Genealogical Indexing of Obituaries Using Automatic Processes

Patrick Schone, Jake Gehring; Family Search, Salt Lake City, Utah.

Abstract

Due to the ability of modern obituaries to provide rich genealogical information for family members who have died within the bounds of "living memory," family history organizations have recently begun to acquire and index obituaries in vast quantities. The indexing process for these documents is typically done using human labor. Yet we describe an effort by FamilySearch which leverages various kinds of machine learning, statistical analyses, and rule-based processing to automatically index such documents without human intervention at rates thousands of times faster than humans while still achieving high levels of accuracy.

1. Introduction

In recent years, genealogical organizations have noted that the older historical records they have typically acquired for family history purposes do not adequately address the needs of patrons who are seeking relatives who existed within living memory. Obituaries, however, do contribute significantly to living memory – especially those that have been created in the past 30 years. Consequently, FamilySearch, like other organizations, has sought to acquire large volumes of obituaries and create genealogical indexes for them. In fact, starting in 2014, FamilySearch began to acquire tens of millions of obituaries that were "born digital" (i.e., created originally in digital form) and announced to their indexing volunteer workforce to be beginning "The Year of the Obituaries" [1]. Many individuals agreed to help in the indexing of this data even though 15-30 minutes are required to index each obituary.

These born-digital documents are raw digital text as opposed to images. So a question comes to mind: can natural language processing technology adequately solve this problem and thus allow volunteers to devote their efforts to other key endeavors?

Entity and relation extraction have been areas of active research over the past twenty years in the research community, and these technologies have the potential of contributing significantly to obituary indexing efforts. *Entity extraction* seeks to identify key elements of a text such as names of people, dates, locations, and so forth. *Relation extraction* in this context is technology to automatically draw connections between identified entities (eg., PERSON#1 is-father-to PERSON#2).

In addition to these forms of content extraction technologies, others capabilities are required in order to solve the entire indexing problem. For examples, gender detection, coreference analysis, and name chunking all could play roles in automatic indexing. *Gender detection* seeks to use the names of individuals and other contextual clues to determine whether a person is male or female. *Coreference analysis* must track that "Robert K Jones, Jr." could also appear in the text as "RK" or "Bobby" or "Mr. Jones" or "he." *Name chunking* seeks to identify the roles of the personal name constituents (eg., Robert K is a given name phrase, Jones is a surname, ", Jr." is an ordinal phrase).

FamilySearch has created machine-learned and statistical versions of these and other related components and has combined them into a system (affectionately referred to, by some, as the "*Robokeyer*") which can produce indexes that resemble those of human indexers. We have evaluated the performance of this automatic indexing system against millions of human-indexed obituaries. Moreover, after our system accuracy attained reasonable levels, we likewise began publishing results to the world using this automation.

A machine indexer can provide massive increases of productivity over what individual humans can provide. Before the end of 2015, FamilySearch was able to "robokey" 26.5 million obituaries --which reference over 200 million individuals—and the results will come online in the beginnings of 2016. This task was performed in five days on a reasonably-sized compute cluster, whereas the same task would have taken at least 1500 career years for a person indexing obituaries as a full-time job.

Despite these huge benefits of productivity, there are also negatives to the use of automation: the Robokeyer still make errors and some of these can be quite unusual. More specifically, humans tend to make errors of omission and interpretation and errors in spelling when they index obituaries. Yet the computer tends to make errors of interpretation regarding key information; and it can also lose the "semantic thread" of textual documents which can result in laughable or even upsetting outcomes. To help mitigate some of these problems, we implemented a machine-learning-based confidence tagger which attempts to predict if the Robokeyer thinks its results will be "sweet" or "sour" with the expectation that only those documents that are deemed non-sour will be forwarded for publication. Though this kind of confidence predictor cannot hope to predict erroneous results with 100% accuracy, it is still able to appreciably increase the perceived resultant quality of the Robokeyer.

The success of this effort has led FamilySearch to ask: could Robokeying still work when obituaries are identified in actual newspaper print as opposed to born digital? Estimates are that there as many as ten times more images of obituaries than there are born-digital ones. This would require a pre-processing step of finding and transcribing obituaries on newspaper pages. Even if people were to pre-identify the image snippets containing the obituaries, commercial OCR engines for image transcription favor transcribing recent documents and are not well-suited to historical newsprint. We have therefore also worked to build an OCR engine which has been yielding results that exceed those of commercial engines on historical newspapers. It is our expectation that we will begin coupling our OCR (which we call *Athilos*) and the Robokeyer in 2016 to begin indexing obituaries from newsprint.

In this paper, we will provide a general overview of the Robokeyer and we describe in some detail its constituent technologies. We also discuss how the system is scored against human indexing, the levels of performance that it attains on different evaluation tasks, the kinds of errors that are observed, and the confidence predictor that is used to enrich the results. At the same time, we indicate the additional pieces of information that the Robokeyer is able to extract which would be beyond the levels that human indexers could afford to do. Lastly, we conclude with a brief overview of our Athilos OCR engine, its performance, its limitations, and expectations we have for coupling it with the Robokeyer.

2. Robokeyer Technology Components

When humans perform the task of indexing obituaries from born digital documents, they must do a number of steps. These include: (a) identification of person names, places, dates, and other key information; (b) drawing connections between these facts such as "date *D* is the birth date for person *P*"; (c) disambiguation of names; (d) discarding of irrelevant pieces of data; (e) making inference regarding the gender of people names; (f) inferring given versus surname bounds, as well as titular, occupational, or generational name components; (g) perform family relationship mathematics (X is Z's sister and Z is wife of Y, so Y is X's brother-in-law); and (h) identifying the principals in the story. When humans are indexing, they also have to determine (i) if the information they are looking at appears to *not* be an obituary, and (j) if they believe they are not personally equipped to handle the obituary. To emulate the human, the Robokeyer must be likewise able to do these same steps. In this section, we describe each of the technologies which we have created to follow human-like processes. In addition, the Robokeyer is able to have functionality which is beyond the scope of what human indexers might be asked to do and we describe that as well.

2.1. Identifying Entities

Entity tagging is a technology which has been in existence for decades [2]. The notion behind entity tagging is that a computer is used to detect certain classes of information in raw textual output such as the names of people, places, dates, times, organizations, and so forth.

For those not in the field of entity tagging, this technology seems like it could be trivially implemented through the use of lists and rules. Usually, though, such solutions are extremely brittle and yield weak and highly error-prone results. These errors are the result of issues such as words having multiple parts of speech ["I gave it to *Pat*" vs "*Pat* it on the head"]; use of the same verbiage for different kinds of entities [as in "George *Washington*" (PERSON), "Seattle *Washington*" (LOCATION), "University of *Washington*" (ORGANIZATION), "gave him a *Washington*" (MONEY), etc.]; and phrases that were not seen before [eg., *Spyro Agnew*].

FamilySearch has created a hybrid entity tagger which is based on coupling machine learning with post-processing rules. Specifically, the system attempts to tag 46 different entity classes: address, age, animal, chemical, gender-specifier, location referencer, coreference marker, anchored date, unanchored date, duration, historical event, religious event, personal event, four types of demonyms (locational adjectives), family member markers, associate markers, foods, games, health conditions, geopolitical entities, earth surfaces, non-earth places, structures, money, occupations, seven types of organizations, percents, people names, phone numbers, flora, quantities, times, titles, vehicles, weapons, websites, and works of art.

The machine learning portion of the FamilySearch entity tagger was constructed using a conditional random field (CRF) which is built using the Mallet system from the University of Massachusetts [3]. The CRF as we have constructed it leverages information about neighboring words, suffixes, case information, punctuation, major place names, places within a given locality, less-rare surnames, and less-rare given names. The CRF is trained using human-provided entity-tagged text which, for us, is a corpus of over five million words. The CRF training processing takes approximately 2.5-3 weeks to train on a single CPU and to the best of our knowledge, it cannot currently be parallelized.

At recognition time, after the trained CRF is applied to the raw text to get a first estimate of the entities contained therein, human-created rules are then applied to compensate for limitations in the CRF tagging. These limitations are usually due to insufficient context and/or to rare or previously-unseen situations.

When applied to obituaries, our entity tagger achieves 95-97% F-score. An "F-score" attempts to balance precision and recall and is defined to be the harmonic mean between the two. So roughly speaking for our system, a 96% F-score means that the system identifies about 96% of the entities that it is supposed to find; and of those it finds, it tags them with about 96% accuracy. An entity tagging system which achieves F-scores in the low 90s% is usually considered to be decent. Clearly, then, 96% -- despite the residual errors -- is a very high performer in the entity space. Even so, it is important to recognize that some of the most frequent errors that are made concern people names and place names ... which happen to be the most important ingredients for indexing.

Figure 1 shows the output of the entity tagger on a previously-unseen obituary. As can be seen from the illustration, our entity tagger is able to tag many of the phrases that appear in the obituary with the class of information represented by the phrase. However, the performance of our system is dependent on the kinds of training data it is provided with, so even though the system may perform well on obituaries, it is not guaranteed to have the same level of performance on some other unseen text genre.

Figure 1: Entity-Tagged Obituary

2.2. Identifying Relations

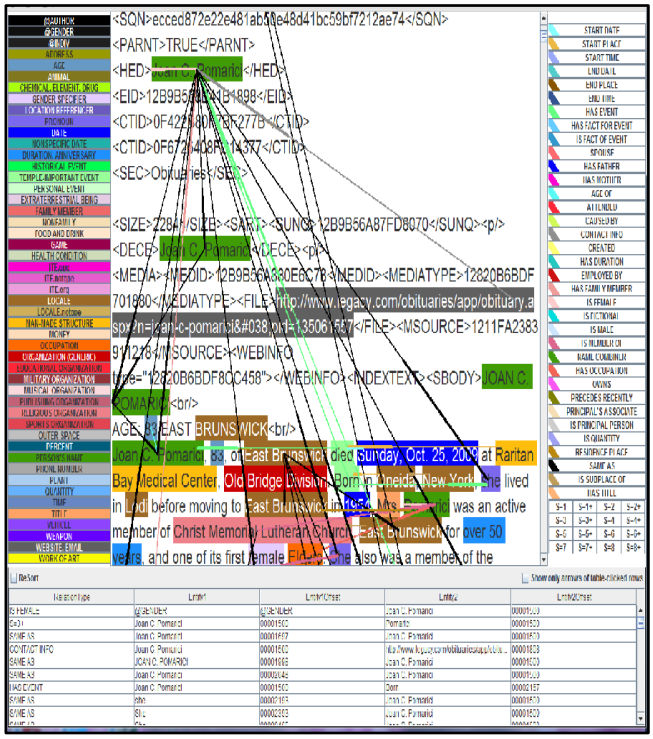
After entities are determined, it is next important to identify the associations between those entities. This is usually referred to as "relation tagging." FamilySearch has created a relation tagger which can wire together the automatically recognized entities. To be concrete about what relations are, suppose there is a phrase such as "Pebbles was born to Fred and Wilma Flintstone." This phrase

gives rise to at least the following relations: HAS_EVENT("Pebbles", "born"), HAS_FATHER("Pebbles", "Fred"), HAS_MOTHER("Pebbles", "Wilma Flintstone"), and HAS_SPOUSE("Fred", "Wilma Flintstone").

Like the entity tagger, our relation tagger is a hybrid system which begins with a machine- learned recognition phase followed by a sequence of human-provided rules. Prior to training, our system requires the existence of human-marked texts which indicate how entities are related to each other. Figure 2 depicts how this might look in that there is a pallette of relation types and the annotators pair together entities with lines that represent the connecting relations.

During training, we supply this human-tagged data as well as word usage statistics, word contexts, word similarity, and other kinds of features to a maximum entropy learner in order to discover when two different entities should be wired together by a relation. We take advantage of the maximum entropy learner found in the OpenNLP [4] package. This system is used in a first-pass relation recognition stage and it is followed by the application of rule-base corrections.

Figure 2: Relation-Tagged Obituary



It should be mentioned, though, that our relation tagger is perhaps a little different than what others might produce. Suppose the text says "John Smith passed on ....He was born Oct 13, 1901." Some systems might like to draw an association between the person "John Smith" and the date "Oct 13, 1901" and say BIRTH_DATE("John Smith","Oct 13, 1901"). However, we prefer to capture the individual sub-steps that lead to that conclusion. In our case, for example, we would attempt to capture HAS_EVENT("He", "born"), STARTING_DATE("born", "Oct 13, 1901"), and IS_SAME_AS("John Smith","He").

By doing this, we tend to capture information that is often closer together textually and therefore less likely to be in error and more likely to be learnable. It also means that with a reasonably smaller number of relational constructs (34 in our case), we can capture almost any information that we have interest in discovering.

Relation tagging is typically a more complicated process than entity tagging because more information is required before one can make a proper decision. From our estimates, our system achieves F-scores of about 90% on obituaries which means that many key relations are not discovered. Fortunately, there is sometimes redundancy in obituaries, so even though a relationship may not be accurately identified in one particular place of the text, the desired information may be ascertained somewhere else.

2.3. Name Disambiguation

In an obituary, the text may refer to "Robert Smith," or "Bob Smith" or "Robby" or "him" – and all these different text strings may co-refer to the same individual. For the sake of indexing, it is imperative that after we identify all of the instances of people that we cluster together those instances that refer to the same person. This is because we want to make sure that the index that gets produced only has one instance of each person. Furthermore, indexing guidelines require the indexer to provide an "OR" separated listed to identify the various name variants of an individual ("Robert OR Bob OR Robby Smith").

As was mentioned briefly in Section 2.2, one of the relations that the relation tagger has been given the task of finding is the "IS_SAME_AS" relation. When the relation tagger has run and has done pairwise similarity analyses on each instance of names and pronouns, we perform a head-finding algorithm to link each textual instance of an entity with its corresponding head – which to us is the earliest-appearing and longest name representing the entity. In the case mentioned at the beginning of this section, "him," "Robby," and "Bob Smith" would all point to the head phrase "Robert Smith."

2.4. Disregarding Irrelevance

Personal names appear in four kinds of contexts in obituaries. The most obvious way is that they appear (1) as the deceased or as an associate of the deceased. Yet the following phrases also illustrate other ways in which they appear: (2) "He was a serious fan of *John Wayne*." (3) "His funeral will be handled by *Bob Jones* and Sons."; and (4) "He liked to read "*Sherlock Holmes*."

When indexing is performed, the indexers are really only supposed to index the first type of name instances (which we will call *type-1* name instances). It is for this reason that we have attempted to recognize a number of odd types of entities such as a "Work of Art." We do not really care that a work of art like "Sherlock Holmes" has been mentioned, but rather, we want to avoid treating the embedded name as if it were the deceased or an associate of the deceased. In most cases, the entity tagger has helped us to be successful at eliminating type-3 and type-4 person mentions using this approach.

Type-2 mentions should be easier to eliminate because they are famous people names and one should be able to use a list for elimination. However, as we have observed human-tagging behavior, we see that they are often inconsistent when eliminating the names of famous people. As will be mentioned later, we score the Robokeyer by comparing it to people; so we have not eliminated famous names as often as we should have. Notable exceptions to this are the names of Jesus Christ and his early associates like His apostles, as well as Elvis Presley, and a few

other people. We have encoded these few specific names directly into our system to have it automatically exclude such results.

2.5. Detecting Gender

The indexing guidelines require that the indexer report the gender of the individuals contained in the obituary IF there is sufficient content to make such a determination. For example, if the document talks about "John Smith" and says "He was born..." then the indexers can conclude and report that John Smith is a male because the individual is referred to as "He."

For the Robokeyer, we first attempt to follow the same guidelines as humans do. If individuals are co-referent with a gendered pronoun (he, him, she, her, etc), we mark the gender of the individuals appropriately. Likewise, if the individual is referred to as "Mr." or "Jr." then we can also provide a gender.

When we are comparing our results to human results, we really can do little more to provide gender information. However, when we are indexing for release of the data to the public, we can further supplement the gender-decision process by making use of a tool which we created called the "genderAPI." The genderAPI is a statistical algorithm which uses FamilySearch's database information and all name parts of an individual to estimate the probability that the name in question is male. When it is able to make a decision, the genderAPI is rarely incorrect (probably less than 0.5% error); but for names like "Stacy," "Whitney," and others which could be more gender-neutral, it is not able to make strong assertions about gender.

Table 1 illustrates the performance of the genderAPI on the names of individuals who were in the news at the time this paper was written according to Google trends [5]. Individuals in the left half of Table 1 are males and those in the right half are females. Given that the genderAPI reports percentage of maleness, one might treat numbers of about 20% or less as being indicative of a female whereas those above 80% are likely indicative of a male. Based on these probability thresholds, one can see that on the Google trends individuals, the genderAPI was able to properly predict the gender of all but one of the individuals.

Table 1. GenderAPI applied to Names from 2016 News

Male Name	Prob(M)	Female Name	Prob(M)
Lamar Odom	94.0%	Ronda Rousey	5.2%
Donald Trump	99.7%	Ruby Rose	1.8%
Charlie Sheen	97.8%	Rachel Dolezal	0.3%
Brian Williams	99.7%	Adele	0.5%
Josh Duggar	99.3%	Caitlyn Jenner	0.2%
Bill Cosby	97.9%	Amy Schumer	0.3%
Taylor Kinney	80.5%	Rumer Willis	***82.5%***

2.6. Chunking Names

Up until now, we have mostly mentioned specific elements of texts that indexers needed to either distill out or disregard in order to index data properly. Yet indexing guidelines also require people to infer some information which is not expressed directly in the obituaries. One of these elements of inference is to determine when names pieces of the individual are surnames, given names, or titles. Obviously there is some subjectivity to this task, but humans do it fairly well.

For the sake of the Robokeyer, we have also created technology to do this task automatically. We refer to the technology as a "name chunker" which attempts to identify given name phrases, family name phrases, titular phrases, ordinal phrases, and so forth.

The technology is basically an entity tagger unto itself. Rather than marking whether a phrase is a person or location, our name chunker treats the name as a stream of text where it must identify the name-entity roles of each word. Our system uses a bigram hidden Markov model to learn name pieces. The chunker can handle many forms of personal names, and it can even work when names appear in reversed order, where surnames appear first; and it can likewise handle nicknames like Richard "Fuzzy" Jones. For names that come from regions of the world where multiple surnames are used, the chunker can also identify the various surname phrases; so "Juan Antonio de la Rosa de la Garza" gets broken up into a given name phrase "Juan Antonio," a surname phrase of "de la Rosa" and another of "de la Garza." Our evaluations suggest that the chunker can accurately parse names with at least 98% F-score when used in isolation.

That said, names do not appear in isolation. For example, a set of names might appear in context as follows: "Mildred, Howard, Robert, and Samantha." It may be the case that "Howard," when used in isolation, is more likely to be a surname than a given name – so our first stage of tagging might mark it thus. However, given that it is in a context where the words surrounding it are all given names, we may conclude that "Howard" is a given name. Thus, we use a second stage of processing which tries to consider the context in which the personal name appears in order to determine if the first stage's result should be overridden.

2.7. Relationship Mathematics

As mentioned before, we try to identify basic relations where the entities that are being connected together with a relation are often fairly close to each other text-wise. Sometimes these sub-relations have to be composed together in order to identify a more complex relation. Family relationships between the deceased and the other people in the obituary is a type of more complex relation which typically involves the combination of multiple sub-relations. For example, suppose the obituary states "*John Brown...is the father of Richard (Nancy) of Boston and Lewis (Anne) of Chicago*." Indexing guidelines require that the indexer must report the family relationship of John Brown to Richard, to Nancy, to Lewis, and to Anne. Under a perfect relation-finding scenario, our system would have created the relation MEMBER_OF("John Brown", "father") and it would have noted that "Richard" and "Lewis" both participate in a relation of HAS-A with the term "father." The system would also identify the relations HAS_SPOUSE ("Richard", "Nancy") and HAS_SPOUSE("Lewis","Anne"). So the question is: what family relationship does one attach to Richard, Lewis, Nancy, and Anne?

To determine the family relationship that should be ascribed to Richard and Lewis, we have created an *inverse-of* functionality. These individuals are ascribed the relationship inverse-of(father). The Robokeyer has been informed that inverse-of(father) equals "child" or, if accounting for gender, it could likewise be "son." Thus the two males are given this family relationship of "son."

The relationship for Nancy and Anne is determined by the "*spouse_of*" function. In essence, this means their relationship is spouse_of(inverse_of(father)). If we simplify based on the equation from the paragraph above, this would be spouse_of(child) or, taking into consideration gender, spouse_of(son). The spouse of a child is "child-in-law" and of a son is "daughter-in-law;" so

these would then be the relationships with which the system would tag Nancy and Anne.

2.8. Principal Detection, Person Multiples and Non-Obituaries

The millions of obituaries that were identified for processing were selected through keyword searches by the company that owns the rights to the obituary data. Their process for identifying obituary articles was effective, but it was not 100% accurate. In addition to actual obituaries, this process identified news stories about death, stories where multiple people had passed away, and stories where no human death is mentioned at all (eg., one sports team may have 'killed' another team).

When a story is reported where there is no valid obituary content, indexers have been asked to report "No Extractable Data Image." For all other stories, indexers are supposed to identify the principals – which are the people who are the subjects of the story *and* who are recently deceased. Both "no extractable data" and the list of principals are not expressly marked by the text itself, so these are items that must be inferred by the indexer.

To make the Robokeyer perform this function, we first used some amount of training data to identify phrases that appear more often in no-extractable documents than in ones where extraction can occur. Phrases that are particular indicators of no-extractable data were strings such as "Lotto :", "Pick Four," "chance of showers," "Saturday partly cloudy," "plaintiff alleges negligence," "commit simple assault," "Senior center menus," and "Public Library will." One can quickly see that these phrases are indicative of weather reports, lotteries, crime logs, and public service announcements.

Though this was helpful, we sought techniques which could significantly enhance our ability to determine whether there would be zero, one, or multiple principals in a given story. Using different features but the same maximum entropy toolkit that we had used in creating our relation tagger, we developed a function that could make the 0/1/2+ principal categorization with over 98% accuracy. Something of particular note in this process was that some of the seemingly unnecessary entity classes mentioned before turned out to especially useful here. For example, "weapons," "vehicles," "chemical," and "flora," were indicators of military stories (eg., jets, tanks, grenades, etc) and of crime logs (eg., drugs, marijuana, etc) where a crime was perpetrated but no death was mentioned.

2.9. Beyond Human Requests

It was mentioned that since the genderAPI achieves high levels of accuracy in predicting the gender of a person by their full name string, we had been asked to have the Robokeyer provide gender information even when there were no clear textual indications of gender. We were likewise asked to have it pull out information that is too costly to have humans do, such as identifying the residences of the individuals in the stories, peoples' occupations, their organizational memberships, information about funerals, and so forth.

Yet there are two pieces of inferential information that are hard for people to extract when not directly stated but which are extremely important for localizing when. These are: (a) establishing what a particular day of the week probably equates to when it says "This person passed away on *Thursday*" given that you know the obituary is being produced (for example) on 26 January 2015; and (b) determining what city an event took place in when it says "This person passed away at Mountain View Hospital" when the newspaper is the Salt Lake Tribune. To do this kind of inference would significantly add to the human costs of indexing, but it is fairly straightfoward for the Robokeyer.

The first of these, determining what would be the likely calendar date something occurred on given an anchoring date, requires one to first coerce the full anchoring date into a Calendar object. Then, the first instance of the unanchored date (eg., "Thursday") prior to that anchored date is treated as the resolution for the unanchored date. Concretely, if the day of newspaper publication is 26 January 2015, we can determine using calendaring that this means *Tuesday*, 26 January 2015. If a person passed away "Thursday," we recognize that the Thursday that is closest *before* the anchoring date was five days previous; so we would report that death occurred on 21 January 2015. A human could likewise make this judgment, but doing the calendar math would greatly increase the indexing cost.

To determine an event's city using only the hospital, facility, or organization, we first identified thousands of structures that are referenced in obituaries and we paired those with their associated newspaper. For those places that repeated sufficiently often, we painstakingly searched the Web to determine if we could identify a single unique place by that name in the newspaper's locality or in the whole United States. For those facilities with unique names in their particular newspaper or national localities, we stored the full address. Our resolved data set was small, but it still consisted of over 5000 location-anchored facilities. For instance, in the vicinity of the Salt Lake Tribute, Mountain View Hospital is unique and it is located in Payson, Utah.

3. Turning Pieces Into Indexes (Conversion)

The output of the entity, relation, and other taggers need to be massaged and, in some cases, seriously manipulated in order to actually produce a human-like index. When people do the indexing of the obituaries, they are asked to identify the following pieces of information:

- Type of record: Deceased or Other
- Given Names and Surnames for all unique individuals
- Titles/Terms for all individuals (eg., "Jr," "Mr" "Mr")
- Gender of the deceased
- Birth/Death day, month, and year of the deceased
- Birth/Death city, county, and state of deceased
- Age of the deceased
- Name of the newspaper
- Relationships of non-principals to the deceased.

The Robokeyer needs to find these same pieces. When the entity extractors have worked perfectly (which it often is but not always), the identification of the people names is straightfoward – so the key difficulties are to determine what the distinction is between given names, surnames, and titles and tags AND to determine when names are unique. The name chunker does a good job at making the given/surname distinction in isolation, but as was mentioned earlier, the converter needs to allow chunking to be overridden when it appears in context. Something that was not mentioned was that the converter also needs to be taught that when names appear like "Robert and Mary (Brown) Jones," then Robert's surname is "Jones."

To properly extract the birth and death facts, one of the two following options need to happen. In both options, the entity tagger needs to find the phrases that indicate places and date. Additionally, in the first option, the entity tagger must also find a

phrase that is indicative of the event (eg., "died", "returned to the arms of his Father in Heaven", "passed away," etc.) , and then the relation tagger must link the individual to the event and then link the event to its associated date and place. In the second option, the relation tagger must draw a direct connection with the person and the relation "HAS_START_PLACE/ HAS_START_DATE" to identify birth information; or "HAS_END_PLACE/ HAS_END_DATE" for death information. The first of these options has been favored in the training data and the second is only used when there is no anchoring event phrase. So for the most part, three kinds of entities and three separate relations must be discovered in order to find all the relevant pieces for each event.

The next piece to identify, gender, has been spoken of earlier. Specifically, if there are any textual information that gives clues about gender, we use those clues. If not, we leverage direct predictions of the genderAPI.

Lastly, and often most difficult, we need to determine whether the people involved are deceased or "other," and if other, was is the relationship to the deceased. Typically there is one word that lets the obituary reader know the relationship between the deceased and a whole list of individuals. In fact, it can even be the case that no words are used to describe the relationship and that it must be derived from the relationship mathematics that was mentioned earlier. Many of the obituaries identify family relations in a way similar to the following: "*Bob Jones....is survived by....his three sons, Rudolph (Ann), Rupert, Theodore (Susan), Shelley, and Alfred (Jane), Burley*." As we saw before, if we can determine that the sons are Rudolph, Theodore, and Alfred, then using relational math, we can determine that Ann, Susan, and Jane are daughters-in-law. Unfortunately, it is often the case that the constructs of how the family members appeared makes it hard to know who actually *are* the relatives. In the case of the sentence fragment above, even a human might think that the sentence should be telling us that Bob has two more sons, Rupert and Burley, and a daughter named Shelley. But these three extra words are really referring to towns in Idaho and the obituary writer is telling us the residence places of the sons' families. Given these complexities, the determination of the relationship between the family members and the deceased is non-trivial and has the highest amount of error. Moreover, the associated people may also be business partners, friends, church officials, etc., and in these cases, the indexer is supposed to mark the connection to the principal deceased as "non-relative."

Given the complexity of inferring the relationship, it is often the case that the converter finds that there is insufficient information to determine the relation. In many of these cases, the converter makes an educated guess. To make this guess, we use the statistics of the gender of the name string and the relationships or suspected relationships of people before and after the unknown relationship. For example, if the relative before the current unknown one was a sister and the relationship afterward is not yet determined, but the current person's name is female, then there is a 65% chance that the unknown relationship is actually "sister." We would indicate this rule by "Sister__?_F => Sister." The converter has almost 500 rules of this kind which do introduce error but which yield the correct relationship more often than not.

The last required scoreable field is that of AGE. The entity tagger distinguishes between QUANTITY, DURATION, and AGE. We use its AGE output to fill this field, though the converter may have to converter a string like "sixteen-and-a-half years old" into "16" as required by indexing guidelines. This information tends to be fairly accurate. Moreover, if we have two elements of the set {BIRTH DATE, AGE, DEATH DATE}, the converter estimates the other missing piece of information.

The name of the newspaper is another desired piece of information for indexing, but since there are external metadata that are available that can be used to supply this information, it is not absolutely required. Nevertheless, the Robokeyer has an entity class ORGANIZATION.pub which identifies publications and we mine the results in order to provide the publication name. This entity class typically fires in the header portion of the raw XML obituary text to identify the name of the newspaper and its location.

When the Robokeyer is being scored, it is only compared to the human-extracted information we have mentioned in this section so far. Yet the Robokeyer also pulls out other relevant information about residences of individuals, burials, marriages, information about the funeral, occupations, and organizational affiliations. Its quality for extracting this information is unknown since there are no gold standard results with which to make comparison.

4. Evaluating Performance

We evaluate the Robokeyer by comparing its output to indexes created by humans on the same obituaries. Humans of course make errors, so at some points, our system will get penalized for correct results. Nonetheless, this strategy provides us with a reasonable estimate of the system's performance and it points us to places where there are problematic outcomes. We will here describe the data sets upon which we evaluate, and we will show the actual way we count human versus machine disagreements.

4.1. Corpora types

We evaluate the Robokeyer on three different data sets. The first of these, which we will call ***Short Septuple***, is a 4K data set of mostly short obituaries. The data set was vetted seven times by humans – so results thereon can be taken as highly likely to be accurate. The second set, which we will call ***MediumObits***, is probably most representative of the typical obituary collection in that it contains 600K average-sized obituaries that were triply indexed by humans. The last set, the ***LargeObits***, is a set of 880K obits which tend to be medium to large in size and which represent samples from all 50 states and which was also triply indexed.

4.2 Evaluation Strategies

To score the Robokeyer, we compute the average per-field F-score. Suppose there are 30 extractable fields in an obituary and that the Robokeyer correctly finds 25 of these, fails to find 2 of them, inserts incorrect information into 3 others, and produces spurious results for 4 more. In this case, the precision would be given by 25/(25+3+4)=78.1%. The recall for this situation would be 25/(25+2+3)=83.3%. The F-score, which was declared to be the harmonic mean of the precision and recall (or the product of the these two numbers divided by the average of the two), would be given to be 80.6%.

There are some kinds of differences between hypothesized results and "gold standards" that are mostly irrelevant in that either they are unlikely to affect search findability or they are subjective. We do not impose a penality for these types of variability in scoring, which are:

- **Ordering of Appearance of Names**
  Eg. Human says John, then Mary; but other indexer (human or machine) shows Mary first followed by John. [Note that we detect these swaps using the "Hungarian Algorithm", [6].]

- **Case Insensitivity**
  Eg. Human says John E; but other indexer says John e
- **Gendered Relations Agrees with genderAPI**
  Eg. Human says the relationship is "Child"; but other indexer says "Son" and genderAPI says the person is a male.
- **Period-stripping**
  Eg. Human says "Mr" but other indexer indicates it as "Mr."
- **Spacing Variations**
  Eg. Human says "Jan VanSanten" and other indexer reports "Jan Van Santen."
- **GN/SN Boundaries**
  Eg. Human says that the name "Michelle Fredrickson Smith" should be broken up into given/surname pieces as Michelle Fredrickson /Smith/ but the other indexer breaks it up as Michelle /Fredrickson Smith/.
- **Common Name Swaps**
  Eg. Human reports a place name as "AZ" but the other indexer reports it as "Arizona."

There is also a situation where partial credit might be afforded even when there are differences between the human result and the other indexer. This is:

- **Handling "OR" names**
  Eg. When human says the person's given name is "James or Jim" and the machine reports only "James," then scoring gives F-score-based credit to machine ("James" is 100% precise, but since only one of the two name variants is provided, "James" is only 50% of the recall ... yielding 66.6% F-score which percentage is also used as the partial credit).

We refer to the method of scoring which permits the above variations as "Level-8" normalization. One could easily argue that two indexes that differ only on these points are virtually the same for all intents and purposes. Level-8 scoring is our primary method for evaluating the machine against the human indexes.

We also consider a "Level-10" normalization where non-key fields are dropped (such as the death day and month) and where FamilySearch's naming authority tables are used to identify name variants like "Johnny" and "John;" so if one indexer says the first and the other indexer says the second, this difference could be potentially overlooked. This method of scoring shows the searchability differences between the two indexes.

4.3. Performance Results

Given the described method of scoring and the strategy for handling close variations, we can now show the Robokeyer's performance on each of the three test collections mentioned in Section 4.1. We will first show the raw results of the system as it is applied to ALL obituaries and then as it is applied to only those obituaries which the confidence predictor has surmised are sufficiently "sweet" (where the accuracy threshold is estimated to be at least 75% per-field F-Score).

4.3.1. Raw Results

Table 2 shows the per-field F-scores of the raw Robokeyer with no "sweetening." The column of "Medium Obits" is probably most representative of new collections since most should tend to be of medium size, and the "Level-8" scoring is probably most similar to the perception of accuracy that a human might place on the results. In that collection, there are an average of 8 people who are mentioned in each obituary.

Table 2: Robokeying Per-Field F-Scores On Each Test Set

Scoring Paradigm	Short Septuple (143.4K fields)	MediumObits (10.9M fields)	LargeObits (44.9M fields)
Level-8	95.10%	94.09%	90.97%
Level-10	95.86%	94.74%	91.69%

Note that for Level-8 on the unfiltered MediumObit collection, we are getting 94.09% F-score on average per field. As was mentioned, this set has 600K obituaries which represent 10.9M fields, or about 18.1 fields/obituary. Since we have 18 fields and, on average 5.91% deletion and insertion error per field, we should on average expect to see 1.06 precision error and 1.06 recall error in almost every obituary.

We can get a better understanding of these results if we look at them pictographically. In Figures 3 and 4 we illustrate histograms for the Medium and Large Obituary sets. The X axis in the plots represents that averaged per-field F-scores of 100-obituary blocks of data; and the Y-axis is the log-2 scaling of the frequency (i.e., 1, 2, 4, 8, 16, etc) of blocks that share the same F-score. Figure 3 shows the histogram of the MediumObits collection and Figure 4 shows it for the LargeObits.

Figure 3: Log-Scaled Histogram of Averaged F-scores of 100-obit blocks from MediumObits

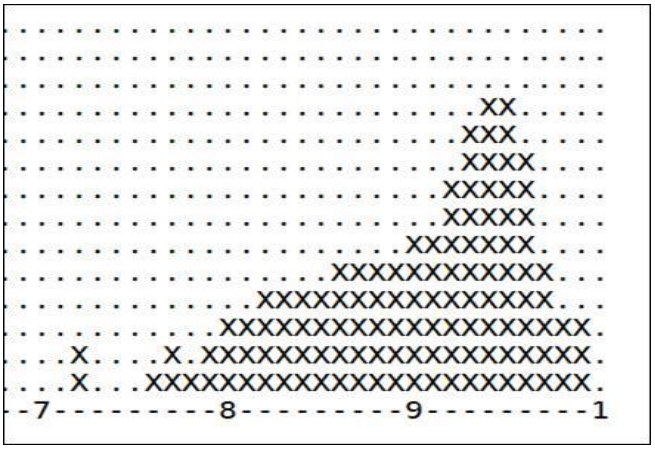
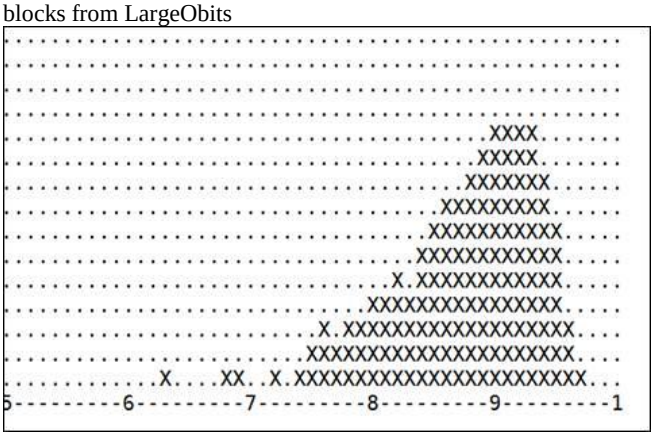


Figure 4: Log-Scaled Histogram of Averaged F-scores of 100-obit blocks from LargeObits



We can see from the figures that the modes for both Figure 3 and Figure 4 are very close to the performance numbers reported in Table 2. Yet it is also clear that there are long tails to each of these

distributions. In Figure 2, for example, we see that are two 100-obit blocks which have averaged F-scores of only 72% -- a very low score for so many obituaries. In Figure 3, we see likewise that there is a 100-obit block with averaged F-scores in the low 60s. Fortunately, as we see in the next section, confidence-based filtering helps reduce some of these long tails.

4.3.2. Results with Automatic Confidence Judgments

As was mentioned earlier, we had built a machine learning process to automatically attempt to determine when the Robokeyer was likely to have done a reasonable or poor job at indexing. Of course, this system cannot be flawless, but there are definitely clues that can be exploited for making predictions. For example, we have some colleagues who evaluated Robokeyer results and determined that if there are more than a certain number of children in the obituary, or too many wives, or too many principals, the obituary is likely to have been auto-indexed poorly. We provided these and many other properties to a maximum-entropy-based machine learner that was trained using a small fraction of the *test* data and allowed it to try to determine when it was likely to have failed. Of course, the use of some of the testing material for building this machine learner has the potential of obscuring its true value or lack thereof, but we assumed we would be able to have a good sense of its helpfulness based on what it did on the test set as a whole. In Table 3, we see that by throwing out all the elements which were automatically predicted to be poorly indexed, we achieved 1.1-1.8% absolute gains in F-score – a definite win! This should mean that the average obituary for this major subset of obituaries has dropped from having 1.06 precision and recall errors to having 0.86 errors of each kind.

Table 3: "Sweet" Robokeying Per-Field F-Scores On Test Sets

Scoring Paradigm	Short Septuple (137.0Kfields)	Medium Obits (10.2M fields)	Large Obits (39.5M fields)
Level-8	95.66%	95.24%	92.71%
Level-10	96.36%	95.84%	93.21%

If we again look at histograms of the higher=confidence results, we see that much of the worst part of the tails have been removed. Figure 5 shows the histogram using the MediumObits collection and Figure 6 shows it with the LargeObits collection.

Figure 5: Log-Scaled Histogram of Averaged F-scores of 100-obit blocks from MediumObits After Confidence Filtering

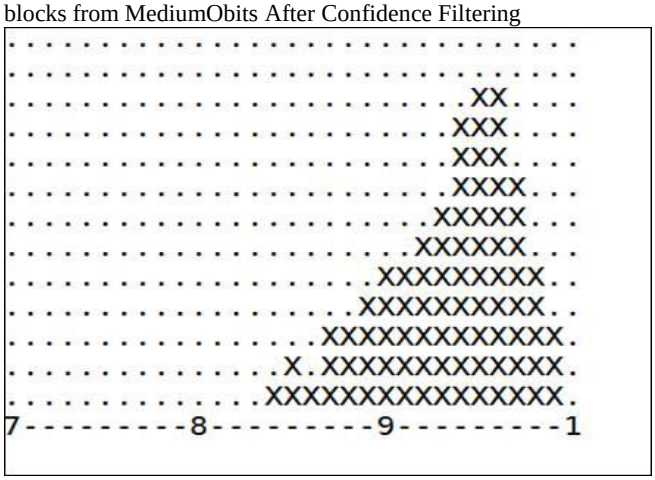
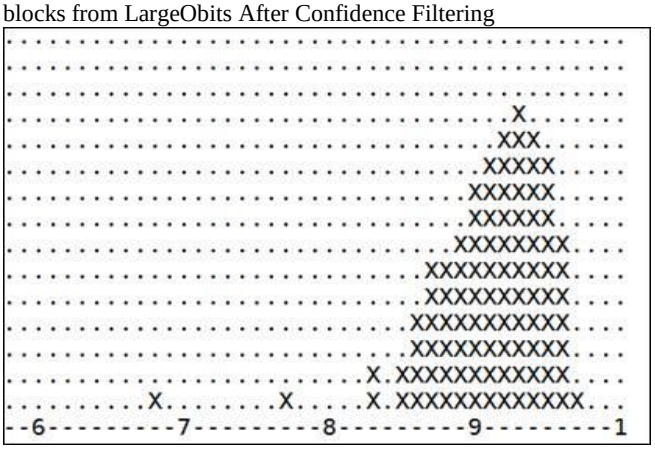


Figure 6: Log-Scaled Histogram of Averaged F-scores of 100-obit blocks from LargeObits After Confidence Filtering



5. Scaling for Production

The test sets we used for Robokeying evaluation were large, so we expected that the system should be about as equally applicable to previously-unseen obituaries as it had been for these which had already indexed by humans. FamilySearch had a collection of 48 million obituaries which it had intended to have indexed fully by humans. Yet for every million obituaries that need to be indexed even using the fastest human indexers (assuming 8 minutes of work per obituary), 15.2 person-years are required (or about 65.5 person-career-years, assuming 2080 hours worked per year). Since 26.5 million obituaries had not been human indexed, these were all run through the Robokeyer.

The Robokeyer was run on a compute cluster with 145 two-generation-old machines using three CPUs each. In about five days, it was able to index the 26.5 million obituaries, run confidence filtering, and identify that about 23.5 million of the obituaries were likely to have sufficiently high quality for publication. This means that the Robokeyer was able to index in five days the same amount of data as would have taken 1539 career-years for a single human to index.

6. Addressing Errors: Future Work

The Robokeyer's speed makes it appealing as a solution to born-digital obituary indexing. Yet the system errors are something that still need to be addressed in some alternative manner in order to have perfectly clean indexes. As was mentioned, probably 50%-60% of the errors that the system makes have to do with ascribing the wrong family relationship to the non-principals of the obituary. Another 20-30% of the "errors" have to do with name interpretation (some of which are actually original indexer errors as opposed to Robokeying one). These often occur because some names appear in reversed form while others appear juxtaposed with the event city (eg: SMITH – Richard Brown vs. OGDEN – Richard Brown). Another reasonable fraction of the errors are caused by failing to determine the correct number of principals. It is reasonable to believe that with further work on the Robokeyer and with clean up of the truth sets, these kinds of errors can be removed. Yet even if not, many of these errors are not egregious.

There are some errorful situations, though, which are either comical or, in some cases, are potentially upsetting. We cannot mention all that have been reported to us thus far, but we can site a few.

6.1. Comical Character Errors

It was mentioned earlier that the entity tagger can identify works of art. Yet *characters* from works of art have no specific markers that identify them as such. Therefore, nothing precludes them from being identified as people. We have since introduced a relation into our system's inventory of "is fictional," but this relation still has almost no training data to help reduce this kind of error. Consequently, we have pulled out TV and comic strip characters as if they were real people. For example, when the creator of the Rocky and Bullwinkle cartoon died, his characters were mentioned by name in the data. The Robokeyer indexed these names as if they were real people. At the time of this publication, these errors were still visible online [see, for example, 7, for "Bullwinkle J Moose"].

Pop star Michael Jackson also died in recent years and his obituary and/or stories of his death appeared in numerous newspaper articles. In one of these stories, the character of his song, "Billie Jean," was reported as his sibling by an early version of the Robokeyer, along with his true siblings Jackie, Tito, Jermaine, and Marlon [see 8]. In fact, the Robokeyer actually did a relatively poor job in indexing the rest of this death story. To help remedy this for the future, we purposefully looked for and tried to train the entity tagger and relation tagger for stories about Michael Jackson and other famous people.

6.2. Error Severity as Perceived by System Users

Other kinds of errors have been reported by patrons which suggest a certain amount of alarm. Since obituaries cover people within the time frames of living memory, FamilySearch patrons may be mentioned by name in obituaries. They have gotten understandably upset when the system has mismarked their particular family relation to a loved one (such as erroneously calling the patron the *spouse* to a beloved grandparent). A few patrons have asked questions on some obituaries like "Did a child index this one?" and others have commented that the person who did the indexing needs to "Get a clue!" Obviously, strategies for rapidly improving Robokeyer errors is of interest. Current thoughts are that we could create simple tools to allow specific FamilySearch affiliates to fix the serious errors that have been identified by the patrons.

7. Feasibility of Robokeying of Obituaries in Newspaper Images

The potential success of the Robokeyer on born-digital obituaries has led individuals to ask: Can the Robokeyer be applied to obituaries that first appeared in the newspaper or "born paper?" Naturally, a first step in Robokeying stories from actual newspaper images is to convert those images into textual representations through OCR. "OCR" or "Optical Character Recognition" is the phrase that is usually used to indicate the automatic transcription of images.

OCR technologies exist commercially and many genealogical companies use these commercial tools to gain textual access to their own content. However, OCR companies have primarily focused their attentions on the kinds of documents and print that has appeared in the past two decades. Print over the last twenty years is often fairly high quality and uses even inking with consistent font sizes. Moreover, image scans from recent newspaper clippings are typically very readable. Nonetheless, newsprint of the deeper past had a number of issues which make automatic transcription thereof much more difficult.

We have therefore worked to develop our own historical newsprint recognition system. In particular, we know that unless we have word accuracies on obituaries that exceed at least 90%, the resultant Robokeying results will be of little or no value to patrons. This system is described in detail elsewhere (see [9]). Yet suffice it to say that our system's performance on pre-zoned US newspaper snippets is fast approaching 90% and it has already exceeded 90% on British newspapers. The key for Robokeying success on these newspaper image snippets will be high quality transcription of people and place names which often are exactly the kinds of terms that OCR systems fail to transcribe well. We are hopeful, though, that we will be able to achieve high-quality OCR results and that Robokeying thereon will be a success.

8. Synopsis

We have here described our Robokeying system which has been used to automatically index born-digital obituaries. We have shown that we have been able to get the system to have reasonably high accuracies at doing the same task that humans normally do, but it can do this thousands of times faster than humans. This speed has led us to apply this technology to tens of millions of obituaries for publication which will become available to the public in the early parts of 2016. Moreover, due to the potential success of this technology, we have pushed for and made significant strides in the development of OCR technology for transcribing historical newsprint with plans to start Robokeying on image-based obituaries by the end of 2016.

9. Acknowledgements

Although this paper has focused on the technological aspects of robokeying, there have been many other components that have been necessary in order for the Robokeyer to reach a state in which its results could be deployed. These have included issues of management negotiations, data acquisition, pre-publication operations, external evaluations, and image processing work. The authors wish express their thanks to Jon Morrey, Laura Giometta, Mark Hamp, and Heather Walgren for management negotiations; to Nate Emmer, Chris Whitehead, Rachael Day, Jeremy Schone, Janae Dean, Melvin and Miriam Pethel, Roger and Marilyn Smith, Kenneth Bailey, and Robin Davis for significant efforts to annotate and/or acquire data for training and evaluating the Robokeyer; to Tom Robbins, David Greenwood, and Matthew Larson for their efforts to transform Robokeying results into productized data; to Janet DiPastena, Scott Chatterton, and Daniel Williams for independent evaluations of this technology; and to Alan Cannaday, Seth Stewart, and Heath Nielson for their help in transforming images into robokeying-enabled documents.

References

[1] K. Gale. "2014: The Year of the Obituaries," at https://familysearch.org/blog/en/2014-year-obituaries-2/, 2014

[2] R. Grisham, B. Sundheim. "Message Understanding Conference 6: A Brief History", COLING-96, 1996.

[3] A. K. McCallum. "MALLET: A Machine Learning for Language Toolkit." http://mallet.cs.umass.edu. 2002.

[4] K. Kottmann, et al. "openNLP." http://opennlp.apache.org/team.html, accessed 27 Jan 2016.

[5] Google Trends. https://www.google.com/trends/topcharts#vm=cat&geo=US&date=2015&cid, accessed 27 Jan 2016

[6] H. W. Kuhn, “The Hungarian Method for the Assignment Problem and how Jacobi beat me by 100 Years”, Seminar, Concordia University, September 12, 2006.

[7] FamilySearch, “Bullwinkle J Moose,” https://familysearch.org/ark:/61903/1:1:QVTP-RL7H, accessed 29 Jan 2016.

[8] FamilySearch, “Billie Jean,” https://familysearch.org/ark:/61903/1:1:QVT4-CZ82., accessed 29 Jan 2016

[9] P. Schone, A. Cannaday, S. Stewart, R. Day, J. Schone. “Automatic Transcription of Historical Newsprint By Leveraging the Kaldi Speech Recognition Toolkit,” DRR 2016, to appear.