Name, Date, and Place Recognition in Unstructured Text

David G. Wiggins III
Scott N. Woodfield

Computer Science Dept.
Brigham Young University

Abstract

Manually searching a text document for important genealogical information is slow, tedious, and error prone. It wastes time, contributes to the widely held perception that family history is boring, and commonly misses important information in the document. We propose a tool, called the *NameDatePlace Extractor* that can automatically scan machine-readable text and extract name, date, and place information. The tool should do it quickly, easily, and in some cases more accurately than traditional approaches.

Introduction

Family history research consists primarily of searching for source documents and extracting relevant information. Before the advent of the computer and the Internet, searching was time consuming and tedious. Family history was considered boring. We are now entering an age where searching for documents is becoming easier. Many documents are being digitized and made available on the Internet. Through OCR, extraction, and indexing we can find pertinent source documents in a fraction of the time it used to take and we can do it in the comfort of our own home.

Much of the current extraction work deals with sources such as census records, birth certificates, death certificates and other vital records. These sources are usually small and semi-structured. For instance, a birth certificate is never more than a page's worth of information and is organized in such a way to make it easier to find important dates, places and people. Other sources are large but can be broken down into small units of semi-structured information. For instance, census records are usually long and can be portioned into lines where each line has the same structure. It is reasonable for a person to read an entire line on a census record or a birth certificate in a few minutes and convert all pertinent information into a machine-readable format. We can then use indexing technology to provide powerful search capabilities that take only seconds to return results. As a benefit, we can inspect the results quickly and determine if the document contains anything of interest. If so, we can electronically or manually copy all relevant information into our source repository.

There are other sources that do not lend themselves as easily to human extraction. This is especially true for long, unstructured text documents. Some examples are, news articles, journals, family history books, biographies, diaries, letters, and blogs. With decreasing storage and electronic communication costs, more of these unstructured text documents are being made available to people on the Internet. Some were and are created as machine-readable documents. Many others are being transcribed or OCRed. In some of the aforementioned cases the information is indexed to make searching quicker and easier. But, there is a problem.

Problem

Even though we may be able to quickly find unstructured sources, it still takes time to extract relevant information. For instance, assume we found a fifty-page history on the Hadley family wherein a Jacob Marley is mentioned. Inspecting the document for collateral information requires that we read the entire document. It could take up to an hour. We may read many paragraphs containing no direct information on dates or names. It is also possible, that as we read such a lengthy document our mind may wander and we may miss important information.

Skimming isn't a complete solution. It does cut down on reading time but can significantly increase the chance we will miss important information.

Indexing software like Google doesn't really help. It can only show us where the key words were found in the document. It cannot show us where other related and important information is located. If we searched for documents with the name "Jacob Marley" the indexing software can show us the documents containing "Jacob Marley" and where that name appears in the document. It cannot show us information about important events in Jacob Marley's life or anything about his relatives.

Books reduce this problem by providing indexes. In the index we find a list of important concepts and where we can find those concepts in the book. The important questions in family history documents are who, when, and where. These concepts manifest themselves as names, dates, and places. There are some family history books with indexes that contain names, dates, and places along with the pages on which these items appear. Our intent is to create a tool, called the *NameDatePlace Extractor* that will dynamically create an index for any text document.

The first version of *NameDatePlace Extractor* will automatically create a book-like index that shows where names, dates, and places appear in a machine-readable text document. In the next section of this paper we will describe the design of the software. We will then describe the various uses and benefits of the tool. Last we describe how we will enhance the software to extract additional information such as events and relations.

Implementation

The algorithm for date, name, and place extraction consists of four steps.

1. Convert the content to plain text.
2. Convert the text from a sequence of characters to a sequence of tokens.
3. Identify the dates, names, and places.
4. Format the results.

In step one we convert any text document in any format into plain text. For instance, the source could be in html or word format and we will produce a plain text document. For the prototype we will implement the extraction from html files. We will also design the code so that it is easy to plug in any other format converter.

Step two performs lexical analysis. It converts a sequence of characters into a sequence of tokens with appropriate token types. It does this in two phases. First it uses a finite state machine to separate the sequence of characters into a sequence of tokens. This is done quickly. In the second phase we categorize every token as to whether it is a title, given name, surname, name suffix, city, county, parish, state, country, or none-of-the-above. We do this by looking up every word in a name authority and place authority in order to classify the word. The lookup algorithm is based on a multi-hash lookup technique used in spell checkers. Because it is hash based its run-time speed is O(n) where *n* is the number of words in the document. It should be noted that a token may be in more than one class . For instance, the word Indiana may be classified as a given name or a state.

Version 1 of *NameDatePlace Extractor* will assume a United States ontology for places. Subsequent versions will be implemented with a more global ontology.

Step two could have been combined with step three but would have required the use of context-free parsing techniques to tokenize the input characters. This would have been slower. So, like compilers, we do lexical analysis first (step 2) and then parse the tokens (step 3).

Step three uses an LL(1) grammar to parse the tokens and extract complete names, dates, and places. The first version of the name grammar will be similar to the following:

Name	→ *title* **SimpleName** | **SimpleName**
SimpleName	→ **NameWithGivenNames** | **LastName**
NameWithGivenNames	→ **GivenNames LastNameAndSuffix**
LastNameAndSuffix	→ **LastName GenerationId** | ε
GivenNames	→ *givenName* **OptionalPeriod RemainingGivenNames**
RemainingGivenNames	→ **GivenNames** | ε
LastName	→ *surname* **RemainingSurnames**
RemainingSurnames	→ *dash surname* | **SurnameList**
SurnameList	→ *surname* **SurnameList** | ε
GenerationId	→ *suffix* | ε
OptionalPeriod	→ *period* | ε

The previous grammar will be used as an initial definition of names. For version two of *NameDatePlace Extractor* we will analyze text documents to determine how to refine the grammar.

To identify dates we use the work of Robert W. Lyon[Lyon2000]. His implemented pattern matching software identifies dates found in text documents. Using his software we are able to quickly identify dates and their subparts.

Extracting places is similar to extracting names. We assume the lexical analyzer has identified possible cities, counties, states and countries. We will then use the following grammar to identify places:

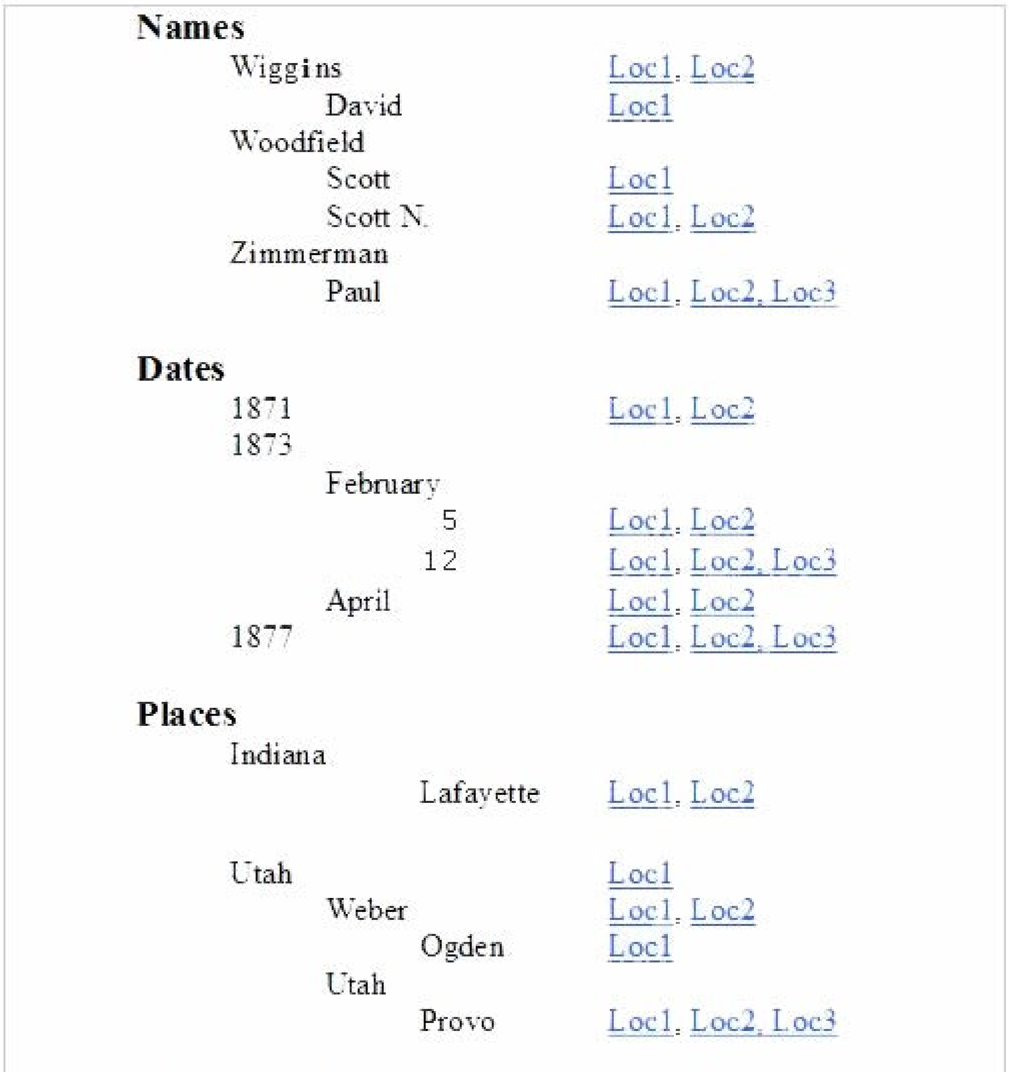
Place	→ *city* **PlacesFollowingCity** |
	CountyStateCountry |
	StateCountry |
	country
PlacesFollowingCity	→ **PossibleComma CountyOrStateOrCountry** | ε
CountyOrStateOrCountry	→ **CountyStateCountry** | **StateCountry** | *country*
CountyStateCountry	→ *county* **CountyDesignator PlacesFollowingCounty** |
	StateCountry |
	Country
CountyDesignator	→ *Left_Paren* **CountyOrParish** *Right_Paren* | **CountryOrParish**
CountyOrParish	→ *countyWord* | *parishWord*
PlacesFollowingCounty	→ **PossibleComma StateOrCountry** | ε
StateOrCountry	→ **StateCountry** | *country*
StateCountry	→ *state* **StateDesignator PlacesFollowingState**
StateDesignator	→ *Left_Paren stateWord Right_Paren* | *stateWord*
PlacesFollowingState	→ **PossibleComma** country
PossibleComma	→ *Comma* | ε

As with names, version two of *NameDatePlace Extractor* will revise the previous grammar to account for further analysis of how places appear in text documents. Subsequent versions of our tool will also handle non-US place ontologies.

Following the name, date, and place identification stage we format and display the results. There are numerous formatting options that could be implemented. For the first version of the *NameDatePlace Extractor* we will implement a plugin for Firefox. It allows the user to press a button and the name, date, and place information will be extracted from the frame of focus. The results will be displayed in another window. They will be formatted to appear like an index but with links to locations in the original document. The user can press any link and will be taken to the appropriate place in the document. Figure 1 shows a possible example of the output.

The Benefits

The *NameDatePlace Extractor* solves a number of the problems that we face in genealogy. First, people generally do not want to read all the text that is given to them. If they do read it, they tend to skip over the text and look for the important parts. Though the human mind is an amazing tool, we will make mistakes and are prone to missing vital things. The new tool should be as accurate if not more accurate than a human being. Precision should be close to 100%. That is, almost everything returned should be a name, date, or place. Recall, which is the percentage of the total names, dates, and places in the document that should have been returned, is another measurement of efficiency. Recall should also be as high, and usually higher, than manual attempts; especially for large documents. Together, recall and precision make the *NameDatePlace Extractor* beneficial. For instance, if a document is a single, densely typed page, it might take two or three minutes to read and process. The *NameDatePlace Extractor* should only take a few seconds. For a 50 page manuscript it will take the average reader well over an hour to evaluate the information. Our tool should take only a few minutes.



```
Names
        Wiggins                         Loc1, Loc2
                David                   Loc1
        Woodfield
                Scott                   Loc1
                Scott N.                Loc1, Loc2
        Zimmerman
                Paul                    Loc1, Loc2, Loc3

Dates
        1871                            Loc1, Loc2
        1873
                February
                        5               Loc1, Loc2
                        12              Loc1, Loc2, Loc3
                April                   Loc1, Loc2
        1877                            Loc1, Loc2, Loc3

Places
        Indiana
                        Lafayette       Loc1, Loc2

        Utah                            Loc1
                Weber                   Loc1, Loc2
                        Ogden           Loc1
                Utah
                        Provo           Loc1, Loc2, Loc3
```

Figure 1 – Sample Output from the *NameDatePlace Extractor*

One of the key benefits of this tool should be to help determine whether a document is even worth reading. After indexing a document the number of names, dates, and places that are of interest should indicate if we want to continue searching to extract inter-personal relationship information (e.g. mother, father, son, daughter) and event information (e.g. birth, death, marriage). Once it is developed, we can use it to do statistical analysis of common genealogical documents to create an automated ranking system.

It is also possible to store the results from using the *NameDatePlace Extractor* and use the results for other purposes. In doing so, the tool helps by providing the power to convert unstructured text to a structured format for future use. For instance, Google does not normalize dates well. A search for "12/10/1951" won't return the same results as "December 10, 1951". It is possible, using the stored results of our tool to build a specialized index of genealogical information that is more specific than general indexing tools like Google.

Conclusion/Future work

The *NameDatePlace Extractor* is still in its design phase. Over the next few months we plan to further specify how version one will be implemented and provide a mock up to graphically demonstrate our expectations. We expect that in the Fall semester of 2009 a group of CS students from BYU will assemble for a capstone project to implement version one. Recruitment for this group is currently underway.

After finishing version one of the *NameDatePlace Extractor* we plan on extending it in several ways.

- Refining Name and Place Grammars and Authorities
  - The name and place grammars used in version 1 can be improved. We will use a random sample of genealogical text documents to evaluate how well our grammars work. We will then modify and extend our present grammars to improve the precision and recall of our name and date extractors. Similar techniques will be used to determine how to improve our name and place authorities.

- Using Ontologies
  - Once a name, date, or place is found, we can use ontologies [Embley1998] to analyze the rest of the sentence. For example, if a name, John Smith, is found in a sentence, we might scan the rest of that sentence to check for event keywords such as “born”. If the word “born” is present we can determine if the sentence structure matches expected ontological patterns for birth events. If so, we can use the pattern to identify and extract birth information about John Smith. Similar techniques can be used to extract relational information from sentences containing keywords such as “daughter”, “parent of”, or “mother”.

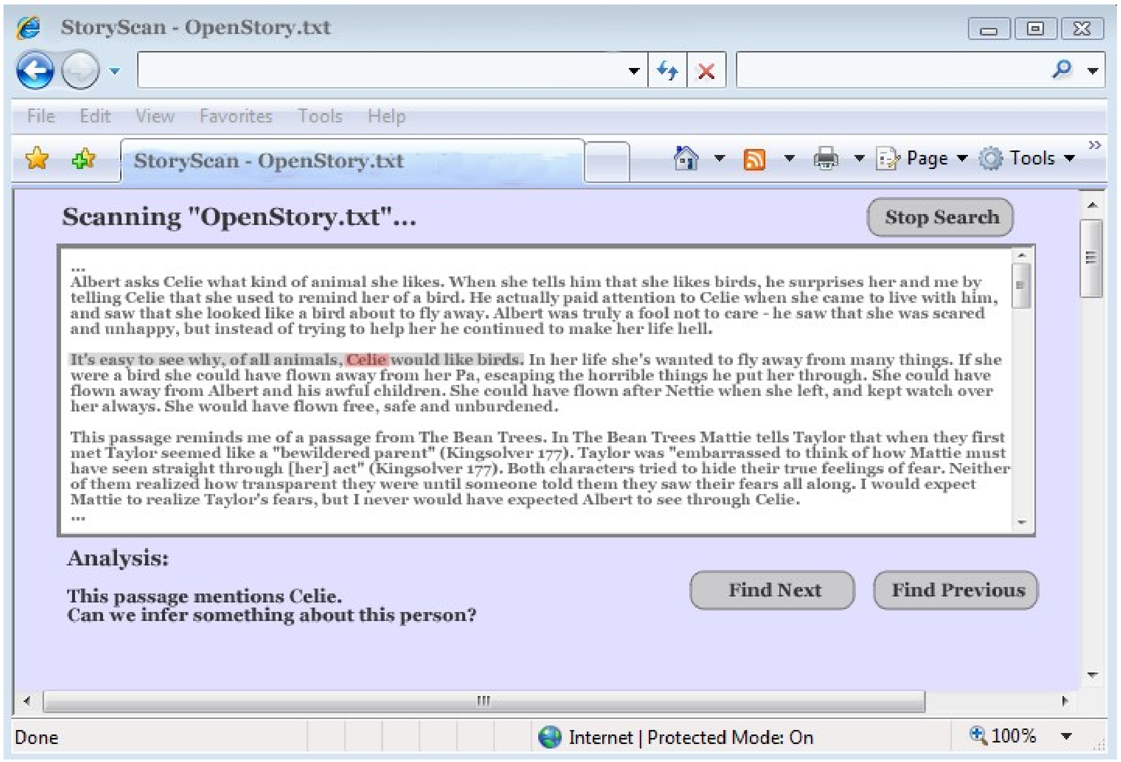
- Rating of Documents
  - Just because a document contains a name, date, or place does not mean the document is useful. Once a document has been identified as containing information of interest, we intend to use basic name, date, and place data, together with event and relational information to calculate its usefulness score. The score will provide a researcher with a metric that helps the user determine if they wish to spend more time with the document. We can also use the score to rank documents to determine which may be most useful.

- Foreign Languages
  - The name and place authorities and grammars used in version one are United States centric. We intend to redesign version one to allow plugins for name and place authorities and configuration files for name and place grammars. Using these enhancements we can easily convert the software to other languages.

- Advanced Web Searching
  - Instead of building our own custom software and database for indexing online genealogical

information, we can use the technology developed for this tool to create a frontend for Google. Entries entered into the frontend could be expanded into alternative representations and used to form a more complex query for Google. For instance, if we enter a date into the frontend, the tool would automatically transform it into the myriad number of legal representations. “Dec. 10, 1951” might become “December 10, 1951”, “10 Dec 1951”, “12/10/51”, etc. The frontend would then send a disjunction of all the alternative date representations to Google. Google would then return all pages with the given date, not just the pages containing the string “Dec. 10, 1951”.

- Developing Alternative Output Formats
  - The index-like format of the output is useful but many other formats are needed. We intend to modify the *NameDatePlace Extractor* so that the formatter becomes a plugin. Thus, the software can be easily converted to create and store machine-readable formats of the result. Output that is more suitable for printing might also be created. We also think there are other interfaces for better perusal of the results. For instance, when a person clicks on a link in the resulting document, we might produce a window that looks like the following:


StoryScan - OpenStory.txt
File Edit View Favorites Tools Help
StoryScan - OpenStory.txt
Page Tools
Stop Search
Find Next
Find Previous
Done
Internet | Protected Mode: On
100%

Scanning "OpenStory.txt"...

...
Albert asks Celie what kind of animal she likes. When she tells him that she likes birds, he surprises her and me by telling Celie that she used to remind her of a bird. He actually paid attention to Celie when she came to live with him, and saw that she looked like a bird about to fly away. Albert was truly a fool not to care - he saw that she was scared and unhappy, but instead of trying to help her he continued to make her life hell.

It's easy to see why, of all animals, Celie would like birds. In her life she's wanted to fly away from many things. If she were a bird she could have flown away from her Pa, escaping the horrible things he put her through. She could have flown away from Albert and his awful children. She could have flown after Nettie when she left, and kept watch over her always. She would have flown free, safe and unburdened.

This passage reminds me of a passage from The Bean Trees. In The Bean Trees Mattie tells Taylor that when they first met Taylor seemed like a "bewildered parent" (Kingsolver 177). Taylor was "embarrassed to think of how Mattie must have seen straight through [her] act" (Kingsolver 177). Both characters tried to hide their true feelings of fear. Neither of them realized how transparent they were until someone told them they saw their fears all along. I would expect Mattie to realize Taylor's fears, but I never would have expected Albert to see through Celie.
...

Analysis:

This passage mentions Celie.
Can we infer something about this person?

Figure 2 – Alternative Result Interface

It shows the context of the name, date, or place of interest. It shows the results of a subsequent analysis of the sentence. It provides a means for the user to go to the next or previous occurrence of the item. The user can also select another name, date, or place on the screen and the tool will now use that as the item of interest.

We have many computerized tools to search for family history information. Once we locate some types of information such as a line on a census record or a birth certificate, the human-based extraction and recording of associated information is relatively easy. However, if we locate a text document, especially a large text document, human-based analysis is time consuming and error prone. The *NameDatePlace Extractor* significantly reduces errors and speeds up the process of extricating names, dates, and places.

Bibliography

[Embley1998] Embley, David W., Campbell, Douglas M., Liddle, Stephen W., Smith, Randy D., *Ontology-based extraction and structuring of information from data-rich unstructured documents*, Proceedings of the Conference on Information and Knowledge Management, Washington D.C, 1998.

[Lyon2000] Lyon, Robert W., *Identification of temporal phrases in natural language*, Masters Thesis, Brigham Young University. Dept. of Computer Science, 2000