships as output. FROntIER uses Duke[5] but could use any any other fact-based entity resolver.

To use Duke, we convert the Jena-inferred RDF triples into a CSV file, which one can view as a table of entity records. For the target ontology in Figure 1, we produce CSV records as follows: (1) convert nonlexicals with a *Value* property into table attributes: e.g. *BirthDate* with a *BirthDateValue* property into the attribute *BirthDate*; (2) convert the quaternary relationship set into *MarriageDate*, *MarriagePlace*, and *Spouse* attributes, where the nonlexical specialization *Spouse* becomes *SpouseName* through its generalization's object existence rule, which is based on *Name*; and (3) calculate the maximum observed cardinality for *Person-Child* instances and for *PersonSpouseMarriageDateMarriagePlace* instances (allowing for more than one marriage) to produce the attributes *Child1Name*, *Child2Name*, ..., *Spouse1Name*, *MarriageDate1*, *MarriagePlace1*, ... up to the maximum number of instances for each.

The Duke entity resolver uses a configuration file to set attribute comparators and parameter values. For our work we used the *ExactComparator* for all attributes. For parameter values, each attribute has a low value for when two attribute-value pairs do not match and a high value for when they do match. Duke combines the values to produce a probability that two entities are the same. For example, we set the high value to 0.73 for birth-date years because we believe that when they match they are moderately discriminating, and we set the low value to 0.002 because mismatched birth-date years are highly discriminating. Gender does not disambiguate persons when they match, but is a strong discriminator when they do not match, so a high value of 0.56 and a low value of 0.01 are appropriate. Similarly, we set other parameter values according to expected significance within the domain.

Given comparator and parameter settings, Duke computes the probability of *same-as* between every pair of extracted person objects. Duke makes conclusions based on a threshold we set. For our extraction and inference results partially shown in Figure 4, a threshold of 0.8 lets Duke correctly conclude for Figure 1 that the first, third, and fourth Mary Ely are the same person, that the first, third, and fourth Gerard Lathrop are the same person, and that all other persons differ from each other.

[5] https://code.google.com/p/duke/

5 Case Study

After creating proper cultural-based inference rules and tuning the entity-resolution engine for the information in *The Ely Ancestry*, we observed that the generated inferred fact assertions that "read between the lines" were always 100% correct with respect to the extracted fact assertions. Indeed, the inferred results are necessarily 100% correct if:

1. the base information is correctly extracted,
2. the cultural inference rules are correctly encoded and application sufficient,
3. the deduplication parameters are tuned correctly for the application, and
4. sufficient information exists and is used to discriminate persons.

Unfortunately, violations of these conditions may occur. Clearly, surnames of children and married female spouses will be missing or incorrect if fathers and husbands are not identified or identified incorrectly. Incorrect gender inference can also cause misidentification. In our experience with automated extraction on fact-filled family-history documents such as *The Ely Ancestry*, we have found that precision errors are rare while (unfortunately) recall errors are prevalent. Thus, though our inferred fact assertions are often incomplete, they are rarely incorrect.

Fortunately, since authors of factual documents can only convey implied information if conditions (2)–(4) hold, it should be possible to correctly encode culture-based inference rules and identify enough discriminating information to distinguish or link individuals. Extracting all needed information, however, is not trivial. Page 419 of *The Ely Ancestry* in Figure 1 is about the families of the children of Mary Ely and Gerard Lathrop. Thus, a reader can easily see the repetition of these grandparents in the family groups; automatically discerning this subtlety, however, is beyond the capability of current extraction engines. Fortunately, in the *The Ely Ancestry*, sufficient extractable information is available on Page 419 to avoid having to discover the implicit exposition of the page's layout.

6 Concluding Remarks

For factual documents such as *The Ely Ancestry*, it is possible to successfully "read between the lines" and extract unstated, implicitly-asserted facts intentionally omitted by document authors. Our FROntIER implementation bases implicit fact-finding on ontologies formalized in predicate calculus. It can therefore use an off-the-shelf reasoner to augment the ontology with new fact types and populate them. Based on both explicit and implied fact assertions, FROntIER can also resolve person object identity with an off-the-shelf deduplicator. Inference accuracy depends only on a correct encoding of cultural inference rules, a sufficiency of information, and proper tuning of entity-resolution software. With proper encoding and tuning, we were always able to achieve 100% inference accuracy with respect to the accuracy of extracted base fact assertions.

Our research in implicit fact finding is already beginning to be applied in family-history applications [13]. As we apply our work in this domain, we are seeing future-work opportunities to expand implicit fact finding by using semantic context. We should, for example, be able to fix some OCR errors based on knowing the semantic type of a data instance and reject some erroneous fact extractions by reasoning that they do not make sense. Overall, we see significant opportunities to improve the quality of extracted information by a reasoned "reading between the lines."

References

1. M.S. Beach, W. Ely, and G.B. Vanderpoel. *The Ely Ancestry.* The Columer Press, New York, New York, 1902.
2. C-H. Chang, M. Kayed, M.R Girgis, and K. Shaalan. A survey of web information extraction systems. *IEEE Transactions on Knowledge and Data Engineering*, 18(10):1411–1428, October 2006.
3. S. Sarawagi. Information extraction. *Foundations and Trends in Databases*, 1(3):261–377, 2008.
4. D.W. Embley, D.M. Campbell, Y.S. Jiang, S.W. Liddle, D.W. Lonsdale, Y.-K. Ng, and R.D. Smith. Conceptual-model-based data extraction from multiple-record web pages. *Data & Knowledge Engineering*, 31(3):227–251, 1999.
5. F.M. Suchanek, G. Kasneci, and G. Weikum. Yago: A core of semantic knowledge. In *Proceedings of the 16th international World Wide Web Conference (WWW 2007)*, Banff, Alberta, Canada, May 2007.
6. O. Etzioni. Open information extraction: The second generation. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI'11)*, Barcelona, Catalonia, Spain, July 2011.
7. T. Mitchell, W. Cohen, E. Hruschka, P. Talukdar, J. Betteridge, A. Carlson, B. Dalvi, M. Gardner, B. Kisiel, J. Krishnamurthy, N. Lao, K. Mazaitis, T. Mohamed, N. Nakashole, E. Platanios, A. Ritter, M. Samadi, B. Settles, R. Wang, D. Wijaya, A. Gupta, X. Chen, A. Saparov, M. Greaves, and J. Welling. Never-ending learning. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI-15)*, Austin, Texas, January 2015.
8. P. Buitelaar, P. Cimiano, P. Haase, and M. Sintek. Towards linguistically grounded ontologies. In *Proceedings of the 6th European Semantic Web Conference (ESWC'09)*, pages 111–125, Heraklion, Greece, May/June 2009.
9. D.W. Embley. Programming with data frames for everyday data items. In *Proceedings of the 1980 National Computer Conference*, pages 301–305, Anaheim, California, May 1980.
10. P. Lindes. OntoSoar: Using language to find genealogy facts. Master's thesis, Brigham Young University, Provo, Utah, 2014.
11. T.L. Packer. *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, 2014.
12. P. Schone and S. Davey. A multilingual personal name treebank to assist genealogical name processing. In *Proceedings of the 12th Annual Family History Technology Workshop*, Salt Lake City, Utah, USA, February 2012.
13. FamilySearch. http://familysearch.org.