HIP '21, September 05, 06, 2021, Lausanne, Switzerland

ACM ISBN 978-1-4503-8690-6/21/09. . . $15.00
https://doi.org/10.1145/3476887.3476895
This is a pre-publication version

Generalized Template Matching for Semi-structured Text

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

ABSTRACT

Conventional template matching for named entity recognition on book-length text strings is generalized by allowing search phrases to capture distant tokens. Combined with word-type tagging and format variants (alternative name/date formats), a few initial templates (class—search-phrase—extract-phrase triples) can label most of the significant tokens. The program then uses its book-length statistics of tag-label associations to suggest candidate text for further template construction. The method serves as a preprocessor for error-free extraction of semantic relations from text obeying explicit semi-structure constraints. On three sample books of genealogical records, an F-measure of over 0.99 was achieved with less than 3 hours' user time on each book.

CCS CONCEPTS

• Information systems • Document representation • Document structure • Content analysis

KEYWORDS

mask matching; tokenization; tagging; data formats; semi-structure

1 Introduction

We revisit generalized template matching for named entity recognition in family books. In contrast to machine learning and neural networks, this approach requires no training or validation data, explains every decision, and provides a natural path for incremental improvement without post-processing. We present recent improvements, answer some queries prompted by our earlier communications, and lay the foundations for our forthcoming report at ICDAR'21 on named relation extraction.

We were motivated to embark on this strand of research by a long-time colleague's participation in a much larger project at FamilySearch. That project's objective is the construction of a pipeline from existing sources – current databases, books, census, parish and military records, archival resources, even photographs of tombstones – to a "universal" genealogical ontology. Yet unexploited sources include over 300,000 already digitized family books (published compendia compiled from the records of a family, parish or town). They are typically a list of assorted facts like names, dates and places presented as quasi-repetitive phrases of keywords and targets with simple \ grammars.

The GreenEx method presented below aims at the rapid extraction of desired facts from entire semi-structured books. Tests on a test set of 18 random pages of 10,391 word tokens yield an average F-measure of 0.993 and Precision of 0.998. We are confident that these results are representative of the 819 pages (35,782 lines and 473,083 tokens) from the three books that were processed. Logged user time was under three hours and computer run-time was under one minute for each book. The method is applicable only to semi-structured text subject to explicit constraints.

2 Prior Work

This presentation is intended to bridge generalized template matching [1] and our report at the main conference on named relation extraction [2]. These papers contain extensive (and heavily overlapping) literature reviews, so here we mention only the most relevant items.

The Brigham Young University and FamilySearch team presented their research at the first HIP Workshop [3]. Their ontology-based system is more fully described in [4], which also proposes seven alternatives for fact and relation extraction from structured text: (1) Extraction Rule Creation with Data Frames; (2) Template Matching with hand-crafted Regular Expressions; (3) Construction of Regular Expressions from Form Filling Example (the REGEX template matching scheme of [5]); (4) Extraction Rule Learning by Text Snippet Examples (based on an early version of GreenEx [6, 7]); (5) Extraction Rule Learning by Text Pattern Discovery; (6) Machine Learning of Extraction Rules; and (7) Extraction Rule Creation by Natural Language Processing and Cognitive Reasoning. Perhaps because of FamilySearch's far broader objectives, we have not yet seen any results directly commensurable with ours.

Researchers from our community combined entity and relation extraction with handwriting recognition. In [8], a category-based language model is compared with a probabilistic finite-state machine model for labeling family roles in handwritten 17th

https://doi.org/10.1145/1234567890

Century Catalan marriage records. With a large fraction (6/7) of the 173-pages used for training, and seven-fold cross-validation, both methods yielded 70-80% Precision and Recall. Information extraction was also the topic of a 2017 ICDAR competition. Using neural networks and conditional random fields, the winning team from Harbin Institute of Technology achieved a character error rate of ~8% on the same database, but 100 of 125 pages had to be manually labeled for training and validation [9].

We reported the progress of GreenEx, on comparable test data, from F-measure = ~95% in [6] and [7] to 98% in [1]. The improvement was due mainly to the introduction of format variants. The major shortcoming in [1] was inflexible templates. The current communication shows that allowing extracts to "float" at arbitrary distance from their search phrase raises the average F-measure to 99.3%, using less than half as many templates. Although our focus is on user time, we also present an algorithmic twist that cuts runtime by 50%.

3 Method

GreenEx tokenizes the unicode OCR output and assigns a Sequence Number and a Tag to every token. Fifteen generic tags are based on the token's characters: type case, letter/numeric, punctuation, and hyphen. They are augmented by book-specific, user-assigned *literals* like `born` and `died` that serve as their own tag, and by built-in or user-specified *aliases* like `b. ≡ born`, `month ≡ {January, Jan., February …}`, `progeny ≡ {child, children, son, daughter, dau.}` or `prep ≡ {in, at, to, from…}`.

The user's primary task is the specification of a set of initial templates via the ClickEx GUI [1]. Each Template consists of a Search Phrase and an Extract Phrase that appear in the text, and a Class chosen by the user. The operator clicks on the first and last tokens of a pair of suitable phrases, and on a pop-up list of Class Labels. An Extract Phrase may be located in the text several lines beyond its Search Phrase. The GUI validates the click sequence (or requests correction), assigns a Template ID, stores the template, and logs the user actions. An example of a template is: `[T3; Parent1; son of; Elisha Mills Ely].`

GreenEx then tags the search and extract phrases of the templates. It also associates each extract phrase with a set of built-in genealogically oriented Format Variants (FVs) of proper names and dates. Format variants are automatically augmented by new tag sequences from templates added to their class. Fig. 1 shows an example of their flexibility. To monitor progress, the user may inspect any page or line of the book showing the text tokens, their tags, assigned classes and the responsible templates (Fig. 2).

```
b   Ohio
b   Carlisle  PA
b   anBure  Twp  Dke  Co  OH
b   14  Nov  1861  Miami  Co  OH
b   1  Sept  1913  Twin  Twp  Dke  Co  OH  EOL
b   30  July  1854
b   1823
```

Fig. 1. Some token sequences matched by format variants for birth dates and places. Two templates, one for birth date and one for birthplace, suffice to extract all these configurations.

The algorithm for template matching is based on the order constraints of semi-structured text. These are formally defined in [2] in terms of the signs and values sought to match a template's search and extract phrase. The constraints require that a sign and its value must be in the same family record; signs and values alternate; different classes cannot share the same sign and value; and if values conflict, the nearest preceding sign prevails.

The program finds every sequence number in the text where either a search phrase or a format variant (of an extract phrase) match the text, then lists the pairs of sequence numbers where the class of a variant and of the preceding search phrase match. The patient reader can follow these critical steps in Fig. 3 and in the simplified pseudocode of Fig. 4.

The initial templates usually yield 85%-95% recall with >99% precision. Rather than quit here, the operator may request recommendations for additional templates from the Autosuggestion routines.

page 576 line 40 first SeqNo 92930

"243356 . **Julia Ely Hyde** , Marion , Perry Co. , Ala . , b . **1824** , ' dau . of EOL"

"NUM6 . **CAP CAP CAP** , CAP , CAP CAP , CAP . , b . **YEAR** , PUNCT PROG . PREP EOL"

NN NN **HEAD: HEAD: HEAD**: NN NN NN NN NN NN NN NN NN NN NN B_DATE: NN NN NN NN NN EOL

nn nn **T_5 T_5 T_5** nn nn nn nn nn nn nn nn nn nn nn nn T_7 nn nn nn nn nn nn

page 576 line 41 first SeqNo 92953

"**Julia Ely and Zabdial Hyde** ; m . **1844** , **Alexander Clark Bunker** , who EOL"

"**CAP CAP** and **CAP CAP** PUNCT m . **YEAR** , **CAP CAP CAP** , who EOL"

PARENT1: PARENT1: NN **PARENT2: PARENT2**: NN NN NN **M_DATE**: NN **SPOUSE: SPOUSE: SPOUSE**: NN NN EOL

T_2 T_2 nn T_6 T_6 nn nn nn **T_9** nn **T_3 T_3 T_3** nn nn nn

Fig. 2. Two lines of text labeled by HEAD, BIRTHDATE, PARENT1, PARENT2, MARRIAGEDATE and SPOUSE templates. Matched tokens, tags, classes and template IDs are red boldface. NN and nn stand for None.

Adam , James , and Jannet Bannatyne , in Hair Lavvis , 1676 EOL
James , 15 Dec. 1672 . EOL
Robert , 15 Oct. 1676 . EOL
(a)

Sorted_SeqNo_SPandFV list (52339 rows)
SeqNo, [[FVclass, FVlength], [FVclass, FVlength], ...]
OR SeqNo, [[TemplateID, SPclass, SPlength], [TemplateID, SPclass, SPlength], ...]

```
EOL      [44, [[1, 'HEAD:', 4], [5, 'CHILD:', 3]]]
Adams    [45, [['HEAD:', 3], ['CHILD:', 1], ['M_PLACE:', 1], ['SPOUSE:', 1], ['TWINS:', 1]]]
James    [47, [['CHILD:', 1], ['M_PLACE:', 1], ['SPOUSE:', 1], ['TWINS:', 1]]]
,        [48, [[2, 'SPOUSE:', 3]]]
Janet    [50, [['CHILD:', 2], ['M_PLACE:', 2], ['SPOUSE:', 2], ['TWINS:', 2], ['CHILD:', 1], ['M_PLACE:', 1], ['SPOUSE:', 1], ['TWINS:', 1]]]
Bannatyne [51, [['CHILD:', 1], ['M_PLACE:', 1], ['SPOUSE:', 1], ['TWINS:', 1]]]
Hair     [54, [['CHILD:', 2], ['M_PLACE:', 2], ['SPOUSE:', 2], ['TWINS:', 2], ['CHILD:', 1], ['M_PLACE:', 1], ['SPOUSE:', 1], ['TWINS:', 1]]]
Lavvis   [55, [['CHILD:', 1], ['M_PLACE:', 1], ['SPOUSE:', 1], ['TWINS:', 1]]]
1676     [57, [['B_DATE:', 1], ['C_DATE:', 1], ['M_DATE:', 1]]]
EOL      [58, [[5, 'CHILD:', 3], [6, 'C_DATE:', 4]]]
Jaames   [59, [['CHILD:', 1], ['M_PLACE:', 1], ['SPOUSE:', 1], ['TWINS:', 1]]]
15 [61, [['B_DATE:', 3], ['C_DATE:', 3], ['M_DATE:', 3], ['B_DATE:', 2], ['C_DATE:', 2], ['M_DATE:', 2]]]
Dec.     [62, [['B_DATE:', 2], ['C_DATE:', 2], ['M_DATE:', 2], ['CHILD:', 1]]]
1672     [63, [['B_DATE:', 1], ['C_DATE:', 1], ['M_DATE:', 1]]]
EOL      [65, [[5, 'CHILD:', 3], [6, 'C_DATE:', 4]]]
Robert   [66, [['CHILD:', 1], ['M_PLACE:', 1], ['SPOUSE:', 1], ['TWINS:', 1]]]
15 [68, [['B_DATE:', 3], ['C_DATE:', 3], ['M_DATE:', 3], ['B_DATE:', 2], ['C_DATE:', 2], ['M_DATE:', 2]]]
```

(b)

Matched_SP_EP_List (30966 rows)
EPSeqNo, SPseqno, EPClass, EPlength, SPClass, TemplateID, SPlength]

```
[45, 44, 'HEAD:', 3, 'HEAD:', 1,  , 4]
[45, 44, 'CHILD:', 1, 'CHILD:', 5,  , 3]
[50, 48, 'SPOUSE:', 2, 'SPOUSE:', 2,  , 3]
[50, 48, 'SPOUSE:', 1, 'SPOUSE:', 2,  , 3]
[59, 58, 'CHILD:', 1, 'CHILD:', 5,  , 3]
[61, 58, 'C_DATE:', 3, 'C_DATE:', 6,  , 4]
[61, 58, 'C_DATE:', 2, 'C_DATE:', 6,  , 4]
[66, 65, 'CHILD:', 1, 'CHILD:', 5,  , 3]
[68, 65, 'C_DATE:', 3, 'C_DATE:', 6,  , 4]
[68, 65, 'C_DATE:', 2, 'C_DATE:', 6,  , 4]
```

(c)

Fig. 3 (a) Three book lines showing a HEAD (Adam, James), a SPOUSE (Jannet Bannatyne) and their first two children with their christening dates; (b) interspersed SPs and FVs sorted by sequence number, with their label, template ID, and phrase length; (c) selected (red) and unselected (black) same-label SP– FV pairs. The selected matches yield Extract Phrases. A token can be labeled only once.

The Autosuggestion routines locate (a) unmatched extract phrases between consecutive matched search phrases, (b) unmatched tokens adjacent to same-tag matched tokens, and (c) unmatched tag sequences identical to either matched or unmatched sequences surrounded by the same tags. The program exploits its book-length statistics to sort the text snippets for new template candidates according to their expected coverage of hitherto unlabeled text. Only the user's stamina limits further template construction, but on each book we stopped before reaching 20 templates (cf. Section 4). In earlier versions of GreenEx, 50-60 templates yielded inferior results).

The output of the program is a list of family records of Labels, PageNos, LineNos, Token-offsets and Values, e.g.:

HEAD:5,7,1 Adam, James SPOUSE:5,7,18 Jannet Bannatyne CHILD:5,8,1 James C_DATE:5,8,8 15 Dec. 1672 CHILD:5,9,1 Robert C_DATE:5,9,9 15,Oct. 1676 CHILD:5,10,1 Margaret C_DATE:5,10,11 6 April 1679

Experimental Protocol and Results

The three books that we processed, the Kilbarchan Register of Marriages and Baptisms [10], the Miller Funeral Home Records [11], and the Ely Ancestry [12], exhibit different arrangements and types of family factoids. The entities of interest account for fewer than half of the tokens in each book. Assigning a label other than NONE to any other token is a Precision error. Irrelevant tokens accounted for almost every one of the few Precision errors.

```
Algorithm LabelTokens(BookText)     % Generate a list of family records from a list of page-length strings of characters
%%% Tokenize book text and tag tokenized text, template SPs & EPs, and format variants: (SP≡SearchPhrase, EP≡ExtractPhrase)
TokText ←Tokenizer(BookText)        % NLTK.Tokenizer partitions the text into tokens,   e.g…b. 1753 , Hilltown …
TaggedText←Tagger(TokText)          % TaggedText is a parallel book-length list of tags, e.g. b. YEAR PUNCT CAP…
Read Templates                      % Templates[k] = [TempID, TempLabel, [SP], [EP]]; e.g. = [6, B_Date, [b.], [1753]]
Read Variants                       %  e.g. Variants[3] = [[B_Date, M_Date, D_Date], [[1753], [April 15 , 1750], [15 April 1750]]]
% Prepare lists of unique tagged SPs and format variants for efficient search:
TaggedSPs←SPSorter(ListTemplates)       % TaggedSPs lists the labels of templates with tagged SP = TagSP
                    % e.g. TaggedSPs[5]←[[b.], [[6, B_Date],[9, B_Place]]]
TaggedVariants←VariantSorter(ListVariants)  % TaggedVariants lists the (unique) variants and their labels
                    % e.g. TaggedVariants[8]←[[NUM, CAP, YEAR], [B_Date, M_Date, D_Date]]
%%% Find every location in the text where either a SP or an EP variant matches:
For x←1 to length (TaggedText):                 % for every tag in the tokenized text
    For s1←1 to length(TaggedSPs):              % for every distinct tagged SP
        For s2←1 to length(TaggedSPs[s1][0]):   % for every tag in this SP
            If TaggedText[x + s2] = TaggedSPs[s1][0][s2]]: break    % if this SP does not fit book tags here, try next SP
        End     For s2
        CandSeqNos.Append ([x, TaggedSPs[s1][1] ])                  % this SP fits here
                    % Add seqno, TemplateID and label; e.g. Candidates[323← [2245, [6, B_Date], [ 9, B_Place]]
    End     For s1
    For t1←1 to length(TaggedVaars)                                 % for every tagged variant
        For t2←1 to length(TaggedVariants[t1][0])                   % for every tag in this variant
            If TaggedText[x+ t2]=TaggedVariants[t1][0][t2]: break   % if it does not fit biotags here, try next variant
        End     For t2
        Candidates.Append ([x, TaggedVariants [t1][1]) ]            % this extract variant fits here
                    % Add seqno and labels to Candidates, e.g. Candidates[324 ←[2248, [B_Date], [M_Date], [D_Date]]
    End     For t1
End     For     x
%%% Make a list of sequence numbers where the class of a variant and the class of the preceding SP match
For y1←1 to length(Candidates):                 % for every seqno where a SP or a variant fit
    If IsInteger(Candidates[y1][1][0])          % i.e., if this is an SP (it has a template ID)
        For y2←1 to length(Candidates[y1][1]    % for every class whose SP fits here
            For y3←y1 to MaxDist                % for every candidate seqno starting at last SP
                If Candidates[y3] is a SP: break    % if no EP found before the next SP
                For y4←1 to length [Candidates[y2[1]    % for every class whose variant fits here
                    If Candidates[y3][1[y4] = Candidates[y1][1]y2]   % if class of variant = class of SP
                        MatchList.append([y3, Candidates[y3][1[y4] ])   % add this seqno and class to Match List
For y←1 to length(MatchList):                   % label every seqno listed in MatchList
    If LabeledText[SeqNo] = "NONE"              % if this token at this seqno  is not labeled yet
        LabeledText[SeqNo]← MatchList [y][1]    %then label it e.g. LabeledText[2248] ←B_Date
End     y
%%% Prepare list of families of labeled groups of tokens for output file:
FamilyRecord←GroupLabels(LabeledText)           % group EPs with identical consecutive labels and
                                                %   add label and Page-Line-Offset to each group
    E.g. [[Head, P4 L5 O1, Adams, James], [Spouse P4 L5 O5 Janet Leigh], [Child P4 L6 O1]], [[Head, P4 L7…,
Return(FamilyRecord)
```

Fig. 4 Pseudo-code for Generalized Template Matching

Most typesetting or OCR search-phrase errors, like `h` for `b`, (or `bom` for `born`) are detected by Autosuggestions and fixed by adding an alias. Transcription errors in extracts, like 164S are detected, but we cannot tell if it should be 1648 or 1645.

I started on each book with a minimal number of templates constructed from a single page. Then I added templates with Autosuggestions until each new template yielded only one or two matches. The program processed the entire book, except front and back pages without family lists. Precision and Recall on the Test Set (the pages for which I had ground truth) was evaluated by a separate program, but I inspected every flagged error and reject. There were too few for credible analysis. Adding more templates with Autosuggestions, or trying alternative algorithms, would be futile without far more ground truth because one could not make statistically valid estimates of even higher Precision and Recall.

Table 1 summarizes the extraction results on the three books and the precision and recall on the corresponding test data. Fig.5 graphs the increase in precision and recall on the Kiilbarchan test set as I added new templates from elsewhere in the book. It appears that one generalized template per class can cover the bulk of the book. Table 2 lists the recorded user-time taken by the various tasks on the same book. The addition of format variants and Autosuggestions expedited the template construction that took most of the time in earlier versions of GreenEx (Fig. 6).

GreenEx generates output in several formats for diagnostics, experiment records, validation against ground truth, and relation extraction. I used Python's hashed dictionaries to manipulate lists of tokens, tags, templates, variants, literals, aliases and book coordinates. Execution time is approximately proportional to the product of the length of the text and the number of templates. Runtimes on a 2.4-GHz Dell Optiplex 7010 ranged from 15 to 50 seconds per book. A zip-file of all of our experiment records, as well as the python and VBA code, are available on request to educators and researchers.

Table 1. Data and Results.

Book	Kilbarchan	Miller	Ely
pages	139	389	301
lines	9296	16040	10446
tokens (inc. EOLs)	82577	239135	151371
templates	17	18	14
labeled tokens	39203	91633	39440
family records	14789	15674	3379
classes	8	10	8
Test Data			
pages	6	6	6
lines	356	268	243
tokens (inc. EOLs)	3126	3842	3423
Precision	0.999	0.996	0.997
Recall	0.972	0.988	0.980
F-measure	**0.985**	**0.992**	**0.988**

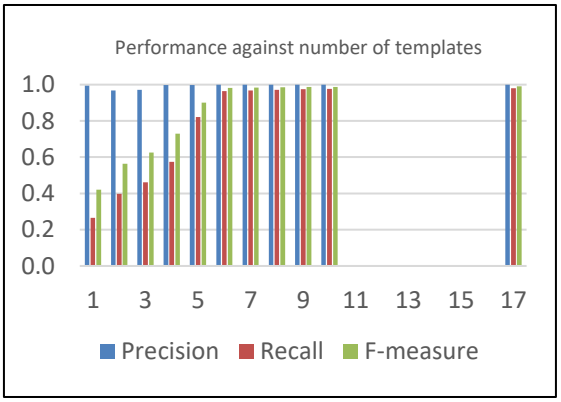

Fig 5. Increase in precision, recall and F-measure with templates. The last seven templates were added with the evaluation program run on the test set only at the end.

Table 2. User-time (minutes) by task (Kilbrchan)

Task	Time	Notes
Browse new book	40	Locate common constructs and front/back pages w/o data
Cut-and-paste directory and files names	5	Home directory, Text-files. Out directory, ClickEx
Initialize ClickEx and select a representative page	22	Enter (11) literals and (12) aliases
Compile (9) place format variants (common date & name variants are already in GreenEx))	14	Additional format variants from templates may be added automatically in the next two steps
Enter one template for each class from selected page	9	Logged by ClickEx
Enter (13) additional templates from auto-suggestion routines	16	Logged by ClickEx
Edit (2) templates based on book-length template-usage statistics	12	Logged by ClickEx
Re-run GreenEx to generate output files	5	Logged GreenEx run-time: 32 seconds.
TOTAL	**123**	Totals for other books within 20%

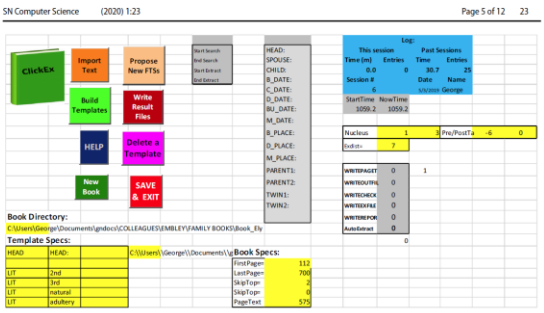

Fig 6. Snapshot of ClickEx GUI, from [1], added at a referee's suggestion. After launching a page-text display, each template requires 2 clicks for the SP, 2 for the EP, and 1 for the class.

5 Conclusion

Much of my career was devoted to neural networks, statistical pattern recognition and machine learning. So these struck me as the obvious approaches to information extraction from genealogical books. It was a major surprise that adding a few NLP twists to template matching, and providing instant user guidance by statistical analysis of partially processed text, would let such a mundane approach leave long short-term memory recurrent networks and other resource and data intensive machinery as unpromising alternatives.

One argument in favor of interaction instead of training is that a few user clicks can resolve ambiguities like `m.` for "mother" in one book vs. `m.` for "married" in another. Any neural network would require massive and perhaps customized training data. Another appealing aspect of interactive processing is its dependable, if increasingly sluggish, approach to perfect recall. This contrasts with the "take it or leave it" termination of most machine-learning algorithms.

The experiments show that generalized template matching requires only modest user effort for near-complete recall of facts from semi-structured books. The same results could perhaps be reached, albeit with far more effort, using regular expressions. The relation between generalized template matching and REGEX is analogous to that between a high-level programming language and machine code. The essence of the difference is the nested search on tagged text illustrated in Fig. 3.

Template matching on family books produces easily understood results, but it is a long-tailed process. Some typesetting anomalies or OCR errors require a new template or alias. It could take dozens of additional templates to halve the remaining 0.1% - 0,3% missing labels. Without information beyond the page-text files, one can only spot, but not automatically correct, typesetting and OCR errors. Fortunately, most of the tags, literals, aliases and format variants depend only on the domain rather than each book.

It is not obvious that complete manual key entry would produce better results, and it would be equally difficult to verify. In operational application, these issues can be addressed through downstream consistency checks (for missing, redundant or inconsistent dates and names), and by agglomerating results from multiple sources.

Before pressing on to other scripts, languages and applications (like industrial parts catalogs and municipal directories), we should find out what fraction of genealogical works "in the wild" conform to the semi-structure constraints that open the door to systematic processing. More OCR errors would increase the interaction necessary for high recall. It should, however, barely affect precision because a search phrase and its extract phrase will seldom fail in a complementary way (cf. the high precision at low recall in Fig. 5). Given the amount of digitized material awaiting processing, I am hesitant to follow a referee's request to prognosticate about handwriting.

We must also confirm that the skill level necessary for interacting with ClickEx and GreenEx does not exceed that of most genealogical software users.

ACKNOWLEDGMENTS

I am grateful for sustained help to my decades-long friend and collaborator, BYU Emeritus Professor David W. Embley, and to his expert colleagues at BYU and FamilySearch. This paper also benefited from the referees' thoughtful suggestions.

REFERENCES

[1] G. Nagy, Green information extraction from family books. Springer Nature Computer Science, Volume 1, Issue 1, January 2020.

[2] G. Nagy, Near-perfect Relation Extraction from Family Books, 16th ICDAR, Lausanne, accepted, 2021.

[3] D. W. Embley, T. Packer, J. Park, A. Zitzelberger, S. W. Liddle, N. Tate, D. W. Lonsdale,, Enabling search for facts and implied facts in historical documents, Proc. Workshop on Historical Document Imaging and Processing, 59–66, Beijing, China 2011.

[4] D.W. Embley D.W. Lonsdale S.N. Woodfield, Ontological Document Reading: An Experience Report, Enterprise Modelling and Information Systems Architectures: International Journal of Conceptual Modeling, Vol. 13, 133–181, 2018.

[5] T. Kim. A green form-based information extraction system for historical documents. MA thesis, Brigham Young University, Provo, Utah, 2017.

[6] D.W. Embley and G. Nagy, Extraction rule creation by text snippet examples, Family History Technology Workshop, Provo, UT, 2018.

[7] D.W. Embley and G. Nagy, Green interaction for extracting family information from OCR'd books, Document Analysis Systems Workshop (DAS'18), Vienna, Austria April 2018.

[8] V. Romero, A. Fornes, E. Vidal, and J.A. Sanchez, Using the MGGI Methodology for Category-based Language Modeling in Handwritten Marriage Licenses Books, Procs. 15th International Conference on Frontiers in Handwriting Recognition, Shenzhen, China (2016).

[9] 9 A. Fornes, V. Romero, A Baro, J.I. Toledo, J.A. Sanchez, E. Vidal, J. Llados, ICDAR 2017 Competition on Information Extraction in Historical Handwritten Records, Proc.s 14th IAPR International Conference on Document Analysis and Recognition, Kyoto, Japan (2017).

[10] 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.

[11] Miller Funeral Home Records, 1917–1950, Greenville, Ohio. Darke County Ohio Genealogical Society, Greenville, Ohio, 1990.

[12] G. Vanderpoel The Ely Ancestry: Lineage of RICHARD ELY of Plymouth, England. The Calumet Press, New York, NY 1902.