Green Interaction for Extracting Family Information from OCR'd Books

David W. Embley
Brigham Young University
Department of Computer Science
Provo, UT, USA
embley@cs.byu.edu

George Nagy
Rensselaer Polytechnic Institute
Electrical, Computer and Systems Engineering
Troy, NY, USA
nagy@ecse.rpi.edu

Abstract— **Repetitively formatted historical books are tokenized and tagged according to eight token types (capitalized words, numbers, punctuation …). To extract family information, templates of short sequences of tags are generated around frequent proper nouns and specified tokens like "born". Each template is associated with a user-assigned class (head of household, father, mother, spouse, geographic location …) and a pointer to an overlapping or nearby fragment of text to be extracted. Matching the template against the book text yields class-labeled factoids. In an interaction cycle, new extraction templates are proposed for user approval or editing. Each edit-then-extract cycle typically yields thousands of factoids and a dozen new templates. With five approximately half-hour interactive sessions, 44,000 genealogical factoids were extracted from a 17th century Scottish register of marriages and births and from published 19th–20th century Ohio funeral parlor records. The experience indicates that this method quickly yields quality results with higher F-score than reported for hand-constructed rule templates.**

Keywords-text template matching; information extraction; historical do-cuments; green interaction

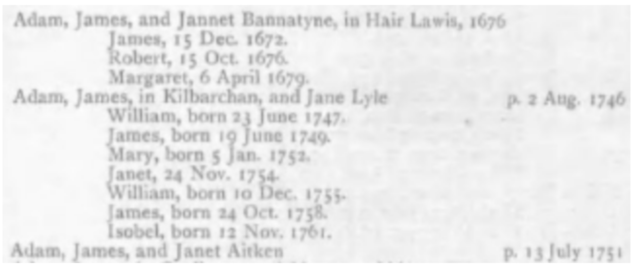
I. Introduction

Many historical books contain family information as organized and quasi-repetitively formatted collections of factoids. Several hundred thousand of these books have already been scanned, OCR'd, and placed online with ever more being added [1],[2]. Fig. 1 shows part of a page image and the corresponding OCR'd transcript from two of these books. Genealogical information needs to be extracted and organized for a variety of applications including, medical research on inherited diseases, understanding the economics of intergenerational poverty, sociological studies of family communities, and tracing the family trees of interested individuals. Tediously extracting the information by hand is nearly infeasible—even by crowd sourcing with thousands of interested volunteers and paid participants—causing those engaged in providing this information to turn to automated information extraction for help. The accommodation of OCR errors is discussed at the end of Section III.

Research on automated information extraction can be dated back at least to Salton's groundwork [3]. For free-running text, NLP and IR researchers' sustained interest in Named Entity Recognition (NER) is exhibited by the NIST-sponsored Text REtrieval Conference [4] which began in 1992 and the major supportive software web sites such as the Stanford CoreNLP NER site [5]. For semi-structured text, 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. Early research in these areas was based on grammars: [6] describes a rule-based system created for address block extraction from text strings, and [7] shows how to induce wrappers to extract information from commercial web sites. Other early applications of information extraction from semi-structured documents include dictionaries [8] and bibliographies and library catalogs [9]. Since these early beginnings, hundreds of research papers in these various disciplines have been published in many journals and conference proceedings. Surveys of this work include [10], [11], [12] and [13], which in addition to surveying the work evaluates and compares several dozen information extraction systems that have been developed over the years.

Several research endeavors specifically relate to various aspects of the work presented here. Extraction of family information from OCR'd documents is described in [14] and [15]. The work in [14] applies machine learning, statistical analyses, and rule-based processing techniques to extract information from obituaries in old newspapers, and in [15] patterns in the abstracted text of full books are discovered from which HMM extraction rules are generated. Our example-based approach for user interaction has some similarities with the end-user-provided training examples used commercially for scanned business documents [16]. Some aspects of our templates, like the use of literals and semantic tags, are anticipated in [17]. The effects of OCR errors on information extraction were discussed in [18]. Being rule-based, the extraction tool we present here is an example of the research called for in [19], which points out that although most recent academic research on automated information extraction relies on machine learning as the methodology of choice, in practice rule-based methodologies dominate deployed information extraction systems.

Like other rule-based systems, our methodology (called *GreenQQ*) exploits the quasi-repetitive format of factoids in semi-structured text to generate and execute extraction rules. GreenQQ is unique, however, in the way it interacts with users to obtain effective new rule templates. It is Green because, like other "green" systems [20], its

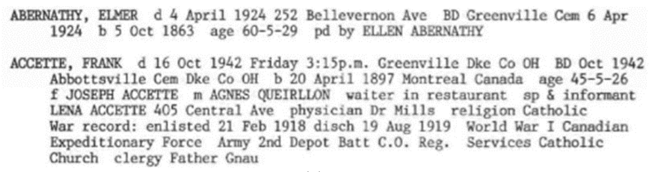


Adam, James, and Jannet Bannatyne, in Hair Lawis, 1676
James, 15 Dec. 1672.
Robert, 15 Oct. 1676.
Margaret, 6 April 1679.
Adam, James, in Kilbarchan, and Jane Lyle p. 2 Aug. 1746
William, born 23 June 1747.
James, born 19 June 1749.
Mary, born 5 Jan. 1752.
Janet, 24 Nov. 1754.
William, born 10 Dec. 1755.
James, born 24 Oct. 1758.
Isobel, born 12 Nov. 1761.
Adam, James, and Janet Aitken p. 13 July 1751

(a)

```
Adam, James, and Jannet Bannatyne, in Hair Lavvis, 1676
James, 15 Dec. 1672.
Robert, 15 Oct. 1676.
Margaret, 6 April 1679.
Adam, James, in Kilbarchan, and Jane Lyle p. 2 Aug. 1746
William, born 23 June 1747.
James, born 19 June 1749.
Mary, born 5 Jan. 1752.
Janet, 24 Nov. 1754.
William, born 10 Dec. 1755.
James, born 24 Oct. 1758.
Isobel, born 12 Nov. 1761.
Adam, James, and Janet Aitken p. 13 July 1751
```

(b)



ABERNATHY, ELMER d 4 April 1924 252 Bellevernon Ave BD Greenville Cem 6 Apr
1924 b 5 Oct 1863 age 60-5-29 pd by ELLEN ABERNATHY

ACCETTE, FRANK d 16 Oct 1942 Friday 3:15p.m. Greenville Dke Co OH BD Oct 1942
Abbottsville Cem Dke Co OH b 20 April 1897 Montreal Canada age 45-5-26
f JOSEPH ACCETTE m AGNES QUEIRLLON waiter in restaurant sp & informant
LENA ACCETTE 405 Central Ave physician Dr Mills religion Catholic
War record: enlisted 21 Feb 1918 disch 19 Aug 1919 World War I Canadian
Expeditionary Force Army 2nd Depot Batt C.O. Reg. Services Catholic
Church clergy Father Gnau

(c)

```
ABERNATHY, ELMER d 4 April 1924 252 Bellevernon Ave BD Greenville Cem 6 Apr
1924 b 5 Oct 1863 age 60-5-29 pd by ELLEN ABERNATHY
ACCETTE, FRANK d 16 Oct 1942 Friday 3:15p.m. Greenville Dke Co OH BD Oct 1942
Abbottsville Cem Dke Co OH b 20 April 1897 Montreal Canada age 45-5-26
f JOSEPH ACCETTE m AGNES QUEIRLLON waiter in restaurant sp & informant
LENA ACCETTE 405 Central Ave physician Dr Mills religion Catholic
War record: enlisted 21 Feb 1918 disch 19 Aug 1919 World War I Canadian
Expeditionary Force Army 2nd Depot Batt C.O. Reg . Services Catholic
Church clergy Father Gnau
```

(d)

Figure 1. Text images from (a) Kilbarchan [Grant1912] and (c) Miller [Miller1990] with respective OCR (b) and (d). The Kilbarchan book was typeset from a transcribed manuscript of original entries in the Kilbarchan Parish Record handwritten by parish vicars. The Miller book, copied from original Miller Funeral Home burial records, was typewritten. Both books were scanned and OCR'd yielding characters with bounding boxes which were then rendered as left-justified lines of text with heuristically set spacing between words and letters in words.

interactive feedback loop increases user productivity. (Q1) it is Quick because its suggestions aid users to quickly generate effective extraction templates and also because it executes quickly allowing for real-time synergies and (Q2) it produces Quality results.

II. Method

Users interact with GreenQQ through either user- or machine-selected samples from the OCR'd text. Initially, users must decide what categories or classes of information are to be extracted, which generally depends on the expected downstream application. For the Kilbarchan parish record, the classes we chose for the evaluation reported below are HEAD (head of household), WIFE, BABY, GEO (geographic location), and DATE of christening, birth, marriage, or proclamation of marriage. From the more complex Miller funeral home burial records, we extracted HEAD (the person being interred), D_Date (death date), BUPD (burial place and date), B_DATE (birth date), AGE, SPOUSE, FATHER, MOTHER, and NRGRANDCH (number of grandchildren). For each class, users select a sample search phrase, and the text to be extracted: e.g. [BABY "James, 15" "James"] where BABY is an explicitly declared class, "James, 15" is the potentially discriminative search phrase, and "James" tells what to extract. The program derives a range index for locating the text to be extracted relative to the search phrase. Here the index is [0, 1] because the extract is the first token of the search phrase. The extracted text can span preceding and following lines and page boundaries, but the current program requires the search phrase to be part of a single line that begins with SOL and ends with EOL

GreenQQ converts the search phrase into a pattern template by tagging each token. The template for "James, 15" is a capitalized word followed by a comma, and a number. (The identical template and index could equally well have been derived from "Margaret, 6" or from many other lines with the same pattern.) Now GreenQQ can search the document for this text pattern and extract the specified information for the class. From (b) in Figure 1, GreenQQ would extract not only "James" as a BABY but also "Robert" "Margaret", and "Janet"—and hundreds of others in the Kilbarchan book with this same pattern.

Moreover, and more importantly for the green aspect of GreenQQ, it also compiles patterns with frequent user-specified tokens (e.g. "born", "p.", … for Kilbarchen) and for proper nouns in the extracted text (here "James", "Robert", "Margaret", …) that were identified as being part of the original pattern template but failed to be identified when they occurred in other text configurations. After completion of each extraction phase on the entire book with all available templates, GreenQQ generates example candidates to help the user create new extraction rules as follows. If some occurrences of "William" (a capitalized word) are identified as BABY because they are followed by a comma and a number, and capitalized words also occur frequently at the beginning of a line and are followed by a comma and "born", then GreenQQ might return an instance from Figure 1b like [BABY "SOL William , born 23 June 1747 . EOL"]. The user can then mark the part of the text to be extracted ("William" in this example) so that GreenQQ can generate a new extraction-rule template (e.g. [SOL capitalized-word comma "born"]). The program also derives the appropriate index (here [1, 2]), which tells the search routine to extract the first word after the start-of-line tag (SOL) wherever the text matches this template. The new rule is then executed, extracting as BABY from the text in Figure 1b not only "William" but also "James", "Mary" , "Janet", another "William", another "James", and "Isobel" and many more from the full book.

After each extraction cycle, GreenQQ offers a user-specified number of search phrases and extract candidates. We find 25 candidates a comfortable batch size to edit. Editing may include changing the class or the extract. (Here, the user could change the class from BABY to DATE and mark "23 June 1747" as the text to be extracted. If the extract is changed, the program adjusts the index.) Continuing in this way, GreenQQ assists in compiling the extracted items into family groups and writing out the collected information into the results file.

Table 1 defines the terminology used in Fig. 2, which shows the overall data flow, and in the list below that illustrates the required steps. To draw attention to the terminology, we capitalize and italicize the terms in Table 1 and in the steps below that describe the process.

TABLE I. TERMINOLOGY

Candidate Extract	*Tokens* to be displayed to the left and right of the first *Token* of the *Search Phrase* of a *Candidate Template*
Candidate Template	automatically generated *Template* with an associated sample *Search Phrase* and corresponding *Candidate Extract* to be approved or edited by the user
Class	descriptors chosen by the user to label the categories of extracted facts
Context Scope	number and location of *Tags* to be prepended and appended to a *Keyword* to construct *Candidate Templates*
Coordinates	page, line and offset of a *Token* in the *Text* or of the corresponding *Tag* in *Tagged Text*
Extract	sequence of *Tokens* (typically 1-4) to be extracted from the *Text* and recorded as output
Display Scope	user-specified length of *Candidate Extract* preceding and following the first word of the *Search Phrase* of a *Candidate Template*
Family Group	factoids associated with the immediately preceding HEAD *Class*, in order of appearance in the text
Index	location of the *Extract* relative to the start of the corresponding *Template*
Keyword	proper nouns and book-specific labels (e.g. John, born, of) that appear in both matched (and therefore classified) and unmatched (still to be classified) portions of the *Text*
Match List	list of the *Coordinates* and *Class* segments of *Text* that matched some *Template*
Page Files	unicode text files of OCR'd pages of a family book
Search Phrase	A sample of a sequence of *Tokens* corresponding to a sequence of *Tags*
Tag	preset label applied to each *Token* (e.g. CAP, NUM)
Template	sequence of *Tags* corresponding to the *Tokens* of a *Search Phrase*
Tagged Text	sequence of *Tags* corresponding to the *Tokens* of the *Text* of the book
Template	sequence of *Tags* (typically 2–5) to be matched against the *Tagged Text*
Text	OCR'd and tokenized book
Token	a punctuation symbol or an alphanumeric string without a blank (space) character

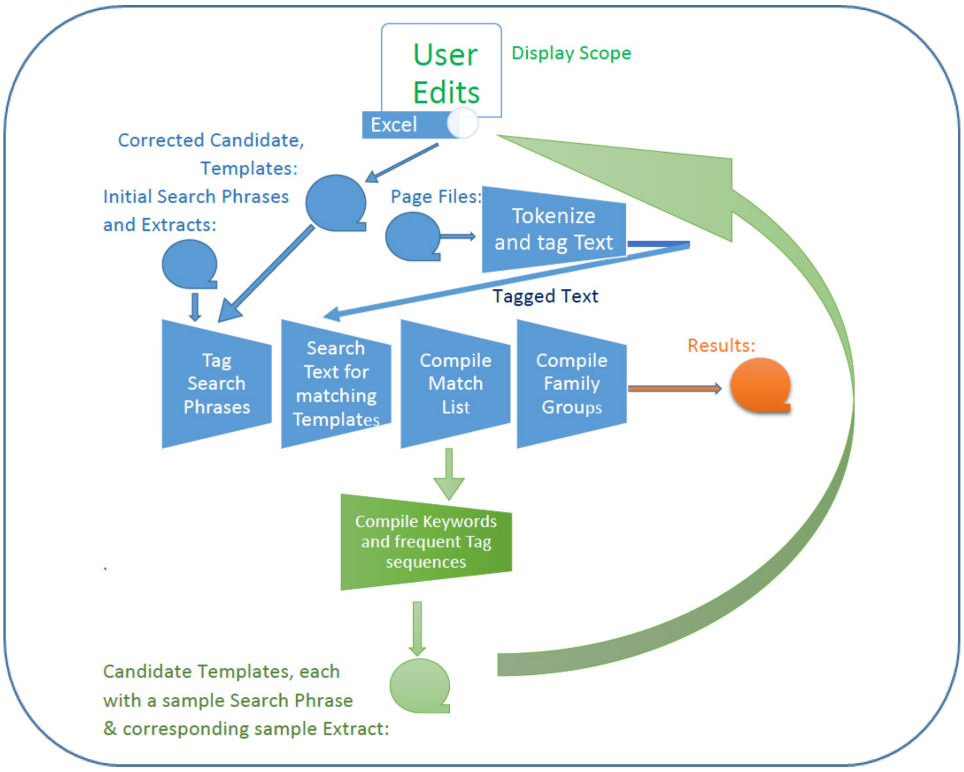

User Edits
Display Scope
Excel
Corrected Candidate, Templates:
Initial Search Phrases and Extracts:
Page Files:
Tokenize and tag Text
Tagged Text
Tag Search Phrases
Search Text for matching Templates
Compile Match List
Compile Family Groups
Results:
Compile Keywords and frequent Tag sequences
Candidate Templates, each with a sample Search Phrase & corresponding sample Extract:

Figure 2. Data flow. (The capitalized terms in the diagram are defined in Table 1).

The list below illustrates the program steps:

1. Read and merge the *Page Files* (cf. Fig. 1b and 1d).
2. Tokenize and tag the *Text*, e.g.:

Text -line (Page 4, Line 6):	/ William, born 23 June 1747.
Text-line *Tokens*:	/ William / , / born / 23 / June / 1747 / . /
Tagged Text line: / SOL / CAP / , / born /ANUM / CAP /ANUM / . /EOL/	

3. Enter initial *Search Phrases* and *Extracts;* on subsequent cycles, edit *Candidate Templates*. In either case, users select representative and discriminative sequences of contiguous *Text*, and a subsequence thereof to be extracted, e.g.:

HEAD	SOL Adam, James,	Adam, James
WIFE	and Janet Bannatyne	Janet Bannatyne
BABY	James, 15	James

4. Construct *Templates*. Tag the *Search Phrase* and add *Indexes* to the initial *Templates.* On subsequent cycles, do the same for *Candidate Templates*, e.g.:

HEAD	SOL CAP , CAP	[1,4]
WIFE	and CAP CAP	[1,3]
BABY	SOL CAP , NUM	[1,2]

5. Sweep each *Template* against the *Tagged Text* and record in a *Match List* the coordinates of every match in the *Text* and the *Class* of the matching *Template*, e.g.:

```
[3, 3, 10, 4, 1, 'GEO']
[3, 5, 0, 0, 4, 'HEAD']
[3, 5, 5, 1, 2, 'WIFE']
[3, 5, 9, 5, 1, 'GEO']
[3, 6, 0, 6, 3, 'BABY']
….
```

6. Compile the list of *Keywords* that appear most often in both the matched and the unmatched portions of the *Text*. Their appearance in the matched portion reveals their *Class*. Some *Keywords*:

Jonet, James, John, Agnes, Robert, Elizabeth, Paisley, William, Lochwinnoch, Kilbarchan, born, m., p.

7. Prepend and append to every appearance of the tagged *Keyword* in the unmatched *Text* the number of *Tags* specified by the *Context Scope* (here [-1,2]), e.g.:

Keyword:	William
Context Scope:	[0, 2]
Frequent *Tag* Sequence:	SOL CAP , born -1 0 +1 +2

8. Compile the most frequently occurring *Tag* sequences formed around *Keywords*. In our running example these include [SOL CAP , born] which is associated with BABY because "William", its source *Keyword*, was matched most often by the initial BABY *Template*.

9. Write a file of *Candidate Templates* consisting of the *Tokens* of the first appearance of each of the most frequently occurring *Tag* sequences and of the corresponding *Candidate Extracts* of surrounding *Tokens* specified by the *Display Scope*, e.g.:

Candidate Extract (with *Display Scope* = 6)::

```
p. 2 Aug. 1746 EOL SOL William , born 23 June 1747 .
-6 -5 -4   -3   -2  -1    0     +1  +2  +3  +4  +5 +6
```

Complete *Candidate Template* (EXX, Ex, and ExOut are program generated separators of the components of the *Candidate Template*):

```
[EXX 4, 11, 0, 2, BABY,
  Ex: 'SOL William , born',
  ExOut: 'p. 2 Aug. 1746 EOL SOL William , born 23 June1747 .']
```

10. Go to Step 3 for another edit-and-search cycle, or to Step 11 to complete the process.
11. Assemble from the final *Match List* the *Family Groups* between consecutive HEADs and write them into the Results Out file, e.g.:

```
['HEAD', 4, 6, 1, 1, 5, 'Adam', ',', 'James', 'WIFE', 4, 6, 6, 2, 3, 'Jannet',
'Bannatyne', 'GEO', 4, 6, 10, 7, 2, 'Hair', 'BABY', 4, 7, 1, 13, 4, 'James',
'DATE', 4, 7, 4, 3, 3, '15', 'Dec.', '1672', 'BABY', 4, 8, 1, 13, 4, 'Robert',
'DATE', 4, 8, 4, 3, 3, '15', 'Oct.', '1676', 'BABY', 4, 9, 1, 13, 4, 'Margaret',
'DATE', 4, 9, 4, 3, 3, '6', 'April', '1679']
```

Steps 1 and 2 are executed only once (the initialization in Fig. 2). Step 3 is either the user's initial selection of *Search Phrases* and *Extracts* for each class or, on subsequent cycles, the user correction of the *Candidate Templates* and *Extracts* (the user-interaction loop in Fig. 2). Steps 4 and 5 construct and apply the templates (the first three trapezoidal boxes). Steps 6 to 9 generate new *Candidate Templates* (the larger green trapezoidal box). In Step 10 the user decides whether to edit the Candidate Templates or accept the current output. Step 11 constructs and emits the final results (the output in Fig. 2).

III. EVALUATION

We evaluated GreenQQ on two books, Kilbarchan [21] and Miller [22], generating the results shown in Tables II, III, and IV. The interactive editing sessions on successive batches of candidate templates, each ending with execution of GreenQQ.py, took less than half-an-hour each (two sessions for Kilbarchan and three for Miller).

TABLE II. EXPERIMENTS.

Book	Lines	Tokens	Classes	Templates	Matches	Runtime	User Time
Kilbarchan	9,464	89,391	5	40	17,206	5 s	~50 min
Miller	16,835	240,198	11	51	27,647	11 s	~95 min

For evaluating accuracy by inspection of selected output, GreenQQ generates the Check File of Fig. 3 for the selected pages, with three lines for every line of input text. The results in Tables III and IV were obtained from Check Files on pseudo-randomly selected pages 20, 41 and 123 of the 143 Kilbarchan book and pages 296, 318, and 363 of the 396 page Miller book. Results are reported using both Soft and Hard scoring rules. We judged an extract 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. Common causes of errors are discussed at the end of this section.

Post-extraction, GreenQQ formed family groups by consolidating consecutive extracts between identified family starting anchors found in the text—fathers in Kilbarchan as is clear from Figure 1a and deceased persons in Miller as seen in Figure 1c. In both Kilbarchan and Miller, we chose to label these anchors as HEADs. Retaining the comma differentiates last-name/,/given-name from first-name/last-name for spouses. The generated family-grouping files were about 500KB in size for Kilbarchan and 1MB for Miller.

As it so happened, the GreenQQ run over the test pages for both Kilbarchan and Miller missed one family HEAD.

```
page 4  line 6
"  SOL Adam, James, and Jannet Bannatyne, in Hair Lavvis, 1676
EOL"
"  SOL HEAD HEAD HEAD , and WIFE WIFE , in Hair Lavvis ,
1676 EOL"
"  SOL T0 T0 T0 , and T1 T1 , in Hair Lavvis , 1676 EOL"

page 4  line 7
"  SOL James, 15 Dec. 1672. EOL"
"  SOL James , DATE DATE DATE . EOL"
"  SOL James , T2 T2 T2 . EOL"

page 4  line 8
"  SOL Robert, 15 Oct. 1676. EOL"
"  SOL Robert , DATE DATE DATE . EOL"
"  SOL Robert , T2 T2 T2 . EOL"

page 4  line 9
"  SOL Margaret, 6 April 1679. EOL"
"  SOL Margaret , DATE DATE DATE . EOL"
"  SOL Margaret , T2 T2 T2 . EOL"

page 4  line 10
"  SOL Adam, James, in Kilbarchan, and Jane Lyle p. 2 Aug. 1746
EOL"
"  SOL HEAD HEAD HEAD , in Kilbarchan , and WIFE WIFE p.
DATE DATE DATE EOL"
"  SOL T0 T0 T0 , in Kilbarchan , and T1 T1 p. T2 T2 T2 EOL"
page 4  line 11

"  SOL William, born 23 June 1747. EOL"
"  SOL BABY , born 23 June 1747 . EOL"
"  SOL T3 , born 23 June 1747 . EOL"
```

Figure 3. Fig. 3 GreenQQ output for checking its accuracy on selected pages. The first line in each group is from the text file after insertion of the line markers SOL and EOL. The second line shows the class assigned to each token (in line 6 Hair Lavvis, was not labeled because this initial run had no "in" GEO template). The third line displays the source template responsible for each match.

TABLE IV. KILBARCHAN ACCURACY RESULTS.

					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

TABLE III. MILLER ACCURACY RESULTS*.

					Soft: Correct = Correct + Partial			Hard: Incorrect = Partial + Incorrect		
Class	Total	Correct	Partial	Incorrect	Recall	Precision	F-score	Recall	Precision	F-Score
HEAD	30	29	0	0	0.97	1.00	0.98	0.97	1.00	0.98
SPOUSE	15	9	3	0	0.80	1.00	0.89	0.60	0.75	0.67
B_DATE	22	15	7	1	1.00	0.96	0.98	0.68	0.65	0.67
D_DATE	30	30	0	0	1.00	1.00	1.00	1.00	1.00	1.00
FATHER	22	13	8	0	0.95	1.00	0.98	0.59	0.62	0.60
MOTHER	20	10	4	0	0.70	1.00	0.82	0.50	0.71	0.59
AGE	26	24	1	0	0.96	1.00	0.98	0.92	0.96	0.94
BUPD	29	8	12	2	0.69	0.91	0.78	0.28	0.36	0.31
NRGRANDCH	8	7	0	0	0.88	1.00	0.93	0.88	1.00	0.93
Overall	202	145	35	3	0.89	0.98	0.94	0.72	0.79	0.75

*Two classes, B_PLACE and DATE, are not included because their extracts were sometimes superseded by the extracts of other classes and therefore not retained in the Check File.

The missed HEAD caused two families to be grouped together as one—a precision error. And it caused both of the grouped families to be misidentified—two recall errors. Thus, for the 72 Kilbarchan families, the Recall is 0.97, the Precision is 0.99, and the F-score is 0.98, compared to the F-score of 0.95 obtained with REGEX in [23]. The corresponding results for the 30 Miller families are 0.93, 0.97 and 0.95.

Precision results for Kilbarchan are near perfect except for the GEO class and are also high for Miller (Soft). Not only does this bode well for genealogical applications, it also means that GreenQQ could be a good candidate for semi-automatic labeling of training data for machine learning. (See [24] and [25] as representative examples of the work being done in this area.)

Recall results vary depending on the complexity of the template. BUPD is by far the most complex, having a Hard Recall of only 0.28, although the Recall jumps to 0.69 when we consider partials as being correct. Many of the multi-line BUPD entries were partial only because the current program does not process templates that cross line boundaries. Some of the lower Recall errors in Miller were caused by not creating templates to accommodate some common OCR errors. For example, mothers are identified by "m" , but the OCR often recognized "rn" as "rn" ("r" followed by "n"). The HEAD missed in Miller was also caused by an OCR error in which "SHUMAKER" was rendered as "SlillMAKER". The HEAD missed in Kilbarchan was caused by a combination of a typesetting error and an OCR error: "Uwing, John" was typeset as "Uwing,John" and OCR'd as "Uwingjohn".

GreenQQ reprocesses every previous template on each run, so templates can be added or corrected any time. Some of the many hurdles that we encountered are:

- OCR errors like born/bom cem/cern 1/I/l/!/] J/j 0/O ,/. ./- that need additional templates.
- Incorrect re-rendering of OCR output as a sequence of text tokens: spaces before punctuation and conjoined words such as "Annejordan" instead of "Anne jordan".
- Page headers and footers, like PARISH OF KILBARCHAN and REGISTER OF MARRIAGES, 1649-1772. Templates must be specific enough to avoid matching their contents.
- Irrelevant pages, like *Foreword, Copyright notice. Acknowledgment*. The program has provisions for specifying the first and last pages to be processed.
- Tokenization errors. For instance, "Nov." become a single token, but "Mar." yields two tokens because "mar" is an English word. This augments the number of templates.
- Proper names that are also the names of months. This is apparently only a springtime phenomenon: *April*, *May*, *June*. We have not yet encountered a child christened *October* or *November*.
- Family names that are also geographic locations, such as *Paisley,* which can result in mislabeled Candidate Templates.
- Inconsistent punctuation. In Kilbarchan, most, but not all, of the family heads are listed with a comma after their last name. Periods are often dropped at the end of a line. Parentheses are used somewhat arbitrarily.

- Missing items. Spouses' last names are occasionally omitted. The day of the month for some births is sometimes missing, and only occasionally labeled with *n.* or *none* or *unknown*.

- Unexpected additional information: *known as, brother of, presented by, jun. natural, born in adultery, Mr., minister,* .... Such words and phrases require additional care in formulating templates and extracts.

IV. CONCLUSION

This work has application not only to family history and its support of medical, intergenerational economic, community demographics but also to semi-automatic labeling of training data. We believe that effective user interaction will be vital for the rule-based information extraction systems that are poised to assume an even more dominant role in the market when combined with machine-learning.

The Hard F-score's 0.98 for Kilbarchan and 0.75 for Miller are respectable considering the irregularities hidden under the seemingly uniform appearance of these books. Since the evaluations were conducted by ourselves, they are necessarily suspect. However, we posted our page image and text files at http://tango.byu.edu/data/ and we will gladly make our code (about 1500 lines of python) and output available for any non-commercial research. Since our results have been improving weekly since the DAS submission date of November 2017, we plan to delay public posting of the dozens of input and out files till GreenQQ stabilizes.

With a view to transferring the technology to publishers of genealogical data, we are currently improving the code to bridge line/page ends, compute additional features, and give users a convenient clickable interface. A parallel effort aims to extend the analysis of the extracted items to interfamily relationships.

REFERENCES

[1] FamilySearch, https://www.familysearch.org.

[2] National Archives, https://www.archives.gov/research/genealogy.

[3] G. Salton, *Automatic Information Organization and Retrieval*, McGrawHill 1968.

[4] Text Retrieval Conference (TREC), http://trec.nist.gov.

[5] Stanford Named Entity Recognizer (NER), https://nlp.stanford.edu/software/CRF-NER.shtml.

[6] D.B. Searls and S.L. Taylor, Document Image Analysis Using Logic-Grammar-Based Syntactic Pattern Recognition, in *Structured Document Analysis*, H.S. Baird, H. Bunke, K. Yamamoto (Eds.), Springer Verlag, 1992, 520-545.

[7] N. Kushmerick, D.S. Weld, and R. Doorenbos, Wrapper Induction for Information Extraction, *Proceedings of the 1997 International Joint Conference on Artificial Intelligence*, 1997, 729–735.

[8] D.J. Ittner and H.S. Baird, Programmable Document Analysis, *Proceedings of the First IAPR International Workshop on Document Analysis Systems*, DAS'94, A.L. Spitz and A. Dengel (Eds), World Scientific 1995, 76-93.

[9] Belaïd and Y. Chenvoy, Document Analysis for Retrospective Conversion of Library Reference Catalogues, *Proc. ICDAR'97*, Ulm, Germany, 1997.

[10] J. Turmo, A. Ageno, and N. Català, Adaptive Information Extraction, *ACM Computing Surveys*, 38:2, 2006.

[11] S. Sarawagi, Information Extraction, in *Foundations and Trends in Databases*, 1:3, 2008, 261–377.

[12] R. Grishman, Information Extraction, *IEEE Intelligent Systems*, 30, Sept.-Oct., 2015, 8–15.

[13] P. Jiménez, R. Corchuelo, and H.A. Sleiman, ARIEX: Automated Ranking of Information Extractors, *Knowledge-*Based Systems, 93:2, 2016, 84–108.

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

[15] T.L. Packer and D.W. Embley, Unsupervised Training of HMM Structure and Parameters for OCRed List Recognition and Ontology Population, Proceedings of the 3rd International Workshop on Historical Document Imaging and Processing, Nancy, France, 22 August 2015, 23–30.

[16] D. Schuster et al., Intellix -- End-User Trained Information Extraction for Document Archiving, *Proc. ICDAR'13*, Washington DC 2013.

[17] Sutherland, S., Learning Information Extraction Rules for Semi-structured and Free Text. *Machine Learning*, 34, 1999, 232-272.

[18] K. Taghve, T.A. Nartker, and J. Borsack, Information access in the presence of OCR errors. Procs. ACM Hardcopy Document Processing Workshop, , Washington, D.C. Nov 2004, 1-8.

[19] L. Chiticariu, Y. Li, and F.R. Reiss, Rule-based Information Extraction is Dead! Long Live Rule-based Information Extraction Systems!, *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, Seattle, Washington, USA, October, 2013, 827–832.

[20] G. Nagy, Estimation, Learning, and Adaptation: Systems that Improve with Use, Proceedings of the Joint IAPR International Workshop on Structural, Syntactic, and Statistical Pattern Recognition, Hiroshima, Japan, November, 2012, 1–10.

[21] F.J. Grant (editor), Index to The Register of Marriages and Baptisms in the PARISH OF KILBARCHAN, 1649 –1772. J. Skinner & Company, LTD, Edinburgh, Scotland, 1912.

[22] Miller Funeral Home Records, 1917 – 1950, Greenville, Ohio, 1990.

[23] D. Embley, S. Liddle, T. Eastmond, D. Lonsdale, J. Price, S. Woodfield. Conceptual Modeling in Accelerating Information Ingest into Family Tree. In: J. Cabot, C. Gómez, O. Pastor, M. Sancho. (eds.) Conceptual Modeling Perspectives, Springer, Cham, Switzerland, 2017, 69–84.

[24] N. Kooli and A. Belaïd, Entity Local Structure Graph Matching for Mislabeling Correction, Proceedings of the 12th IAPR Workshop on Document Analysis Systems, Santorini, Greece, April 11–14, 2016, 257–262.

[25] I. Rehbein and J. Ruppenhofer, Detecting Annotation Noise in Automatically Labelled Data, Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, Vancouver, Canada, 30 July–4 August 2017, 1160–1170.