Green Information Extraction from Family Books

George Nagy, Professor Emeritus, RPI, 110 8th Street, Troy, NY 12180 USA nagy@ecse.rpi.edu

Abstract There is rising demand for the retrieval of genealogical information from semi-structured books. Nuggets of personal interest are currently transcribed piecemeal by volunteers. GreenBook is an effective alternative for recovering most of the desired information from any of the hundreds of thousands of books of ancestral records that have already been scanned and digitized. It minimizes human intervention by letting the user benefit from automatically compiled statistics from each book, and lets computer search benefit from user insights. GreenBook combines enhanced template matching with spreadsheet-based interaction for the rapid specification of the text to be extracted. The accuracy and completeness of the extracted information is limited only by the user's stamina. About three hours of user interaction yielded 99% precision and 97% recall on books of Scottish birth and marriage records, Ohio funeral home records, and family history spanning 300 years. The system is designed to facilitate transition to new books.

1. INTRODUCTION

The preservation of information about one's ancestors appears to be a common desire, perhaps even a duty, in almost every culture. Until recently, family chronicles were recorded on conventional media. Genealogical data is also available from census, court, voter roll, church, cemetery, funeral parlor and military s service records, and DNA. Most of the printed information about family ancestry was compiled from handwritten or typewritten material. Several hundred thousand digitized books contain family records of parents, spouse(s) and children with the dates and places of their birth, marriage and death. Extraction of these entities and relations generally relies on volunteers working from home. Since extracting all information from a whole book is so time-consuming, volunteers usually extract and contribute only details relevant to their own interests. Possibly disparate facts about any individual may appear in several collections. The National Archives, FamilySearch and Ancestry.com, among other organizations, integrate data from multiple sources to facilitate the construction of comprehensive family trees.

The objective of this project is the development of interactive software that reduces the time required to extract (almost) all useful information from a whole book to a few hours. *GreenBook,* a template-matching program with a graphical user interface (GUI), was designed and implemented for this purpose. The method was evaluated on three books: *KBook*: Kilbarchan Parish Record 1649-1772 from Scotland [1]; *MBook*: Miller Funeral Home Records, 1917 – 1950 from Greenville, OH [2]; and *EBook*: The Ely Ancestry ~1659-1900, mostly N-E USA [3].

The rest of the article is organized as follows. After listing our putative contributions, we review relevant prior work. Section 3, Methodology, explains and illustrates each step of the process. Section 4 reports the performance of GreenBook. Section 5 is presents our rationale for the choice of template matching for the designated task. The Conclusion summarizes our results, points out where we came up short, and mentions possible applications beyond genealogy.

1.1 Contributions

GreenBook is a viable solution to a problem that has long hampered the genealogical information-extraction community. The GreenBook software consists of *GreenEx.py*, a python template-matching program, and *ClickEx*, a spreadsheet-based graphical user interface that invokes GreenEx via the Windows Command prompt. Noteworthy aspects of this package are:

1. GreenBook's design minimizes and simplifies human labor while keeping the user in charge. Rapid search promotes user reactions to whole-book results. GreenBook avoids frustratingly repetitive intervention. After most of the useful information has been extracted by a dozen user-built templates, GreenEx exploits the results to autonomously extract some missed text configurations.

2. Based on statistics collected by the program, GreenEx suggests to users the most relevant unextracted text sequences for building additional templates. The context for proposed text snippets and the number of snippets to be displayed are chosen by the user.

3. An extensible word-type tagger was constructed for quasi-repetitive non-sentential text that is characteristic of family books and directories. The tagger is based on simple built-in python functions. It is faster and more easily customized than previous taggers for family books based on character-level regular expressions.

4. A multi-string template matching scheme was devised specifically for semi-structured text. The user designates queries (i.e., search phrases) and the desired extracts from text pages or from proposed text snippets. GreenEx adds query-specific data frames to extract variable configurations of tokens (for instance person names with one or several given names and initials, or dates in different formats). Separate windows with arbitrary offsets or overlaps between query and extract offer an improvement over conventional single-window template matching.

5. We built a clickable graphic user interface instead of key entry for template construction. ClickEx checks every user interaction and allows for immediate or delayed correction of errors. Help panels and requests for confirmation provide guidance to inexperienced users.

This combination of design features is largely responsible for reducing the processing time for a 400-page book from weeks to hours.

2. PRIOR WORK

We first trace progress on information extraction, then cite contributions directly relevant to our work. Research on automated information retrieval (IR) was put on a firm footing by Salton [4]. Natural Language Processing (NLP) and IR researchers' sustained interest in Named Entity Recognition (NER) on free-running text is exhibited by the enduring popularity of the NIST-sponsored Text REtrieval Conference [5] and of the major supportive software web sites such as the Stanford CoreNLP NER site [6]. In 1992, Searles and Taylor described a rule-based system created for address block extraction from text strings [7], and soon thereafter wrappers were introduced to extract information from commercial web sites [8]. Early researchers within the database, library/information science, document analysis, and AI communities also applied their diverse toolkits to search *semi-structured* documents like dictionaries

[9] and library catalogs [10]. Surveys of the hundreds of ensuing contributions include [11, 12, 13] and [14], which also evaluates and compares several dozen information extraction systems.

For the reader eager to explore the deep waters of ontology, we recommend [15], which presents a thorough review of the most recent relevant research as well as some experiments on the same books as we used. The authors describe a pipeline with form-based user-entry and seven extraction tools, for possible integration into FamilySearch's *Family Tree* [16]. They report "fully automatic extraction run[s]" for KBook and MBook and for some pages of EBook. We will cite their results on Section 4. An innovative proposed ontological application from the same BYU research team is deep data cleaning [17].They demonstrate automated discovery of errors (like inconsistent dates) in sources of data—including, among others, EBook.

Several earlier research endeavors relate to various aspects of the work presented here. We found reports of recovering family information from OCR'd obituaries [18] and from lists abstracted from family books [19]. Tagging with the well-known *Stanford Named Entity Recognizer* [20] yielded only mediocre results because it was really designed for sentence analysis. We adopted forty-year old data frames [21]. Our example-based approach for user interaction has some similarities with the end-user-provided training examples used commercially for scanned business documents [22]. Some aspects of our templates, like the use of literals and semantic tags, were anticipated in [23]. The effects of OCR errors on information extraction were discussed in [24]. We reviewed recent trends in document analysis in [25]. The rule-based extraction tool we present here is an example of the research called for in [26], 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, ours exploits the quasi-repetitive format of factoids in semi-structured text to generate and execute extraction rules. An early version of GreenEx was introduced at two workshops [27,28]. The GreenBook sobriquet predates the admirable current movie. It is *Green* because, like other "green" systems [29], its interactive feedback loop avoids wasting user energy.

3. METHODOLGY

Section 3.1 defines some common terms as used in this article, §3.2 illustrates the interaction between user, ClickEx.xlsm and GreenEx.py, §3.3 describe ClickEx, §3.4 traces tagging, template matching and conflict arbitration in GreenEx, §3.5 presents two routines for autonomous labeling, §3.6 explains the compilation of the Family Records output, and §3.7 adds a few remarks about programming considerations.

3.1 Definitions

Token: A string of alphanumeric symbols. E.g. *Adam* or *,* or *1762* (i.e., 'Adam', or ',', or '1762')
Token Sequence: A sequence of contiguous tokens. E.g. (*Adam , James , and Jannet*)
Tag: Element of a set of token types. E.g. CAP (capitalized) or NUM (numeric) or YEAR (year)
Literal: An alphanumeric string that serves as its own tag. E.g. (*and*→and) or (*born*→born) or (*,*→,)
Alias: A pair of identically processed synonyms. E.g. *(b*; *born*) or (*spouse*; *sp*) or (*Nov*; *November*)
Tag Sequence: A sequence of contiguous tags. E.g. (*Adam , James , and Jannet*) → (CAP , CAP and Cap)

Sequence Number (SeqNo): A unique integer assigned to a token. E.g. Token(50) = *Jannet*; Tag(50) = CAP
Class: element of a set of labels for tokens. E.g. SPOUSE or CHILD or B_DATE or B_PLACE
Label: A Class assigned to a specific token. E.g. Label(50) = SPOUSE
Frame: A preset Tag Sequence. E.g. (NUM CAP , YEAR) for dates
Frame Set: A set of Frames associated with a Class. E.g. [B_DATE (NUM CAP , YEAR), (YEAR)]
Query: A Tag Sequence associated with a Class. E.g. [(born); B_DATE]
Extract: A Tag Sequence associated with a Query. E.g. [(born); B_DATE] → (NUM CAP , YEAR)
Template: A user-specified pair of Query and Extract. E.g. [(born); B_DATE, 1, (NUM CAP , YEAR)]
Template ID: A unique integer identifying a Query. E.g. ExNo([(born); B_DATE, 1, (NUM CAP , YEAR)]) = 4

3.2 User, ClickEx, GreenEx Interaction

The symbiotic relationship between a user and our software is illustrated in Fig. 1.

User →	**ClickEx.xlsm** ←→	**GreenEx.py** →	**Output**
Initiates & ends session	Control buttons for GreenEx functions	Imports book text text page(s)	family records file tuples file
Specifies book file location page numbers data frames literals aliases keys	Entry fields for literals aliases keys page numbers display parameters	Tokenizes & tags book text text page(s) data frames templates	check file program and user logs evaluation file
Builds templates from text page(s) freq. tag sequences	Displays clickable text page(s) freq. tag sequences class selection box current templates	Labels book text with user-built templates AutoQuery AutoExtract	report file
Approves / disapproves each template output files	Displays immutable help panels error messages confirmation requests user log	Proposes frequent sequences	
Deletes incorrect templates redundant templates		Compiles family record groups tokens lists relational tuples	

Figure 1. The User, ClickEx and GreenEx interact to generate the desired outputs.

Every session is likely to be different. GreenEx logs only the time and the number of clicks. The following is a short yet plausible sequence of actions by the user and the programs:

1. User opens a fresh copy of ClickEx in Excel.
2. enters file location of KBook text in ClickEx
3. specifies some book-specific literals and aliases and approves default display parameters
4. asks ClickEx to display Page 4 text for building templates
5. ClickEx requests GreenEx to locate and import Page 4 text

6. GreenEx tokenizes Page 4 text and passes it to ClickEx for display
7. User builds and approves templates for HEAD, SPOUSE, M_DATE, M_PLACE, CHILD, B_DATE
8. commands ClickEx to generate Output files
9. ClickEx passes the request and the current templates to GreenEx
10. GreenEx tokenizes and tags the book text and the current templates
11. sweeps the templates against the book text and finds all matches in the book text
12. labels the book text, groups same-class tokens, and generates output files
13. User inspects the Check File and requests frequent tag sequences (FTSs) for Key "twin"
14. GreenEx complies and passes the FTSs including the token "twin" to ClickEx for display
    (KBook reports 49 sets of twins, but none on Page 4)
15. User builds templates for TWIN1 and TWIN2 and requests new output
16. inspects the Family Record file, is satisfied, and presses SAVE & EXIT button on ClickEx
17. ClickEx saves page text, frequent tag sequences, templates and log, then exits Excel.

3.3 ClickEx Graphic User Interface

The ClickEx GUI provides a means of rapid and accurate templates construction and full control of every step in the information extraction procedure. The ClickEx dashboard is shown in Fig. 2. It lets the user request a new page to be loaded into the Edit Sheet for template generation, build templates using the Edit Sheet (Fig. 3), delete a templates via the Selection Sheet (Fig. 4), write selected output files, quit and later resume the session, invoke a Help file, or start work on a new book. Except for entering LIT and ALIAS tokens, and page numbers, all user interaction is point-and-click.

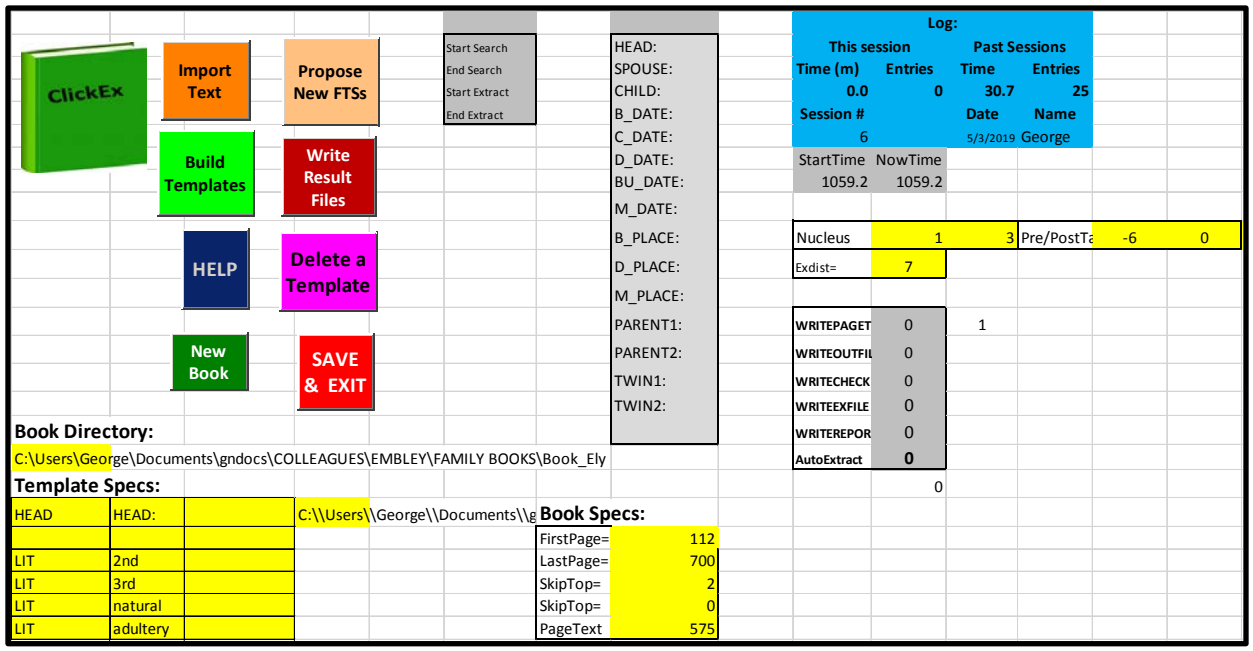
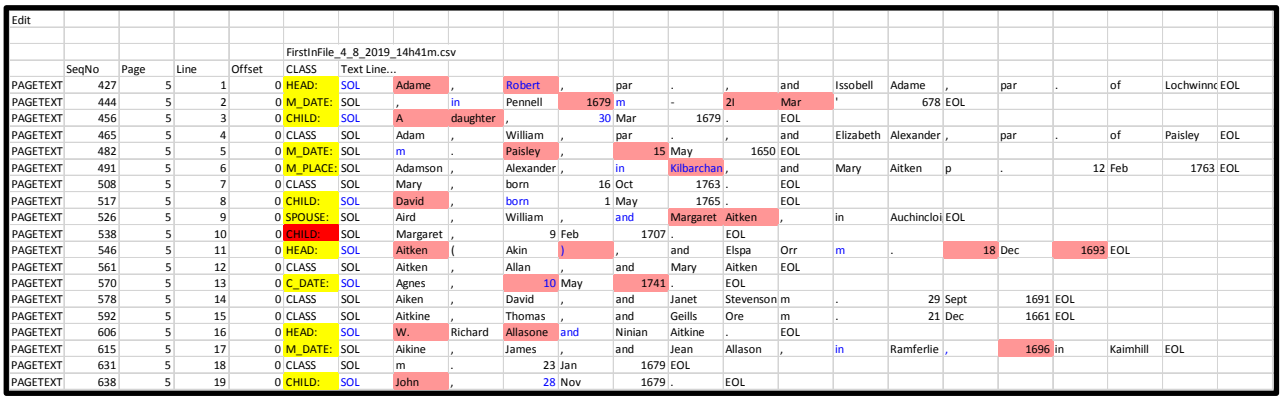


Fig. 2 ClickEx Dashboard. Used to enter initial parameters and select the next action. Import Text would load page 575 (under Book Specs). Write Result Files gives the user the option of writing out any or all output files. The gray cells are used internally by ClickEx.

The class of a new template is selected from a pop-up CLASS box similar to the gray box at the top of the dashboard. Then a query and extract are selected by clicking on their first and last tokens in the text on the Edit Sheet (Fig. 3). The user can abandon a partially constructed template after each click, and is requested to confirm (or reject) the specified class, query and extract before being asked whether to continue building templates or return to the dashboard.



Edit

FirstInFile_4_8_2019_14h41m.csv

	SeqNo	Page	Line	Offset	CLASS	Text Line...																
PAGETEXT	427	5	1	0	HEAD:	SOL	Adame	,	Robert	,	par	.	,	and	Issobell	Adame	,	par	.	of	Lochwinnc	EOL
PAGETEXT	444	5	2	0	M_DATE:	SOL	,	in	Pennell	1679	m	-	2I	Mar	'	678	EOL					
PAGETEXT	456	5	3	0	CHILD:	SOL	A	daughter	,	30	Mar	1679	.	EOL								
PAGETEXT	465	5	4	0	CLASS	SOL	Adam	,	William	,	par	.	,	and	Elizabeth	Alexander	,	par	.	of	Paisley	EOL
PAGETEXT	482	5	5	0	M_DATE:	SOL	m	.	Paisley	,	15	May	1650	EOL								
PAGETEXT	491	5	6	0	M_PLACE:	SOL	Adamson	,	Alexander	,	in	Kilbarchan	,	and	Mary	Aitken	p	.	12	Feb	1763	EOL
PAGETEXT	508	5	7	0	CLASS	SOL	Mary	,	born	16	Oct	1763	.	EOL								
PAGETEXT	517	5	8	0	CHILD:	SOL	David	,	born	1	May	1765	.	EOL								
PAGETEXT	526	5	9	0	SPOUSE:	SOL	Aird	,	William	,	and	Margaret	Aitken	,	in	Auchincloi	EOL					
PAGETEXT	538	5	10	0	CHILD:	SOL	Margaret	,	9	Feb	1707	.	EOL									
PAGETEXT	546	5	11	0	HEAD:	SOL	Aitken	(	Akin	)	,	and	Elspa	Orr	m	.	18	Dec	1693	EOL		
PAGETEXT	561	5	12	0	CLASS	SOL	Aitken	,	Allan	,	and	Mary	Aitken	EOL								
PAGETEXT	570	5	13	0	C_DATE:	SOL	Agnes	,	10	May	1741	.	EOL									
PAGETEXT	578	5	14	0	CLASS	SOL	Aiken	,	David	,	and	Janet	Stevenson	m	.	29	Sept	1691	EOL			
PAGETEXT	592	5	15	0	CLASS	SOL	Aitkine	,	Thomas	,	and	Geills	Ore	m	.	21	Dec	1661	EOL			
PAGETEXT	606	5	16	0	HEAD:	SOL	W.	Richard	Allasone	and	Ninian	Aitkine	.	EOL								
PAGETEXT	615	5	17	0	M_DATE:	SOL	Aikine	,	James	,	and	Jean	Allason	,	in	Ramferlie	,	1696	in	Kaimhill	EOL	
PAGETEXT	631	5	18	0	CLASS	SOL	m	.	23	Jan	1679	EOL										
PAGETEXT	638	5	19	0	CHILD:	SOL	John	,	28	Nov	1679	.	EOL									

Fig. 3 Edit sheet**.** Pop-up boxes (not shown) let the user pick the class, query and extract for new templates without leaving this sheet. The red box indicates a deleted template. Start/end clicks turn the start or end of the insert blue, and the start or end of the extract pink.

Selections	Edit ClassCell	Class	Search Phrase	Extract
2	F5	B_DATE:	b	1817
3	F6	PARENT2:	and	Jane Clark
4	F7	CHILD:	SOL 1	Thomas Ely
5	F11	HEAD:	SOL 243327	Rev Ben
0	F15	PARENT1:	dau	Porter Moore
7	F19	D_DATE:	d	1877
8	F14	SPOUSE:	m . 2nd	Abbie Amelia
0	F14	BU_DATE:	m . 2nd	1873
10	F26	SPOUSE:	m	Beale Steenberger
11	F12	SPOUSE:	m	Elizabeth Eudora
0	F12	M_DATE:	m	1848
13	F14	M_DATE:	m . 2nd	1873
14	F31	HEAD:	243322 . Zebulon	Zebulon DeForest
15	F48	SPOUSE:	m	Catharine Hayes
0	F50	M_DATE:	m	Apr 13 , 1817
17	F50	SPOUSE:	m . Apr 13 , 1817	Harriet

Fig. 4.Selection sheet**.** The second column points to the location of the template on the Edit sheet. The next columns list the class, query and extract for each template. Clicking on any cell in a row disables that template by changing its ID to 0 and painting it red.

A dozen templates constructed from a page or two usually suffice to let GreenEx classify over 70% of the tokens. It is, however, nearly impossible for the user to guess where to find useful but less frequent token strings. This is where GreenEx really earns its keep. When the user clicks on Propose New FTSs (Frequent Tag Sequences), the program searches the text for the most frequent tag sequences surrounding one or more user-specified KEY words. The new FTSs are appended to the Edit Sheet

(sorted by frequency and surrounded by a dashboard-specified number of "context" tokens). The top N candidates (N is also a dashboard choice) are displayed in the same format as page text, ready for the construction of additional templates. For example, given the KEY "&", GreenBook proposes: …PARENT2: married farmer EOL SOL sp & informant ANNA MINNICH….The user may then build a SPOUSE template with query sp & informant and extract ANNA MINNICH. Here only the assigned class is shown for already classified tokens (like PARENT2).

At the first session, the user is asked to enter a given name. Henceforth the program keeps track of the duration and number of sessions and interactions. The user may quit at any time and resume later by re-opening the same ClickEx.xlsm workbook.

3.4 Template matching with GreenEx

The successive transformations from a raw text line to the Family Records output require the following steps:

1. Tokenize the entire book text and assign a sequence number to each token.
2. Tag each token to reveal structural similarity.
3. Expand each template extract with the appropriate class-specific data frames.
4. Locate every tag sequence in the book text that matches any template's query and extract.
5. Arbitrate overlapping matches according to match length and sequence number.
6. Assign a class label to each matched token according to the class of the matching extract
7. Group same class tokens (like multi-token person or place names or dates).
8. Assemble class groups into family records behind HEAD tokens.
9. Optionally, create relations tuples, like (CHILD–B_DATE: David→1 May 1765), from family records.

The effect of each step (except #9) on a single line of text is illustrated in Figure 5.

(a) page 4 line 9 first SeqNo 83
(b) Adam, James, in Kilbarchan, and Jane Lyle p. 2 Aug 1746
(c) SOL Adam , James , in Kilbarchan , and Jane Lyle p . 2 Aug 1746 EOL
(d) SOL CAP , CAP , in CAP , and CAP CAP m . NUM MONTH YEAR EOL
(e) SOL HEAD HEAD HEAD NONE NONE NONE NONE NONE SPOUSE SPOUSE NONE NONE M_DATE M_DATE M_DATE EOL
(f) SOL T1 T1 T1 none none none none none T4 T4 none none T5 T5 T5 EOL
(g) HEAD:,4,10,2,3,Adam,",",James,SPOUSE:,4,10,10,2,Jane,Lyle,M_DATE:,4,10,14,3,2,Aug,1746, …

Fig. 5 (a) Book coordinates of first token; (b) Text line; (c) Tokenized text line; (d) Tagged text line, with *m* (married) substituted for alias *p* (proclaimed); (e) Labeled text line; (f) Template IDs; (g) Beginning of Family Record with offsets and lengths of class-groups. Children on lines below are not shown.

Fig. 6 lists the tags. We use about a dozen generic tags, like NUMeric, CAPitalized, ACAP (all upper case), YEAR, MONTH, PROG(eny) for son, sons, dau, daughter, and NUM6 (larger numbers used as an index in EBook), plus LITs (literals). Most LITs are token strings that are used as queries: b for born, d for died, m for married in KBook or for mother in MBook, BD for burial date. LITs can be declared by the user or designated programmatically based on their high frequency.

ALIASes are synonyms such as (born→b), (died→d), (married→m), in Ebook and (Cem, cem, cemetery→Cemetery) in MBook. A token aliased by the user to a LIT is tagged as that LIT.

[**CAP ACAP YEAR MONTH NUM NUM6 PNUM PREP PUNCT yyy SOL EOL] + LITs**

Fig. 6 List of Tags**.** PROG(eny), MONTH, PUNCT(uation), and PREP(ositions) are specified by lists. The default tag **yyy** is assigned to any lower-case token that is not a LIT or aliased to an upper-case token.

The user need not know the tagging scheme. Although templates appear to the user as text, the program tags each query and extract. The natural language tokens are used only for user displays. The template matching is based entirely on the class-specific query and extract tag sequences.

GreenEx compiles a class list from the matched templates. Fig. 7 list in alphabetic order the classes selected by a user for KBook. Only the HEAD class, which serves to group family records, is mandatory.

B_DATE:, CHILD:, C_DATE:, HEAD:, M_DATE:, M_PLACE:, SPOUSE:, TWIN1:, TWIN2:

Fig. 7 Class Names for KBook. Short names keep the class selection box that pops up on the Edit Worksheet in ClickEx to a manageable size.

A particular token may be classified by several templates and data frames, associated with the same or different classes and queries. The final class assignment to a token gives first priority to the longest query, and second priority to the longest matching extract of the longest query. Extracts can bridge text lines and even page breaks. Tokens that remain unclassified are assigned the NONE label. EOLs and SOLs retain their designations.

A useful diagnostic output of GreenEx is the Check File. For each text line of the book, the Check File reports the tokens, tags, matching labels and template_IDs corresponding to lines (a), (c), (d), (e) and (f) of Fig. 5. This unicode file is typically over 1 MB, but can be easily searched with any editor.

3.5 AutoQuery and AutoExtract

After template matching, the user may invoke two routines to augment the number of classified tokens. AutoQuery creates *augmented tag sequences* by surrounding each classified tag and each NONE tag by a specified number of *pre-tags* and *post-tags.* We used [-2,6] for all experiments, which adds two tags before and six tags after the selected tag. These values were chosen by trials on two validation pages.

AutoQuery ranks all the data frames according to their match frequency with the augmented tag sequences. High-frequency data frames that match tokens unambiguously (i.e., if only frames belonging to the same class match the target sequence) become candidate classifiers for the NONE tags with priority based on match frequency and frame length. The pre- and post-tags therefore serve as virtual queries. In the Check File, Auto-Query class assignments are listed with the source query Template_ID prefixed by "Auto_".

As an example, in a KBook line "SOL Archibald , bom 19 May 1758 . EOL", 1758 was not classified by the original query because of the OCR error "bom" for "born". AutoQuery classified it correctly because YEAR classified as B-DATE was often found with pre-tags NUM, MONTH and post-tag EOL SOL CAP , b NUM (with the last five post-tags in the next line).

As a further precaution against wrong class assignments, after the first pass with only user templates AutoQuery makes a list of Person Names and Class Names. It does not assign a Person or Place class to a

token unless the name had at least θ_N occurrences assigned by user-templates. This is consonant with our policy of tolerating several rejects to avoid a single classification error.

AutoExtract is based on a similar idea, except that instead of matching frame sequences in context, it matches individual tags surrounded by contextual tags.

3.6 Grouping and Family Records

When every token has been assigned a Class or NONE label and the user calls for generating output, GreenEx collects contiguous tokens with the same Class label into *Class Groups*. The sequence (1 `May` `1765`), with all three tokens labeled B_DATE, would become a Class Group. Class groups, like extracts, can bridge text lines and pages.

In the final step, each class group is assigned to the *Family* of the immediately preceding HEAD. In this work, family is the highest level of agglomeration. The family assignments complete the information extraction stage. The Family Record, Check, Report, and Evaluation files, all in comma-separated (`.csv` or `.txt`) format, can now be written out. Fig 5 (g) shows part of a Family Record.

As already mentioned, GreenEx can also produce a list of relations, similar to Resource Definition Framework (RDF) tuples, for populating a databased or an ontology. For example, a Parent-Child tuple would consist of the names of a Parent and a Child. The associations are based on the expected order of the items. Birth and death dates and places usually follow the name of the subject. Parent-Child orders vary: in KBook the parents are listed first, whereas in MBook the HEAD's parents' names follow the name of the HEAD, and the HEAD's children are listed after he HEAD's parents.

The tuple files are much larger than the family record files because each class-group appears only once in the family record, but it can appear several times in the tuples file. A CHILD group, for example, could belong to PARENT1-CHILD, PARENT2 -CHILD, CHILD-B_DAY, and CHILD-B_PLACE tuples.

The Report File is simply the redirected GreenEx screen print (with the level of detail under user control). It echoes all file names, input parameters, and the many statistics accumulated by the program. This file also serves to preserve the *provenance* of the experimental results.

3.7 Data Structures and Coding Issues

Our platform is a 2.4 GHz Dell Optiplex 7010 with 8GB RAM running Python 3.6 under Windows 7.0. Pervasive use of python's prodigious *dictionary* structure accounts for GreenEx's speed. Dictionaries translate sequence numbers to page-book-offset coordinates, and tag and class strings to integer identifiers. Dictionaries also keep track of class, tag, template and frame frequencies, data-frame to template associations, and provide quick access to classified tokens with matching frames as well as to unclassified tokens with frames that match only the surrounding context tokens.

The ClickEx procedure includes two dozen VBA modules. The most complex are those that map Start and End Query and Extract cells clicked on the Edit worksheet to internal template formats.

GreenEx.py and ClickEx.xlsm each contain about 2000 lines of code. The python 3.6 and Excel-VBA code, text input data, ground truth, and voluminous output files are freely available on the TANGO website [30] for replication and improvement by other researchers. The three books are in the public domain.

4. EXPERIMENTAL RESULTS

We chose books that were part of pilot experiments on ontology construction conducted by the Brigham Young University team of a long-term collaborator [15]. The sizes of the three processed book text files and of the data sets used for training, evaluation and test, are reported in §4.1. The contributions to Precision and Recall of the user templates, data frames, AutoQuery and AutoExtract are tabulated in §4.2 for the dozen pages for which we have ground truth. In §4.3 we provide what comparison we can with results achieved by others.

4.1 Data

Table 1 shows the sizes of our data sets in terms of pages and tokens. The proportion of training and text tokens labeled in the ground truth (i.e., "useful" tokens) is about 52% in KBook, 41% in MBook, but only 21% in EBook. The fraction of useful tokens is one measure of *structure*. According to this measure, EBook is least structured. It contains, interspersed among crisp lists of factoids, narrative paragraphs about the protagonists, distinguished civilian and military careers, and potential sources of additional information. Because these items are only of marginal use in genealogy, they are not extracted.

Table 1 Data Sets, Ground Truth, and assigned class labels

Book	Data Set	Pages	Tokens	GT Class labels	GreenEx Class labels
EBook	training	575	1663	456	
	validation	576,577			
	test	610-613	2384	645	
	all	112-700	279065		58054
KBook	training	5	1625	695	
	validation	4,6			
	test	50-53	2401	1032	
	all	4-127	60731		31532
MBook	training	7	2056	809	
	validation	8,9			
	test	90-93	2704	1106	
	all	7-395	223830		98147

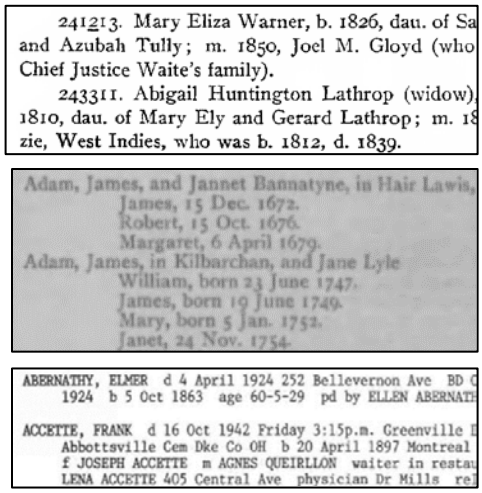


Fig. 8 Snippets from the three books from the searchable OCR PDF output.

Fig. 8 shows portions from the PDF file of each book. The unicode txt files (e.g. Fig. 5 b) that we used lack most of the formatting (indentations, type sizes and styles) of the original OCR output. Using such information would require book-specific layout rules.

Although the print quality of the three books is fair, the OCR accuracy is below what can be expected from current products. Many of the errors are standard: `rn` for `m`, `bom` for `born`, `Gem` for `Cem`, `Lavvis` for `Lawis`, `j` for `J`, `I` or `|` for `1`, and period/comma confusions. There are, however, some anomalies, including 223 instances of `i860` instead of `1860 in` EBook. These cannot be explained by the shape of the "1", because there are less than a dozen OCR errors among the thousands of other dates. We have not yet been able to trace when and where these books were OCR'd.

4.2 Evaluation

Table 2 supports the following observations. For these three books, the average fraction of GreenEx labeled tokens is about 40% (194924 / 563626). This corresponds roughly to the proportion of useful tokens estimated from the GT. So most of the factoids were labeled by GreenBook. User-built templates are directly responsible for about three-quarters of these labels (164143 / 194924). Most of the remaining labels are contributed by the data frames. AutoQuery and AutoExtract labeled about 4%. Class groups contain, on average, two tokens. There are almost 10 class groups per family. The longest family records appear in Ely, which reports the family head's and spouse's parents and sometimes the children's spouses and *their* parents as well. The runtime is roughly proportional to the product of the number of templates and the number of book tokens. Table 3 shows the class-by-class classification results for MBook.

Table 2 Templates, Data frames, AutoQuery and AutoExtract performance

Book	Processed tokens	Templates	Labeled by user templates	Labeled by user templates + data frames	Labeled by Auto Query	Labeled by Auto Extract	Total labeled tokens	Class Groups	Family Records	RunTime seconds
EBook	279065	18	45228	58054	927	962	59943	34176	1763	22.17
KB00k	60731	19	31070	31532	358	196	32086	14249	2677	10.22
Mbook	223830	23	87845	98147	796	3952	102895	38788	4782	35.41
Total	563626	60	164143	187733	2081	5110	194924	87213	9222	67.8

How accurate is GreenBook? GreenEx classifies every token in each book from start-page to end-page. Comparison of the output labels with a Ground Truth File (GT) generates an ERROR File (Fig. 9) for computing performance statistics. "NONE" in the GreenEx output is the *default class* for unlabeled tokens in the GT (marked as "?"). Each ERROR file includes a summary table with class-by-class counts of the possible outcomes. The experiments below generated 15 ERROR files.

When the wrong class is assigned to a GT-labeled token, or any class other than NONE is assigned to a token unlabeled in the GT, it is counted as an *error* or *false positive*. NONE assigned to a token with a class label given in the GT is a *reject* or *false negative*. When the correct class is assigned, or NONE is assigned to an unlabeled token, it is deemed *correct* or *true positive*. (We also evaluated accuracy using only class-labeled tokens in the GT. There was little difference because almost all unlabeled tokens were rejected. The "error" and "reject" categories are useful for characterizing multi-category performance.) For this application, we set parameters to trade off an error for several rejects.

Table 4 shows the Precision and Recall on the training and test sets, with and without data frames, AutoQuery and AutoExtract. The error counts are obtained from the corresponding ERROR files. The F-measure on the test sets with all the modules ranges from $F= 0.96_6$ to $F=0.98_8$. The subscripts indicate unreliable estimates due to sample size. Missed HEADs may have significant consequences: the spouse or child of a missed HEAD could be assigned to the family of the previous HEAD! The only missing Head in the test sets was a family name without any given name that was labeled as CHILD.

We were unable to conduct human-factors experiments that would need far more ground truth, and subjects with diverse backgrounds. We found that time-consuming correction of our mistakes decreased

rapidly with experience. After 3-4 hours of practice, template construction by the author plateaued at about one minute per template. We processed all three books in 220 minutes.

Like everybody else, we blame most of our program's mistakes on flaws in the input data. Many misclassifications are actually due to OCR errors. The OCR segments 1753 as `175 3`, 1769 as `1 769`, and Margaret as `M ' argaret`. In KBook, OCR miss-segmentation accounted for 9 of the 14 rejects and 1 error. In Ely, the first child is often rejected because of a misrecognized numeral 1. Since most of these OCR defects would not stump even a poor human reader, we should be able to keep them from confusing GreenBook otherwise than by building a template for each defect or waiting for the perfect OCR. We believe that we could eliminate about a quarter of the current non-OCR rejects by optimizing various user-settable parameters separately for each book. We could also improve tagging to avoid, for example, children named April, May, June or August from being tagged as MONTH.

page 4 line 29 first SeqNo 274								
SOL	Adam	,	John	,	and	Jean	Reid	EOL
SOL	HEAD:	HEAD:	HEAD:	NONE	NONE	SPOUSE:	SPOUSE:	EOL
SOL	HEAD:	HEAD:	HEAD:	?	?	SPOUSE:	SPOUSE:	EOL
---	CC	CC	CC	---	---	CC	CC	---
page 4 line 30 first SeqNo 283								
SOL	John	,	14	Nov	1673	.	EOL	
SOL	CHILD:	NONE	C_DATE:	C_DATE:	C_DATE:	NONE	EOL	
SOL	CHILD:	?	C_DATE:	C_DATE:	C_DATE:	?	EOL	
---	CC	---	CC	CC	CC	---	---	

Fig. 9 Excerpt from KBook ERROR file. The second line is the class assigned by GreenEx, the third is the ground truth (? means that the token is not a potential extract), and the fourth is the result of the comparison: CC (correct), EE (error), RR (reject), ---.(correct, default class).

Table 3 MBook Classification

	Groups	Tokens		
Class		Template-labeled	Auto Inseert	Auto Extract
BU_DATE:	3651	9809	261	604
BU_PLACE	4106	9500	7	51
B_DATE:	2922	8677	9	26
B_PLACE:	3606	10542	6	76
CHILD:	10148	15651	1429	3418
D_DATE:	4302	12644	56	34
D_PLACE:	4787	11195	7	1115
HEAD:	4261	15638	17	49
PARENT1:	2934	6290	22	60
PARENT2:	2656	5691	12	47
SPOUSE:	2124	3956	129	232
Total	45497	109593	1955	5712

Table 4 Precision and Recall

Data Set	N	Process	Precision	Recall	F
EBook					
Train	1663		0.998	0.984	0.991
Test	2384	User Templates	0.999	0.841	0.913
		Plus DataFrames	0.999	0.956	0.977
		Plus AutoQuery	0.997	0.965	0.981
		Plus AutoExtract	0.996	0.971	0.983
KBook	1625				
Train	2401		0.999	0.989	0.994
Test		User Templates	1.000	0.951	0.975
		Plus DataFrames	0.999	0.975	0.987
		Plus AutoQuery	0.998	0.978	0.988
		Plus AutoExtract	0.997	0.982	0.989
MBook					
Train	2056		0.982	0.951	0.966
Test	2704	User Templates	0.993	0.938	0.933
		Plus DataFrames	0.993	0.938	0.965
		Plus AutoQuery	0.992	0.938	0.964
		Plus AutoExtract	0.985	0.947	0.966

4.3 Comparison with Other Work

The only publications that we found about information extraction from entire family books are from BYU and FamilySearch. They report an experiment with OntoES/FROntIER which extracted information on 8539 individuals from KBook with 25 hand-coded regular expressions [15]. Based on a check of several randomly chosen pages the F-score "was judged to be near 95%". The same result is reported a few pages later with the complete pipeline.

Under pipeline runs in [15] there is also a report of a complete automated run on MBook that extracted 12,226 "individuals." No F-score is given here. We are not sure exactly what was extracted and do not believe that our 22123 MBook "person" extracts (as in Table 3) are comparable. The first stage of recent experiments reported by the BYU team used GreenQQ, an early version of GreenEx without ClickEx, AutoQuery, AutoExtract or data frames, instead of their FROntIER regex approach [31]. Only final pipeline results are reported in detail.

5. RATIONALE FOR CHOICE OF METHOD

GreenBook's components are conventional: only their combination is new. We did not start with a solution in search of a problem but because even with many acolytes, cutting and pasting from family books was obstructing progress on a large project [15]. Why do we propose a dowdy methodology instead of resorting to modern machine learning techniques with a record of excellent results on character, speech and face recognition? Because in our experience with neural networks and statistical classifiers, every data set from a new source required so much joyless covert effort ("tuning") for respectable results. And even with canned feature extraction and classification software, there is no way to avoid laborious labeling of possibly inappropriate training data.

Machine-learning experiments seldom track user time. Is it considered expendable? Researchers seldom report the number of runs and adjustments (hopefully only on the training and validation sets) before their (single?) run on the test set. In contrast, GreenBook interaction is out of the closet. The user interacts openly with ClickEx. We understand that most new books will require some adjustments (we had to introduce a new tag, NUM6, for EBook's identifiers), but believe that our model-based approach promotes more predictable and understandable, less data-intensive, and quicker extension to new input than would machine learning.

Without interaction, one is stuck with the best results obtained with the available engine and training data. Active and semi-supervised learning cannot yet take full advantage of user insight. Post- correction does not improve the engine. The same errors must be corrected again and again.

Precision and recall criteria must be ultimately based on the (estimated) downstream cost of failures relative to extraction cost. With *any* available method, for the task at hand computer cost is negligible compared to human time. Human-friendly built-in interaction based on transparent template-matching output provides a significant multiplier of human annotation with a guarantees that any performance criterion can be reached. That is why we believe that GreenBook is a step in the right direction.

6. CONCLUSION

Over 360,000 digitized and OCR'd family books have been collected by FamilySearch alone [17]. Applications of GreenBook other than to historical documents are unlikely. Why would anyone today record important facts anywhere but in a Cloud? We do, however, want to extract information from some other semi-structured documents before they all get recycled. We have already began collecting OCR'd city directories and sales catalogs (dictionaries and gazetteers are too easy: they have all been already converted to "digital resources"). We also have some interest in automating the harvesting of semi-structured documents from the web, and are therefore experimenting with some quantitative measures of document structure. We are relieved to see that Moby Dick, Tale of Two Cities and Brave New World score lower on these measures than our family books.

Another task awaiting attention is verification of our conjecture that GreenBook is essentially script and language independent – at least for Latin- and Greek-based scripts and languages. Chinese *Zupu* (or Jiapu) family records are seldom printed, but many are well within the scope of current Chinese OCR. Many families in India also kept family records that could provide an appealing application.

The major shortcomings of the current edition of GreenBook are the inflexible displacement between query and extract and vulnerability to OCR errors. We must replace our tie-rod coupling between query and extract by an elastic shock cord, and introduce some context-sensitive edit distance into tagging and matching to allow for misrecognized characters.

Although GreenBook can help a user to extract *all* relevant information in a book, it is a long-tailed process. While on these books 99% precision and 95% recall are achievable in a few hours, 99.8% precision and 98% recall could take the same operator several days. It is even possible that keyboarding an entire book would be faster than extracting everything using GreenBook.

We are happy to be able to report low error rates for sample pages from the three books, and are not unduly concerned about missing some birth and marriage places. For one thing, many of these are missing in all three books. For another, genealogical information extraction has something in common with picking blueberries: it is not essential to collect every berry from each bush. For the record, the weighted average Precision/Recall F-measure for the three books (Table 4), a target for future research, is 0.98_3.

ACKNOWLEGMENT

My friendship and collaboration with Professor (now Emeritus) David W. Embley of Brigham Young University dates back so many decades that I can no longer tell which ideas were his and which were mine. This work would not have been possible without his and his team's essential contributions.

Conflict of Interest: The author declared that he has no conflict of interest

REFERENCES

[1] 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.

[2] Miller Funeral Home Records, 1917 – 1950, Greenville, Ohio 1990)

[3] The Ely Ancestry (Vanderpoel 1902),

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

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

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

[7] 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.

[8] 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.

[9] 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.

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

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

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

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

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

[15] D. W. Embley, S. W. Liddle, D. W. Lonsdale, S. N. Woodfield, Ontological Document Reading, An Experience Report, International Journal of Conceptual Modeling, pp. 133-181,February 2018.

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

[17] S.N. Woodfield, S. Seeger, S. Litster, S.W. Liddle1, B. Grace, and D.W. Embley; Ontological Deep Data Cleaning: 37th International Conference, ER 2018, Xi'an, China, Proceedings. 10.1007/978-3-030-00847-5_9.

[18] 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).

[19] 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.

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

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

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

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

[24] 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.

[250 G. Nagy, Disruptive developments in document recognition, Pattern Recognition Letters (2015 http://dx.doi.org/10.1016/j.patrec.2015.11.024) December 2015.

[26] 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.

[27] D.W. Embley and G. Nagy, Green Interaction for Extracting Family Information from OCR'd Books, Document Analysis Systems Workshop (DAS'18), Vienna, April 2018.

[28] D.W. Embley and G. Nagy, Extraction Rule Creation by Text Snippet Examples, Family History Technology Workshop, Provo, UT, February 2018.

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

[30] Table Analysis for Generating Ontologies: https://tango.byu.edu/

[31] D. W. Embley, S. W. Liddle, D. W. Lonsdale, and S. N. Woodfield, Inter-Generational Family Reconstitution with Enriched Ontologies, ER 2019, First International Workshop on Conceptual Modeling for Digital Humanities, Salvador, Bahia, Brazil, Nov. 4-7, 2019.