Intelligent Indexing: A Semi-Automated, Trainable System for Field Labeling

Robert T. Clawson

Brigham Young University - Provo

Follow this and additional works at: https://scholarsarchive.byu.edu/etd

Part of the Computer Sciences Commons

BYU ScholarsArchive Citation

https://scholarsarchive.byu.edu/etd/5307

This Thesis is brought to you for free and open access by BYU ScholarsArchive. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of BYU ScholarsArchive. For more information, please contact scholarsarchive@byu.edu, ellen_amatangelo@byu.edu.
Intelligent Indexing: A Semi-Automated, Trainable System for Field Labeling

Robert Clawson

A thesis submitted to the faculty of Brigham Young University in partial fulfillment of the requirements for the degree of Master of Science

William Barrett, Chair
David Embley
Eric Mercer

Department of Computer Science
Brigham Young University
September 2014

Copyright ©2014 Robert Clawson
All Rights Reserved
ABSTRACT

Intelligent Indexing: A Semi-Automated, Trainable System for Field Labeling

Robert Clawson
Department of Computer Science, BYU
Master of Science

We present Intelligent Indexing: a general, scalable, collaborative approach to indexing and transcription of non-machine-readable documents that exploits visual consensus and group labeling while harnessing human recognition and domain expertise. In our system, indexers work directly on the page, and with minimal context switching can navigate the page, enter labels, and interact with the recognition engine. Interaction with the recognition engine occurs through preview windows that allow the indexer to quickly verify and correct recommendations. This interaction is far superior to conventional, tedious, inefficient post-correction and editing. Intelligent Indexing is a trainable system that improves over time and can provide benefit even without prior knowledge. A user study was performed to compare Intelligent Indexing to a basic, manual indexing system. Volunteers report that using Intelligent Indexing is less mentally fatiguing and more enjoyable than the manual indexing system. Their results also show that it reduces significantly (30.2%) the time required to index census records, while maintaining comparable accuracy. A helpful video resource for learning more about this research is available on youtube through this link: https://www.youtube.com/watch?v=gqdVzEPnBEw

Keywords: Image Processing, Document Image Analysis, Green Interaction, Interactive Learning, Document Indexing, Machine Learning
ACKNOWLEDGMENTS

Producing this thesis has been soul-stretching for me, and would not have been possible without a couple key players. I express my sincere gratitude to Dr. Barrett for providing key insights, motivation, and counsel as we explored ideas together, developed the ideas presented in this thesis, and demonstrated our research through conferences, meetings, papers, and posters. Dr. Barrett provided me opportunities to learn and grow in ways for which I will always be grateful. I also thank Dr. Barrett for funding my research. I couldn’t have chosen a better mentor.

I appreciate our friends at FamilySearch, including Jake Gehring, Alan Cannaday, Pat Schone, and Jon Morrey for providing ground truth data and expressing interest in our research.

My gratitude goes out to Milan, Glen, Kevin, Doug, Ian, and Dr. Barrett for our collaborative effort on the DRR paper, and to Kevin as well for work on the document clustering competition.

I am also grateful to Doug Kennard for all his help with the Word Warping code and for coding help in general. I also thank Doug, Cameron, Joel, Josh, and others in the lab with whom I shared many interesting conversations and laughs.

Many people stepped up and volunteered their time as indexers for the user study. I am very grateful for their effort.

I thank Dr. Embley for his valued insights on my project, and for helping me improve my writing. I also thank Dr. Embley, Dr. Mercer and others who provided feedback at my thesis proposal.

On a more personal level, I thank my parents, Glenn and Toni Clawson, for believing in me and showing interest in my research, and my Father in Heaven for providing me answers when I didn’t know where else to turn. Finally, I appreciate my patient wife, Alison, who learned to stop asking when I would graduate, who trusted in me, encouraged me, and who bore the lion’s share of the burdens at home while I devoted time to this research.
# Table of Contents

List of Figures ........................................... vii

List of Tables ............................................. x

1 Introduction .............................................. 3

2 Related Work ........................................... 7
   2.1 Work with Similar Objectives ......................... 7
   2.2 Active Learning .................................... 8
   2.3 Incremental Learning ................................ 8
   2.4 Interactive Learning ................................. 9

3 Intelligent Indexing .................................. 11
   3.1 Data .................................................. 12
      3.1.1 Image Preprocessing ............................ 12
      3.1.2 Metadata ........................................ 14
      3.1.3 Precomputing Morphing Costs .................. 15
   3.2 Interactive Field Labeling ......................... 18
      3.2.1 Recommending Fields for Automated Labeling 19
      3.2.2 Visual Consensus ................................. 20
      3.2.3 Changing a labeled field ...................... 22
      3.2.4 Removing a field from the preview window .... 22
      3.2.5 Thresholds ...................................... 22
3.2.6 Transitive learning .................................. 24
3.2.7 Smart recommendations ............................. 25
3.3 Indexing Client ........................................ 27
  3.3.1 User Interface ...................................... 27
  3.3.2 Minimizing Context Switching ................... 29
3.4 Summary ................................................ 31

4 Results ..................................................... 33
  4.1 Experiment Design .................................... 33
    4.1.1 Indexing Server .................................. 34
  4.2 Experimental Results .................................. 35
  4.3 Quantitative Results .................................. 36
    4.3.1 Indexing Time .................................... 36
    4.3.2 Indexing Accuracy ................................. 37
    4.3.3 Indexing Automation ............................... 38
  4.4 Qualitative Results ................................... 39
  4.5 Visual Results ........................................ 39
  4.6 Summary and Conclusion .............................. 40

5 Future Work .............................................. 47
  5.1 Difficult Categories .................................. 47
  5.2 Between-Category Dependencies .................... 48
  5.3 Learning the Threshold ................................ 49
  5.4 Optimal Ordering of Labeling ....................... 49
  5.5 Better Preprocessing .................................. 50
  5.6 Alternative Uses for the Training Set ............... 51
    5.6.1 Deselection Knowledge ........................... 52
    5.6.2 Using Characteristics of Labeled Data .......... 52
# List of Figures

1.1 FamilySearch Indexing in use to label a 1911 Canadian census. 4
1.2 Historical documents with differing structure. 5

3.1 Raw 1920 Census Image. 13
3.2 Preprocessing for background removal and image enhancement. 13
3.3 Rotating the image. 14
3.4 Document Metadata. 15
3.5 Organization of Cost Matrices. 17
3.6 Example cost matrix. 18
3.7 Before and after the indexer labels a field. 19
3.8 Coloring categories fields and column headers. 20
3.9 Before and after the indexer deselects a recommendation. 21
3.10 Threshold analysis graph. 23
3.11 Before and after the indexer adjusts the threshold. 24
3.12 Interactive, transitive learning from an incremental training set. 26
3.13 Matches to the training set are a separate prompt that occurs after a labeling event. 27
3.14 Intelligent Indexing user interface. 32

4.1 Histogram of times for indexing pages using basic indexing and Intelligent Indexing. 37
4.2 Scatterplot showing correlation between automation rate and time to index a page. Calculated for all pages indexed using Intelligent Indexing, with outliers removed. 38
4.3 Mismatched fields tend to cluster in the back because the list is sorted by match strength. This is quickly remedied by adjusting the threshold slider to remove the “son”s.

4.4 A degraded page made recognition difficult.

4.5 Even the gender category can be difficult with degraded handwriting.

4.6 When there is a large amount of repetition, the preview window is still helpful even in conditions where recognition is difficult.

4.7 Preview window negative example

4.8 The small vocabulary, single character fields worked out very well in general.

4.9 The Sesame Street principle (see Section 3.3.1) is well demonstrated in this example. In this case the word glyph “wife,” which is not like the others (“son”), can be deselected by clicking on it, after which the remaining word glyphs can be labeled as “son” with a single key stroke. This is the power of Intelligent Indexing.

5.1 Example of gridlines in word glyphs.

5.2 Example of descenders highlighted in red.

B.1 Example page from the 1920 Utah census.

B.2 Census page with ground truth data overlaid over fields.

B.3 Document 00166, the first document in the basic indexing timed evaluation.

B.4 Document 00167, the second document in the basic indexing timed evaluation.

B.5 Document 00168, the third document in the basic indexing timed evaluation.

B.6 Document 00213, an example of the documents used in the Intelligent Indexing timed evaluation.

B.7 Document 00426, an example of the documents used in the Intelligent Indexing timed evaluation.

B.8 Document 00786, an example of the documents used in the Intelligent Indexing timed evaluation.
List of Tables

3.1 Capturing the intelligence of humans and the computational power of computers. . 12
3.2 Data categories from the 1920 Utah census used for testing Intelligent Indexing. . 12

4.1 Ages of volunteers. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 35
4.2 How recently volunteers had indexed. . . . . . . . . . . . . . . . . . . . . . . . . 36
4.3 Lifetime hours indexed. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 36
List of Algorithms

3.1 Preprocessing ......................................................... 14
3.2 Building all cost matrices in the document collection ............... 16
3.3 Build Cost Matrix ...................................................... 17
3.4 Choosing candidate word glyphs (cwgs) ........................... 20
3.5 Interactive field labeling ............................................ 21
3.6 Calculating training set matches (Transitive Labeling) ............ 25
Chapter 1

Introduction

The problem we have approached in this thesis is to reduce the tedium and increase the efficiency of indexing and transcription in handwriting recognition and field discrimination through interactive training and incremental learning. The application domain is structured documents (like Census records) where sufficient repetition exists to justify the use of a learning system to semi-automate the process. In this context, the indexer performs manual annotation (training), selects (deselects) matching (nonmatching) candidate word glyphs through visual consensus and group labeling, adjusting threshold(s) interactively, as needed.

FamilySearch Indexing [1] is an ambitious crowdsourcing project where volunteers index historical records one field at a time. They record information such as name, age, gender, marital status, and place of birth (see Figure 1.1). These indexes allow genealogists to find information about ancestors contained in many types of documents with simple queries. FamilySearch’s vast and growing collection of indexed records is possible only through the combined labor of the tens of thousands of volunteers who give many hours of time to the effort.

Repetitive tasks, like indexing, are strong candidates for automation. Thousands of man hours could be saved, and more work accomplished, if indexing could somehow be automated. Unfortunately, there are many image processing challenges to overcome in an end-to-end automated indexing system, and one of the most challenging is automated handwriting recognition. Recognizing handwriting automatically has been studied for many years, and yet continues to be impracticable except in constrained circumstances.
Despite the challenges, if we don’t learn from user input, improvements in the efficiency of computerized indexing will be incremental at best, asymptoting in spite of improved user interfaces. Adding learning and assisted automated labeling can potentially provide significant improvement in throughput. However, our community is still trying to discover what that learning looks like.

Despite the challenges, some work has already been done in leveraging handwriting recognition to increase the rate of document indexing. However, these methods tend to silo the automated recognition and the manual recognition into two separated processes. Some fields are automatically labeled and the rest are manually indexed. Rather than take this approach, we present a semi-automated learning system which uses both the strengths of human indexers and handwritten word recognition technology. We call our learning system Green ICR and the indexing platform built on it we call “Intelligent Indexing”. “Green”, because of the conservation of time for the indexer, and ICR for Intelligent Character Recognition. ICR is a term used to distinguish automated handwritten word recognition from machine printed word recognition, which is called OCR.

There are many types of historical documents, ranging from structured tabular documents (Figure 1.1), to free form text (Figure 1.2(a)). Other documents, like Figure 1.2(b), have fields that
constrain what type of information to expect based on their label. For example, there are fields for names, places, and ages. This thesis targets historical documents that contain structure such that the vocabulary is constrained for each field (e.g. Figures 1.1 and 1.2(b), and not Figure 1.2(a)).

Our research does not pertain only to indexing historical documents. While we are currently focusing on the issue of indexing certain types of historical documents, there are other types of labeling, including part-of-speech tagging or image categorization, where a semi-automated approach would likely do well.

(a) Unstructured text is not the target of this thesis.

(b) A marriage license is an example of a form that has a standardized template with blanks to be filled out.

Figure 1.2: Historical documents with differing structure.
Chapter 2

Related Work

Closer integration and interaction of humans and computers is currently a highly researched area in the machine learning and pattern recognition communities [16]. Recently developing fields of study in this vein include active learning, semi-supervised learning, incremental learning, and interactive learning. We present here how these research areas are related to and differ from the work in this thesis.

2.1 Work with Similar Objectives

Sibade et al. have published work on an automated system for indexing French census registers [29]. In their system, the entire corpus is processed up front and all handwriting samples are either labeled automatically, or sent to manual indexers. Our work is different in that we intend to adapt our learning models through incremental, interactive training as the indexer progresses through the corpus. We expect both greater efficiency and accuracy for data fields with a constrained vocabulary.

Recent papers have been published on semi-supervised labeling of historical weather reports and Lampung characters (an Indonesian script)[24]. The authors focus on handwritten characters and digits in these papers, and show promising results. Unlabeled instances are transformed into several different features spaces. In their case, the feature spaces were the raw images, a PCA reduced space, and a reduced space from a stacked auto encoder. Each feature space is clustered, after which the cluster centroids are labeled by the human indexer. This label is applied to each instance in the cluster. Because instances participate in a cluster for each of the different feature
spaces, they will have multiple labels applied. When these labels are not homogeneous, a majority vote decides which label to apply. This approach is different from our approach because it does not contain a notion of interactive learning, or adaptability to changing authors and content. We consider these aspects to be a critical step forward in the field of labeling.

2.2 Active Learning

Active learning[28] is represented by a set of algorithms that answers the question of choosing the best candidate to label from a set of unlabeled examples, so that a supervised learner can achieve the highest possible accuracy from the fewest number of labels. Active learning is for situations when labeling data is expensive and can minimize the number of instances that need to be labeled. It has even been shown to increase accuracy over using the entire dataset [27]. Active learning helps to focus manual effort on the instances that will most help machine learning models extrapolate to the rest of the dataset [25].

Active learning is related in spirit to this thesis because it involves a human and machine learner working together to label an unlabeled dataset, but we are not doing active learning. In our current system, the user has freedom to label fields in any order, corrections will be made on the fly, and we make no claim of a minimum number of training examples to label in maximizing accuracy. However, employing active learning techniques is something we would like to consider in future work.

2.3 Incremental Learning

Machine learning models that adapt as new training instances are encountered are called incremental learners. Much of the work on incremental learning targets infrequent changes to large models like MLP or SVM, and these do not update at interactive rates [23, 26, 31]. The work we present can be thought of as an incremental learner, though the incremental learning comes trivially because
we are using k-Nearest Neighbor (k-NN). Adding new instances to the training set automatically updates the model.

### 2.4 Interactive Learning

Though in a different domain, the key contribution in Intelligent Scissors [18] is an excellent metaphor for the contribution of our research. Though graph search methods had been in use for many years in image segmentation, the addition of a human providing simple guidance in Intelligent Scissors made human guided segmentation a reality as a real time tool. Likewise, though handwriting recognition has been studied for many years, by pairing it with real-time human guidance, an efficient and improved indexing system is the result of our research.

We also consider the face labeling portion of Google Picasa™ to be very similar in concept to our research. Picasa™ automatically detects faces in images, and presents the cropped face images to the user to be labeled. As labels are paired with faces, Picasa™ will try to match labeled faces with unlabeled faces and automatically propagate the label to the other images with the same face, inviting user feedback and correction in the process. Thus, with minimal effort, an entire image collection can be indexed by a single person.

More directly related to our research, Doug Kennard performed research to demonstrate interactive training of the word warping algorithm for document annotation [12].

Research has been done to provide a methodology for evaluating machine assisted annotation techniques [8]. The tool is employed to show that machine assistance can be used to increase both annotator speed and accuracy. The tool can also reveal the level of accuracy required by the machine assistance to produce these positive gains.

George Nagy is one of the foremost pioneers and proponents of interactive learning and green technology. He built the CAVIAR (Computer Assisted Visual Interactive Recognition) system, which is used for recognizing faces and flowers. CAVIAR has shown that interactive recognition is more than twice as fast as the unaided human, and yields an error rate ten times lower than state-of-the-art automated classifiers. Nagy states in his paper that “whether such an approach can
be equally effective in the domain of documents as it is for flowers and faces is unproven, and adapting CAVIAR to document analysis requires further research.” [20] Nagy later approaches interactive learning in documents and provides a good summary of existing methodologies [19]. Our research also addresses this subject.
Chapter 3

Intelligent Indexing

Intelligent Indexing is a new methodology for annotation tasks that establishes a framework for learning from human interactions and improving over time. While using Intelligent Indexing, users will spend less time on repetitive actions and energy-wasting context switching and more time on higher-level tasks that require human intelligence.

Intelligent Indexing is suitable in any situation where repetition in the data can be leveraged to provide automation and reduce the load of manual effort. The system looks for similar items in the dataset, and allows the user to label whole sets of items at a time instead of each individually. The user has input into the classification boundary so that an individual’s preference for precision versus recall can be satisfied. Should errors be present in the list of matched items, it is simple to fix these errors before they are applied. This removes the need for post correction, and does so with surprisingly little cost. The high-level coupling and interaction between the indexer (user) and computer (client) is summarized in Table 3.1 where the goal is to capture the intelligence and knowledge of the indexer while taking advantage of the computational power of the client.

This chapter covers all of the data preprocessing, interactive learning, and details about the indexing client that constitute the main contribution of this thesis. In this research, we have striven to follow the axiom of trying the simplest thing first, and only transitioning to a more complicated strategy when necessary. Although not all of the ideas presented in this chapter were immediately obvious to us at the onset, they are all relatively straightforward. It is encouraging that the ideas presented here, while simple, have been shown to be effective. It stands to reason that if such gains can be had from simple techniques, there is plenty of room to grow in this domain.
This section describes the data from the 1920 Utah census, used in Intelligent Indexing. The census is composed of a preamble that contains information about the census including the state, county, and enumerator (census taker). The rest of the document is a large table. The table has a heading describing each column, then room for fifty individual records. The columns on the census include name, relationship to head of household, gender, race, whether they can read or write, marital status, place of birth, father’s place of birth, mother’s place of birth, and occupation.

For Intelligent Indexing, we chose to focus on six of the possible columns of information. The data encountered in these columns is summarized in Table 3.2. Several example images of the 1920 census are included in Appendix B.

<table>
<thead>
<tr>
<th>Census Category</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relationship to Head of Household</td>
<td>Relationships like “Wife”, “Daughter”, “Son”, or “Brother”.</td>
</tr>
<tr>
<td>Gender</td>
<td>(M)ale, (F)emale</td>
</tr>
<tr>
<td>Marital Status</td>
<td>(M)arried, (S)ingle, (W)idowed, (D)ivorced</td>
</tr>
<tr>
<td>Place of Birth, Father’s Place of Birth, Mother’s Place of Birth</td>
<td>Place names like “Utah”, “Wales”, or “New York”</td>
</tr>
</tbody>
</table>

Table 3.2: Data categories from the 1920 Utah census used for testing Intelligent Indexing.

### 3.1.1 Image Preprocessing

Straight from the scanner, historical documents are not ready for handwriting recognition or Intelligent Indexing (see Figure 3.1). The gray background must first be removed, which can be nontrivial and can affect the quality of the binarized handwriting (line 3 of Algorithm 3.1, see Figure 3.2). A bilateral filter with a large width is applied to the image to find the background, which is...
then subtracted from the original image. The images are then rectified (Figure 3.3) so that individual cells can be cropped out (line 4 of Algorithm 3.1, see Figure 5.1). [6, 9] Cells are cropped using a manually constructed template (line 5 of Algorithm 3.1). [6]

![Image](image.png)

Figure 3.1: Raw 1920 Census Image.

![Image](image.png)

Figure 3.2: Preprocessing for background removal and image enhancement

To prepare the fields for handwriting recognition, a histogram of gradients method is used to determine handwriting slant, which is removed for each field with a horizontal shear transform (Alg 3.1, line 6). The fields are then binarized using a connected components based binarization.
method [11]. The fields are then stored in individual files and are ready to be compared to generate similarity scores. [6]

Algorithm 3.1 Preprocessing

1: Input:
   
   \( \text{Img} = \) Original digital scan
   \( \text{CellBoundaries} = \) Bounding boxes

2: Output:
   
   \( \text{ProcImg} = \) Rectified, background removed document
   \( \text{Crops} = \) Cropped fields used for building cost matrices
   \( \text{Metadata} = \) Document metadata contains bounding boxes matched with record IDs

3: \( \text{bkgRemoved} \leftarrow \text{RemoveBackground(Img)} \) (Fig. 3.2)
4: \( \text{ProcImg} \leftarrow \text{Rectify(bkgRemoved)} \) (Fig. 3.3)
5: \( \text{Crops} \leftarrow \text{CropFields(CellBoundaries,ProcImg)} \) (Fig. 5.1)
6: \( \text{RemoveHandwritingSlant(Crops)} \)
7: \( \text{Metadata} \leftarrow \text{BuildMetadata(CellBoundaries,ProcImg)} \) (Fig. 3.4)

3.1.2 Metadata

For each image in the document collection, a matching metadata file is created (Alg 3.1, line 7, see Figure 3.4 for a partial example). The metadata stored contains the field location on the page, the category of each field, the enumerator for each page, the ground truth label for each field as provided by FamilySearch Indexing (see Appendix B), the label provided by the indexer in the Intelligent Indexing client, whether the label was applied automatically, and the time it took to label the field.
3.1.3 Precomputing Morphing Costs

The similarity scores generated using handwriting recognition are the backbone of the automated labeling system. For our system, these similarity scores were precomputed to guarantee interactive speeds while the client is running. The census collection was split into chunks according to handwriting style. This was possible because the enumerator (the census collector) wrote their name on each page, and because the census had been indexed previously, with the enumerator field included in the index. The similarity scores for a particular category and a particular group are stored in a file we call a cost matrix.

Figure 3.5 is provided to help the reader visualize how the document collection was grouped. Six of the twenty-seven categories on the 1920 census were chosen to test Intelligent Indexing. These are “Birthplace”, “Marital Status”, “Relationship to Head of Household”, “Father’s Birthplace”, “Mother’s Birthplace”, and “Gender”. For each of these categories, forty-two groups were created, one for each enumerator. The groups were simply given the name of the enumerator that defined that group. Algorithm 3.2 shows how the document collection is broken down for computing each cost matrix. The function called on line 10 is provided in Algorithm 3.3.
Algorithm 3.2 Building all cost matrices in the document collection

1: **Input:**
   - EnumeratorPages = Map of enumerator to list of documents
   - Categories = List of categories

2: **Output:**
   - All cost matrices are created and written to file

3: **for all** name : EnumeratorPages **do**
4:     pages ← EnumeratorPages[name]
5: **for all** category : Categories **do**
6:     fields ← {}
7:         **for all** page : pages **do**
8:             fields ← fields + GetFields(page, category)
9:         end for
10:     cm ← BuildCostMatrix(fields)
11:     SaveToFile (cm, category, name)
12: **end for**
13: **end for**

The reasons for chunking the collection into smaller groups are twofold. First, because each field is compared to each other field in a category, there is an inherent $O(n^2)$ complexity in computing the similarity scores. Anything that can be done to reduce the size of $n$ will reduce the compute time. Even with computing cost matrices in groups, it still took around 240 computing hours with 12 threads to create all the cost matrices. Second, we have shown in previous work [6] that handwriting recognition accuracy drops significantly when comparing between different handwriting styles and out-of-vocabulary words. The main ramification of the decision to break the collection into groups is that learning in one group does not transfer to the next group.

Cost matrix files are stored with a header that lists the fields being compared, followed by the actual floating point comparison values. Algorithm 3.3 shows how this is done. Because comparisons are symmetric and the comparison of a field to itself isn’t interesting, the resulting matrix is upper triangular. This symmetry reduces the number of comparisons that need to be performed by a factor of two, and it also reduces the cost matrix file size by half. While it is logical to think about the matrix in two dimensions, it is stored internally as a 1-dimensional array into which we index to make sure the correct comparison is used. Figure 3.6 illustrates a very simple cost matrix.
Figure 3.5: Organization of Cost Matrices.

The first occurrence of “Head” in Figure 3.6 has a cost of 0 when compared with itself and a cost of 3.2 when compared with another instance of “Head.” But 3.2 is still much smaller than 8.0 (“Wife” compared with the first instance of “Head”) or 9.3 (“Wife” compared with the second instance of “Head”). Note that a high cost indicates a greater difference between the words (for example, “Daughter” compared with “Wife”, “Head” (1) or “Head” (2)). This also allows candidate word glyphs (see Section 3.3.1) to be previewed in order of increasing cost, putting the most likely matches at the top of the list. Precomputing these Word Morphing costs allows comparisons to be performed at interactive rates using a simple look-up that is performed during interactive cell labeling by the indexer.

**Algorithm 3.3 Build Cost Matrix**

1: **Input:** fields = List of fields  
2: **Output:** Cost Matrix  
3: datastream ← ""  
4: for all field : fields do  
5:     datastream+==field.id  
6: end for  
7: for i = 0..COUNT(fields) do  
8:     for all j = (i + 1)..COUNT(fields) do  
9:         datastream+==WORDMORPH(fields[i], fields[j])  
10: end for  
11: end for
Figure 3.6: Example cost matrix.

3.2 Interactive Field Labeling

Document indexing is essentially composed of many individual labeling events. This section covers the different moving parts at work during field labeling. The use of training sets is discussed, by which labeled fields in each of the different categories are used to help label future occurrences (see section 3.2.6). Also, thresholds exist for each category of data and can be tuned by the indexer. These thresholds govern how aggressive the recognition engine is. Thresholds are covered in section 3.2.5.

When the recognition engine recommends fields to the indexer, these are displayed in a preview window (see section 3.3.1). The indexer has the opportunity to deselect any fields that don’t match (see section 3.2.4), after which the indexer applies the label to all the fields at once (see Figure 3.7). This mechanism is important because it allows the indexer to make quick adjustments to the automated learning when it happens instead of post correcting later. This is one way in which the intelligence of the indexer is leveraged (see Table 3.1). The process of approving the fields in the preview window is called visual consensus and is covered in section 3.2.2. By these means, labels are applied to fields in groups.

Fields with labels are grouped by color according to their labels as shown in Figure 3.8. The column headers for the columns that should be indexed are highlighted in red so that indexers can easily tell which fields to index. When columns are completed, this column header indicator changes colors so that indexers can be sure that they have not missed a field (Figure 3.8). While visual
consensus is likely the most important feature for avoiding labeling errors, this field coloration scheme is meant to assist in maintaining accuracy as well.

(a) Indexer selects a field (red box) and types the label (“son”) in the input box to the side. Other fields with a similarity score below a threshold are presented to receive the same label.

(b) All the “son”s are labeled, and the next field is selected.

Figure 3.7: Before and after the indexer labels a field.

### 3.2.1 Recommending Fields for Automated Labeling

There are two opportunities for the recognition engine to recommend fields for automated labeling. The first is when a field is selected. When a field is selected, all unlabeled fields are compared to it to find those that match below the current threshold for that category, as shown in Algorithm 3.4. Those that do match below the threshold are presented to the indexer, who can remove any that don’t match before applying the label (see Algorithm 3.5). Once the label is applied to the currently selected field, all fields that remain in the preview window are labeled as well.

The second opportunity follows a labeling event. When a field is labeled, all remaining unlabeled fields are compared to each of the fields in the training set that have the same label as the field just labeled (see Algorithm 3.6). Any that match under the threshold are prompted to the indexer in a preview window. This window is unique in the indexing client in that it is the only time that the system interrupts the indexer’s flow and forces a response. This prompt provides a way to use the training set to help provide automation while still keeping the paradigm of working directly
Figure 3.8: Column header coloring indicates progress through the page (green means the column is done). Cell coloring helps the indexer quickly see which fields have been indexed. Also, each label is assigned a different color, so fields that should have the same label, but have a different color, can be quickly identified.

on the page. Also, because the label is already known for these training set matches, it saves the indexer having to type it in again for these fields.

Algorithm 3.4 Choosing candidate word glyphs (cwgs)

1: **Input:**
   
   \( f \) = active field
   
   \( ufs \) = List of unlabeled fields with the same category as \( f \)

2: **Output:** List of unlabeled fields to display to the user as cwgs

3: \( matchingFields \leftarrow \{\} \)

4: **for all** \( uf \) : \( ufs \) **do**

5: \( \text{if WORDMORPHCOST}(f, uf) < \text{THRESHOLD} \) **then**

6: \( matchingFields += uf \)

7: **end if**

8: **end for**

9: **return** \( matchingFields \)

3.2.2 Visual Consensus

When an indexer selects a field, the cost matrix is used to determine which fields on the page are similar to the currently selected field. The threshold for the selected field’s category determines how many fields are considered candidate matches. These candidate word glyphs (cwgs) are displayed to the indexer so that they can be validated on the spot. This interaction is described in Algorithm 3.5. The user interface presents the matching fields to the indexer (see Figure 3.7) so that they can
be confirmed (See Algorithm 3.5, line 2). Fields that don’t match are removed by either clicking on them directly or by reducing the threshold (lines 5-7). Finally, all of the labeled fields are added to the training set (line 10).

After the interaction described in Algorithm 3.5, the selected field is passed as input into Algorithm 3.6, where potential matches to the training set for the selected field’s category are calculated and presented for transitive labeling. These are validated by the user in a similar manner as the matches to the selected field, with the added advantage that the label does not need to be re-keyed since it is already known.

**Algorithm 3.5 Interactive field labeling**

1: Indexer **selects** a cell, \( C \), to label (Ex. cell containing handwriting “Son”) - Figure 3.7(a)
2: Computer **presents** candidate word glyphs, \( cwgs \) - Algorithm 3.4
3: Indexer **keys** in a label, \( L \) for \( C \) (Ex. Indexer types “Son” in input box) - Figure 3.7(a)
4: **while** \( \exists f \in cwgs \text{ s.t. } \text{Label}(f) \neq L \) **do**
5: **user deselects** \( f \in cwgs \) that do not match \( C \) by
6: **a. clicking on** \( f \in cwgs \) that do not match (see Figure 3.9) and/or
7: **b. adjusting threshold slider** - Figure 3.11
8: **end while**
9: all \( cwgs \) are assigned label \( L \) - Figure 3.7(b)
10: all \( cwgs \) are added to training set

![Figure 3.9](image1.png)  
(a) Two fields are provided by the recommendation engine.  
(b) The indexer clicks on the one that is out of place.

**Figure 3.9:** Before and after the indexer deselects a recommendation.
3.2.3 Changing a labeled field

When a labeled field is corrected, it is either because the indexer changed their mind about a field, mistyped, or failed to prevent an automatic labeling from applying a label erroneously. In any of these cases, it is possible that other labeled fields in the column have the same error, so we learn from this interaction by considering all labeled fields and comparing them to the corrected field. Any fields under the current threshold are prompted as candidates for also having their label changed.

3.2.4 Removing a field from the preview window

In earlier iterations of Intelligent Indexing, there was no preview window. After a field was labeled, all fields considered matches to that field were automatically labeled, and it was up to the indexer to figure out what mistakes had been made. This was tedious, time-consuming, and frustrated the indexer. The preview window was a critical revelation, as it allows the indexer to proactively remove mistakes before they happen.

To remove a field from the preview window, the indexer either clicks on it (see Figure 3.9), or adjusts the threshold until the field is no longer considered a match. The fields in the preview window are sorted by match strength, so the fields most likely to be mismatched are clumped at the end of the list. This makes it easy to either click out all the mistakes, or see how the threshold is affecting the list.

Currently, clicking on a field simply removes it from the match list. However, as described in Future Work, we see opportunity to leverage this information in other ways as well.

3.2.5 Thresholds

A proper setting of the threshold is crucial for Intelligent Indexing to save time for the indexer. A threshold set too low means little or no automated labeling occurs. A threshold set too high results in many spurious fields showing up in the preview window, which will tend to frustrate the indexer. Both of these situations can be seen in the graph provided in Figure 3.10. For this graph, accuracy is calculated using a leave-one-out strategy. This is a cross-validation technique where the training
set and test set are split so that only one instance is in the test set, and the rest are in the training set. This is repeated so that each instance participates as the test set. The accuracy is the percentage of the instances that are labeled correctly. Labeling was done using nearest neighbor.

For low thresholds, the accuracy (blue line) is very high, but there are also few matches below the threshold. As the threshold increases, the percentage of fields with at least one match below the threshold grows and approaches 100% (red line). However, the accuracy (blue line) also continues to decrease. The accuracy will plateau when the threshold is such that all fields have a nearest neighbor below the threshold.

In our user interface, the indexer is given a slider that can be used to adjust the threshold to suit their preference. There is also a keyboard shortcut for adjusting the slider. As is mentioned in Table 3.1, setting this threshold to suit preference is a task well suited for the indexer. The effect of changing the threshold is immediate, meaning that if a field is selected and the slider is adjusted, the matching fields are reevaluated on the spot. This interaction is shown in Figure 3.11.

Figure 3.10: Graph depicting both accuracy (blue line) and “percent of fields with at least one match below the threshold” (red line) using different thresholds. This was calculated across all fields (approx. 1500) in the Father’s Place of Birth category for a particular enumerator.
(a) Indexer selects a field (red box). All fields with a similarity score below a threshold are presented to receive the same label.

(b) Increasing the threshold increases the number of matching word glyphs in the preview window, allowing more cells to be labeled.

Figure 3.11: Before and after the indexer adjusts the threshold.

### 3.2.6 Transitive learning

Not only can the neighbors of a selected field be labeled automatically, but potentially the neighbors of those neighbors as well. This transitive learning could get out of hand, that is, with potentially too much learning from one field. In practice one transitive step often yields a few extra labels without too many mistakes. A visualization of how this transitive labeling works is provided in Figure 3.12. In 3.12(a), the indexer selects a field (marked in green). Matches to that field (purple word glyphs) are then presented to the indexer (3.12(b)). Both green and blue word glyphs are added to the training set. After the indexer approves these fields and labels them, the neighbors of the training set (pink word glyphs) are confirmed as well (3.12(c)). These neighbors are calculated using Algorithm 3.6. This is illustrated in Figure 3.13 where the pink “Utah’s” have already been labeled and included in the training set. The other 10 “Utah’s” (bracketed within the dark box) are then previewed to the indexer as candidates for the same label with the prompt “Are the following also Utah? (Y/N).” In this case, since all of the candidate word glyphs are also “Utah,” the indexer would type “Y” on the keyboard. If there were any that were not “Utah” the indexer would deselect those and/or adjust the threshold. Note that the indexer does not need to type “Utah” on the keyboard since the label is already known.
This kind of “transitive learning” leverages and propagates the original label onto as many word glyphs as possible without having to re-key the label, thereby increasing efficiency and minimizing keyboard errors.

**Algorithm 3.6 Calculating training set matches (Transitive Labeling)**

1. **Input:**
   - \( f \) = last indexed field
   - \( ts \) = Training set corresponding to the category of \( f \)
   - \( ufs \) = List of unlabeled fields
2. **Output:** List of unlabeled fields to display to the user as matching candidates (see Figure 3.13)
3. \( matchingFields \leftarrow \{\} \)
4. **for all** \( uf : ufs \) **do**
5.   **for all** \( tsField : ts \) **do**
6.     **if** \( tsField.Label \neq f.Label \) **then**
7.     **continue**
8.     **end if**
9.     **if** \( \text{WORDMORPHCOST}(tsField,uf) < \text{THRESHOLD} \) **then**
10.    \( matchingFields +\= uf \)
11. **end if**
12. **end for**
13. **end for**
14. **return** \( matchingFields \)

### 3.2.7 Smart recommendations

The order in which fields are labeled matters. Starting at the top and working down the columns is one possible order, but it isn’t necessarily the ideal path. We built into the client a simple option that allows the indexing client to choose which field is labeled next. This option was disabled by default, but could be enabled through a menu option. The reason it was disabled by default is because the jumping back and forth throughout the column could be disorienting, and was potentially not worth the labelings it saved. The value added by this functionality remains to be tested.

Equation 3.1 is used to choose the next field to index. In this equation, \( T = \{t_1, t_2, t_3, \ldots, t_n\} \) represents fields in the training set, and \( U = \{u_1, u_2, u_3, \ldots, u_m\} \) represents the unlabeled fields on the page. The function \( \text{COST}(x,y) \) takes two fields as input and returns their morph cost. The result, \( r \), is the next field the indexer should label.
(a) Indexer selects a field (green).

(b) The selected field’s neighbors are prompted to the indexer (blue).

(c) After the indexer applies the label to the green (and blue) fields, the neighbors of the neighbors of the selected field are prompted to the indexer (red).

Figure 3.12: Interactive, transitive learning from an incremental training set.
Figure 3.13: Matches to the training set are a separate prompt that occurs after a labeling event.

We chose to use \textit{argmax} in this equation, because this forces the indexer to label word glyphs that are most different from one another. The idea is that by doing so, the system can more quickly learn the decision boundaries and how the threshold should be set.

\[ r = \arg\max_{u \in U} \left[ \min_{t \in T} \left( \text{COST} (u, t) \right) \right] \]  \hspace{1cm} (3.1)

3.3 Indexing Client

This section covers aspects of the user interface that we felt were an important part of the interactive learning. The section then discusses the different steps we took to minimize the amount of context switching required of the indexer. We felt in building the indexing client that a poor user interface could undo any benefit afforded by the intelligence of the system, and that context switches specifically should be avoided if at all possible.

3.3.1 User Interface

If a picture is worth a thousand words, a video is worth a million words. To get a better feel for the look and feel of the user interface, or to get a better idea of what is discussed in this section, see \url{https://www.youtube.com/watch?v=qgqVzEPnBEw} (also linked in abstract).
Candidate fields for automatic labeling are presented to the indexer beside the field to be labeled (see Figures 3.7, 3.13, and part D of Figure 3.14). Grouping potential matches and juxtaposing them next to the field being labeled turned out to be a very effective system for validation and consensus. It appears to be an innate human ability to look at a group of things and quickly pick out which are different from the others. We call it the “Sesame Street principle”, based on the common learning technique employed on Sesame Street to choose the item out of place in a small group [10, 30]. “Which one of these is not like the others?”

**Autocomplete**

For most categories, the same words come up again and again. Instead of requiring the user to type these words in over and over, auto-complete will finish what the user is typing as long as it is a label that has been previously seen. Moreover, with transitive labeling, the word does not even have to be retyped, as the label is already known.

**Coloring fields**

We wanted some way to quickly spot-check that fields have been labeled correctly. We do this by coloring the fields according to their label. For example, in the "Relationship to Head of Household" category, all fields labeled “head” are the same color, and all fields labeled “son” are another color (as in Figures 3.8 and 3.14). We selected a small set of light pastel colors that would not compete with the underlying handwriting. These are sufficient for the columns with small vocabularies. After these colors are exhausted, a new random color is chosen for each label. These random colors are chosen to ensure that they are also bright enough to not conflict with the underlying handwriting.

**Mouse over highlight**

Fields are outlined in red under two conditions. The first is when a field is selected. Under this condition, the field is given a bold, solid red outline that helps it to stand out. The second condition
is when the mouse hovers over a field that is selectable. This outline is also red, but is dashed and thinner. Outlining fields as they are moused over is a subtle UI element that reduces confusion about which fields need to be indexed on the page.

**Auto advancing selected field**

When the indexer finishes labeling a field, Intelligent Indexing automatically advances the active field to the next unlabeled one in the column. If the column has been completed, it advances the active field to the next unlabeled field in the next column. The goal of the auto advancing feature is to allow the indexer to keep their hands on the keyboard and not have to use the mouse or arrow keys. Auto advancing the active field saves hundreds of mouse clicks or key presses for each image indexed.

**Auto page re-positioning**

Making the indexer scroll the page is a waste of time when it can be prevented. The display should automatically adjust itself so that everything that needs to be displayed can be displayed. Since the active field is automatically advanced, this will sometimes move it off of the displayed portion of the image. The active field is kept on screen by automatically scrolling the image.

**3.3.2 Minimizing Context Switching**

**Interacting directly with image**

FamilySearch Indexing current displays the document image and below it a form or spreadsheet for entering the labels (see Figure 1.1). The problem with this approach is that it separates spatially the label from its matching field. This makes it difficult for the indexer to be sure that the right information ends up in the right cell. Also, the entry form takes up valuable screen real estate. The result is that the indexer is looking at the page with tunnel vision. Worse, the indexer has to context switch, moving their gaze or focus of attention, each and every time a label is entered, making it easy to get off track.
Rather than take this approach, we have the user interact directly with the image. To select a field to label, the indexer simply clicks on it (in Figure 3.14 part B, the user has clicked on the “Utah” field). A red rectangle marks the place on the page where the indexer is currently working, not unlike a cursor in word processing software. There is also the question of how to indicate to the user what data on the page should be indexed. With the spreadsheet indexing layout, the fields that should be indexed are inherent to the design of the spreadsheet. In our system, however, column headers are colored to specify which columns should be indexed, and only fields that need to be indexed are selectable (see Figure 3.8). A red column header indicates more work is necessary, while a green column header indicates that the column is done.

**Working down columns**

On a census page, records of an individual are presented across rows. It is perhaps natural then to advance the selected field across rows, allowing the indexer to record the information for one individual at a time. However, the items of information about a person, including the relationship to the head of household, the gender, the marital status, and the birth place, are dissimilar. Horizontal movement across categories causes the indexer to have to bring back to mind the different possibilities and rules associated with that category.

To minimize context switching between categories, we advance the field down the column. In this way, the indexer enters all of the fields in one category first, then moves on to the next category. In the end, this amounts to an assertion that the context of the column is more important than the context of the individual’s record. As can be see in Figure 3.14, the indexer is working down the “Birth Place” column.

**Label entry adjacent to field**

Originally we had the label editor on the side bar of the application. However, this was a violation of our “minimize context switching” principle. By placing the label editor adjacent to the field, it is
now much easier to see both the field and entered text (see part B of Figure 3.14). We believe this is not only faster and easier for the user, but will reduce the number of mistakes that are made.

**Preview matches are localized**

Deciding where to put the auto-labeling preview was difficult. On the one hand, overlaying fields on top of the document image risks being too confusing for the indexer. On the other hand, we did not want to break our “minimize context switching” principle and place the preview on the sidebar. In the end, we chose to allow the indexer to keep their focus locally, and to take measures to ensure the preview stands out from the page. An example of how this looks in the software can be seen in part D of Figure 3.14, where the candidate matches to the currently selected “Utah” field are displayed below the label entry form.

**Labels displayed adjacent to fields**

Even while using colors to link fields with the same label, indexers will still want to be able to see exactly what they typed to correct any possible mistakes. Displaying these labels all the time for all fields would get distracting, so we display them only for the category of the field currently selected (blue and green machine-printed text spelling “Utah” and “England”). The labels do obscure part of the image, but we consider this to be an acceptable consequence, since it obscures only a part of the image that is not germane to the current task (for an example, see part E of Figure 3.14). We color the machine-printed text blue for fields the indexer has labeled, and green for fields that have been automatically indexed.

### 3.4 Summary

Intelligent Indexing is a trainable system for field labeling that improves over time. It works on tabular document images like the 1920 Census. These document images are preprocessed and linked with metadata. The differences between individual field images are precomputed and stored in cost matrices to be used while indexing. Entering labels for fields is an interactive experience. A
Figure 3.14: Intelligent Indexing user interface. Notice that everything the indexer needs for the current task is localized, so minimal context switching is required. A) Column headers are red for columns that need to be indexed, and turn green when those columns are completed. B) The active field has a solid red outline. C) The label entry form is placed directly beside the active field. D) Candidate word glyphs from the recommendation engine are presented in a preview window. E) Labels for the fields already indexed are shown. Blue means they have were indexed manually, green for automatically. F) Labeled fields are colored according to their label.

The recommendation engine uses the precomputed cost matrices to determine matches to the selected field, which are displayed to the indexer in a preview window for visual consensus. If there is a field out of place in the preview window, the indexer is able to easily notice this by the Sesame Street principle and deselects it. A threshold exists for each category and is used to determine how many matches are considered. Fields are colored to indicate clusters and provide a visual mechanism for catching mistakes. Column headers are colored to indicate where indexers should work, and change color when a column has been completed.
Chapter 4

Results

The chapter begins with a brief description of the experiment design, followed by the results of the experiment. The full table of results is provided in Appendix C.

4.1 Experiment Design

We chose to perform a user study to validate our thesis supposition, because the user is at the heart of Intelligent Indexing. The study was a comparison between two indexing systems. The first is a row-by-row, manual indexing system that represents the control, and the second is Intelligent Indexing. The basic indexing system had the same user interface, it simply did not have the recommendation system, and it indexed across rows (in an attempt to preserve the same behavior found in FamilySearch Indexing). The comparison between these two systems was in terms of both accuracy and time required to index. We also queried the volunteers for a qualitative comparison between the two indexing modes.

The documents used for the experiment came from the 1920 Utah census. We used six of the columns on the page for evaluation: relationship to head of household, gender, marital status, and the place of the birth columns for the individual as well as the mother and father.

We built an experiment program that provides volunteers a few sample pages to index and that was instrumented to collect data about their experience. The experiment program progressed through a series of states. It first collected data about the indexer through a brief survey. The survey asked for information about the participant’s prior indexing experience. The next state allowed the
volunteer to practice with both the basic and Intelligent Indexing modes without being timed. This practice time was meant for the volunteer to learn the user interface and ask questions about how it worked before proceeding to the timed section. After the volunteer was satisfied in their ability to use both indexing modes, they then moved on to the next experiment state, which was to index three pages using the basic indexing mode. These pages were the same for every indexer. After this state, the final state was to index three pages using Intelligent Indexing. These pages were randomized for each indexer, but were always collectively from the same enumerator. During these final two states, the time the indexing cursor spent on each field was measured, as well as the label entered for each field. When the volunteer ended each state, the timing and label information was sent back to the server.

When choosing the experiment design, we had to make decisions with compromises. If we chose to have each volunteer index the same pages, it would limit the strength of the results with questions of whether the pages chosen were representative of the document collection. Choosing to have each volunteer index different pages is also problematic. When results are inconsistent, it is difficult to assign a cause to explain the variation. These issues diminish as the sample size increases. We could have had volunteers index both the same pages and a randomized set, but this would have further burdened volunteers of whom much had already been asked. In the end, the fact that we pulled randomly from the census for Intelligent Indexing, paired with the results presented in this chapter, provides strong evidence that Intelligent Indexing not only works in idealized situations, but in a more generalized manner.

4.1.1 Indexing Server

A server was built to facilitate the capture of timing and label information for each indexing session, as well as to provide batches to the indexers. This was a matter of convenience more than of necessity. The alternative would have been to manually assemble all the images with their associated metadata and cost matrices needed for the experiment and bundle them with the executable. Since the pages were randomized for the Intelligent Indexing portion of the experiment, a different bundle
would have to be generated for each participant. The participant’s results would have to be stored to their local machine as they worked, and then returned through some transfer method like email. Instead, we decided to build an indexing server to automate these steps and provide scalability to the process of generating results.

The indexing client was augmented to communicate with the server. Field label and timing information was stored in an XML format at the document level. When returned from an indexer’s machine, these XML files were stored in a flat directory, and a separate file was used as an index to link the volunteer’s survey responses with the corresponding XML files.

4.2 Experimental Results

The indexing client was deployed on Mac, Windows, and Linux. Volunteers had to simply download the program, follow instructions, do the indexing, and the rest was taken care of. Images, metadata, and cost matrices were sent to the local machine via the indexing server, and the results were sent back to the server and stored. Results were collected over a six week period. Volunteers were solicited at conference sessions, poster sessions, through local community groups, from among family and friends, through FamilySearch Indexing connections, and finally from poster advertisements in stairwells.

The results for ten participants were collected, each of which completed three pages of basic indexing and three pages of Intelligent Indexing. The demographics of the participants are shown in Tables 4.1, 4.2, and 4.3.

<table>
<thead>
<tr>
<th>Ages</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-25</td>
<td>6</td>
</tr>
<tr>
<td>26-50</td>
<td>2</td>
</tr>
<tr>
<td>51-75</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 4.1: Ages of volunteers.
How Recently Indexed | Count
---|---
within the last week | 1
within the last month | 4
within the last year | 3
longer than a year or N/A | 2

Table 4.2: How recently volunteers had indexed.

| Lifetime Hours Indexed | Count |
---|---|
less than 1 hour | 1
1 to 5 hours | 2
6 to 20 hours | 4
more than 20 hours | 3

Table 4.3: Lifetime hours indexed.

4.3 Quantitative Results

4.3.1 Indexing Time

The indexing times for individual fields were aggregated to produce an indexing time per page. For the ten indexers who completed both batches, Intelligent Indexing on average decreased the indexing time by an impressive 2 minutes (117 seconds) per page. The average completion time per page across all participants was 7.5 minutes (450.7) seconds for basic indexing, and 5.77 minutes (346.1) seconds for Intelligent Indexing, which is a 30.22% reduction in time. This decrease in the time it took to index using Intelligent Indexing was statistically significant ($p < 0.05$). For all but one experiment, Intelligent Indexing was faster than the basic indexing.

Histograms of the indexing times for basic indexing and Intelligent Indexing are shown in Figure 4.1. As can be seen from these graphs (and as the statistics attest), the distributions do appear to belong to different populations.

FamilySearch Indexing (FSI) also kept metrics for the time it took to complete 1920 Utah census pages using their indexing system. However, because FSI indexed several more columns than we did for the test and because the data is at the page level, and not at the field or column level, the times are not really comparable. That being said, the average indexing time per page (within the 25th and 75th quartiles) was 25.93 minutes (1555.69 seconds).
4.3.2 Indexing Accuracy

Through comparison to ground truth data provided by FamilySearch (see Appendix B for more information about the ground truth data), the volunteers were found to be 98.85% accurate for basic indexing on average. For Intelligent Indexing, the accuracy for manually indexed fields was 96.58%, and the accuracy for automatically indexed fields was 98.46%, with an aggregated accuracy of 98.12% accurate for Intelligent Indexing. The difference in accuracy between Intelligent Indexing and basic indexing was not measured to be significant (p<0.05). A complete table with the results is available in C.

Errors came in a couple different forms. A common mistake was to misspell the more difficult state and country names. Another mistake was typing a single letter instead of the full phrase (in cases other than “Gender” or “Marital Status”). Presumably, this occurred at transitions between columns where the indexer expected the auto complete function to work. Another small set of “mistake”s were ones in which the entered label was semantically equivalent to the ground
truth, but the two were regarded as different because of, for example, abbreviations. To see all of the different mistakes made by indexers, see Figures C.2 and C.8.

### 4.3.3 Indexing Automation

Volunteers using Intelligent Indexing had 59.18% of the fields filled out automatically on average per page. This automation rate ranged from 44.33% to 76.33%. In Figure 4.2 a scatterplot shows the correlation between the time to index a page using Intelligent Indexing and the percent of fields automatically indexed (Automation rate). The $R^2$ value for the scatterplot is 0.45, showing a strong correlation between automation rate and indexing time. This correlation shows that the difference between the times to index using the basic indexing system and Intelligent Indexing may be attributed to more than just the switching of row-based indexing to column-based indexing. It suggests that a higher automation rate produces faster indexing times.

![Figure 4.2: Scatterplot showing correlation between automation rate and time to index a page. Calculated for all pages indexed using Intelligent Indexing, with outliers removed.](image-url)

$$R^2 = 0.4525$$
4.4 Qualitative Results

Some volunteers noted that the pages chosen to test Intelligent Indexing were more difficult to index than those chosen to test the basic indexing. The basic indexing pages were hand chosen, while the pages for Intelligent Indexing were pulled randomly from the census. Random pages were chosen so that the results were not dependent on the particular image. The logic at the time for making the basic indexing pages the same was so that the different volunteers could be directly compared. Without making a formal comparison, we are left to speculation. However, it seems likely that the Intelligent Indexing results are negatively biased by the fact that more difficult pages were used to test it.

A couple volunteers responded to the questions asked in the instructions (see Appendix A). Some of these quotes are provided in Appendix C. The critical responses to Intelligent Indexing usually had to do with minor bugs or issues with the user interface. Also, some felt that the experiment needed more instructions. For positive responses, a common sentiment was that column-based indexing was much better than row-based indexing. Also, those who figured out how to use the adjustable threshold found it to be effective. Interestingly, not everyone used the threshold in the same way. Some adjusted the threshold as they went and used the adjuster to remove mismatches from the recommendations. Others set a very generous threshold, allowing for several mistakes but capturing more of the correct matches as well. In this case, the work flow was to click to remove the mismatches from the recommendations window.

4.5 Visual Results

This section is composed of screen captures that present the preview window under a variety of conditions. In Figure 4.3, we demonstrate the value added by sorting the items in the preview window. In Figures 4.4, 4.5, and 4.6, we see the effects of a degraded page on accuracy.

In Figure 4.7, we highlight the fact that for a given word in a category, the indexer does not know how many times it appears on the page, and so it isn’t always obvious whether turning up the
threshold will yield any benefit. Figure 4.8 shows an example where a high degree of automation is possible. Finally, in Figure 4.9, we show how easy it is to pick out word glyphs that don’t belong in the preview window.

Figure 4.3: Mismatched fields tend to cluster in the back because the list is sorted by match strength. This is quickly remedied by adjusting the threshold slider to remove the “son”s.

4.6 Summary and Conclusion

We have shown through a user study that Intelligent Indexing is faster than a basic indexing client. The level of automation (percentage of fields automatically labeled) achieved by the indexer is correlated with the speed with which they complete the index. Accuracy results were comparable between the two systems, with the highest degree of accuracy belonging to fields that were automatically indexed.

The look and feel of the program was successful. For the most part, label entry was very well streamlined and the indexer only occasionally needed to use the mouse. Volunteers expressed
preference for Intelligent Indexing, with the caveat that it took a little bit longer to get used to. Volunteers stated that Intelligent Indexing was less mentally fatiguing and was more engaging. It became a game to see how well the threshold could be adjusted so that the most automation occurred without having to correct too many mistakes.
Figure 4.5: Even the gender category can be difficult with degraded handwriting.
Figure 4.6: When there is a large amount of repetition, the preview window is still helpful even in conditions where recognition is difficult.

Figure 4.7: It isn’t always clear why a mismatch appears in the preview window. In this case, “England” and “Denmark” are about the same length, and both begin and end with “tall” letters. It could be that all the “Denmark”s on the page are being shown, and simply that the threshold is set too high.
Figure 4.8: The small vocabulary, single character fields worked out very well in general.
Figure 4.9: The Sesame Street principle (see Section 3.3.1) is well demonstrated in this example. In this case the word glyph “wife,” which is not like the others (“son”), can be deselected by clicking on it, after which the remaining word glyphs can be labeled as “son” with a single key stroke. This is the power of Intelligent Indexing.
Chapter 5

Future Work

As we delved into the heart of this research, we soon realized that there was much more to be done than we had time for. We have chosen to capture as many of our ideas as possible in this chapter, while understanding that in doing so it has become rather long. Our intention is that this chapter be a working document for those who may wish to continue our research in this topic, and also for those who may develop these ideas in a more complete indexing pipeline.

5.1 Difficult Categories

In our evaluation, we chose a subset of the columns on the 1920 census to test our system. The columns we chose had a limited vocabulary, and thus had a large amount of repetition. The repetition is where we get the most benefit from Intelligent Indexing. We did not explore ways of semi-automating the indexing of fields with large vocabularies, although such work would be very beneficial. Two categories in particular that deserve added individual attention are names and ages.

Names are difficult both because of the large variety of possible names, and because the full name sometimes appears in the same cell. Really, the surname and given names would be best handled separately, so a way of cropping out individual names is necessary. When families are indexed on census records, the last name is sometimes abbreviated in some way instead of being written out for each family member. Being able to automatically recognize and populate the last name, whether or not it is abbreviated, is one area that could be further explored.
For the “age” category in the 1920 census, the number was either in years or a fraction of a year. There are enough different ages that there is little repetition to harness for only a couple of pages. However, if the digits of the age could be cropped as separate components, they could be individually recognized and combined. If this could be done, the category would only have ten possible values, ignoring the ages that are a fraction of a year. In some cases, the context of the page could be brought to bear in calculating an \textit{a priori} probability that the age is in some range. For example, when the census is organized by families and has a category for relationship to head of household, it can be inferred that the ages of the children are less than the ages of the parents with high confidence (though it is not guaranteed, as the children may not be biologically related to the parents).

### 5.2 Between-Category Dependencies

Often it is possible to infer information about one category from information in another category. As mentioned above, information about the possible age range of a person can be inferred by their familial relationships. Some other inter-column dependencies on the 1920 census exist between “relationship to head of household”, “gender”, and “marital status”. While the particular information gathered for any given document may vary, the principle of leveraging relationships between the different categories of information can always be applied.

In a project unrelated to Intelligent Indexing, we did some proof of concept work to improve recognition accuracy for the “relationship to head of household”, “gender”, and “marital status” categories. Using a maximum entropy model and the similarity scores from word morphing, we were able to get a modest increase in accuracy (85% to 87%) in the number of records where all three of these cells were correctly classified. Other work has also been done in a similar vein [22].

Exploiting inter-column dependencies can also be used as a validation method. While indexing hundreds of records as quickly as possible, it is not uncommon for an indexer to make a mistake. If the combination of labels for a record is calculated to be very unlikely, the offending field can be flagged immediately for the indexer to confirm. Sometimes the error is actually on the
page. For example, on one document in the 1920 census, a person who was a wife and married had an ‘M’ in the gender column. This person was almost certainly a woman, so the automated system could flag this field for careful attention and correction in the index.

5.3 Learning the Threshold

In the Intelligent Indexing application, the threshold is manually adjusted by the indexer to suit their preference. We made some effort to adjust the threshold automatically, but did not find our efforts to be helpful in practice. However, it does seem reasonable to expect the automated system to be able to determine the ideal setting of the threshold to maximize the number of fields previewed while minimizing errors. What isn’t immediately clear is the definition of “ideal” in this context. It could be that the indexer is best served by never having to see a mistaken classification, or it could be that the cost of more frequent errors is more than offset by the increase in the number of fields that can be automatically labeled.

Another possibility is that the nature of learning and labeling changes over time in a session. Perhaps it is good to not label too aggressively at first. Propagating a label could be a function of the number of fields already assigned that label. Should the threshold be set aggressively enough that it will occasionally make a mistake, which would allow it to quickly discover decision boundaries and also allow the indexer to feel more like he is training the system? In other words, is it okay, or even better, to get one wrong?

5.4 Optimal Ordering of Labeling

A fascinating question to pursue is how to optimally order the cells in a category to minimize the time it takes to index. It can be hard to get some intuition about why the order matters. By way of example, suppose one field that is “daughter” is close in similarity score to two other “daughter” fields. Now imagine there is another “daughter” field that is closer not only to those same two
“daughter” fields, but to an additional three “daughter” fields. It is clearly better to index the second one first, as it will require at least one fewer labeling event.

Another issue related to the order of labeling are two conflicting philosophies. The first is that the indexer should label each word in a category until there are no more, and then transition to the next word. For example, label all the “Utah”s first, and then start on the other place names. The second philosophy is to successively label the most disparate unknown examples on the page, which would could be used to quickly hone in on the optimal threshold. Both of these ideas appear to have merit, but it isn’t clear which is superior in the context of a user interface.

5.5 Better Preprocessing

The 1920 Utah census, like many other historical documents, has grid lines that define the cells to be filled in. These grid lines can interfere with handwriting recognition. We did a very simple crop of the cell at the same place on each page, relying on the alignment of the pages to ensure that each form element was in the same place on the page. With this simple method, many grid lines were left in the cropped handwriting images that were used to determine similarity scores (see Figure 5.1). These grid lines undoubtedly affected the accuracy. Previous research has shown that the presence of grid lines can reduce word morphing accuracy significantly [6]. Future time and effort could be applied to better preprocessing of these images to minimize the instances where grid lines appear in the cropped handwriting images. This could be done with line detection, inpainting, or possibly with a more aggressive crop that may lose some of the handwriting but preserve enough to still be distinguishing.

![Figure 5.1: Example of gridlines in word glyphs.](image)

Another possible preprocessing step to add would be to remove ascenders and descenders from the word glyphs [3]. Like grid lines, ascenders and descenders can significantly affect word
morphing scores. For example, we would want to remove the descenders (red segments) from the cell containing “head” so that the word warping is not influenced by these extraneous strokes (Figure 5.2).

![Figure 5.2: Example of descenders highlighted in red.](image)

We use the rectified, background removed images to present to indexers in the indexing client. It would be possible to use the original scans, and backwards map all of the bounding boxes and mouse clicks. Some documents have been encountered by Intelligent Indexing volunteers where the handwriting is extremely faint. It could be that in these cases, the original document would be easier to read from.

### 5.6 Alternative Uses for the Training Set

We choose the known example from which to draw the label for an unlabeled example simply by whichever is closest in similarity score. This appears to work pretty well, but it would be interesting to see what improvements, if any, are possible by using 3-NN or some other voting scheme. This new scheme would be used for the training set matches calculation, while the matches to the currently selected field would stay the same. Another tempting use of the training set is to build clusters using hierarchical agglomerative clustering and for each unlabeled field calculate the distance to each “cluster center”. The distance between clusters could be defined as some metric, like averaging, of the distances from each field in one cluster to each field in the other.


5.6.1 Deselection Knowledge

If a candidate field for labeling is removed from the list of matches, either as a match to the current selection or a match to the training set, we have learned two things. First, we have learned that the removed field does NOT have the currently selected field’s label. In cases where the number of possibilities for the column is severely constrained, like gender or marital status, the actual label can almost be inferred from knowing what the label is not. However, in general this information is not immediately useful, except that the field can be removed from match lists for the erroneous label. But it can also be marked as not having a particular label, commencing a process of labeling by elimination. Second, we have learned that the removed field is very similar in terms of morph cost to other fields that should have a different label. Conceptually, this information could be used to help differentiate between two sets of fields that cluster close to each other. However, currently we simply use a global threshold and a simple, unweighted nearest neighbor approach for deciding matches, so we are unable to fully exploit this information at this time. A simpler use of the information is to mark the removed field so that it cannot participate in automatic labeling, and must be labeled manually. Since it is more like different fields than its own, this can prevent future mislabelings as well.

5.6.2 Using Characteristics of Labeled Data

As the indexer proceeds through the page, and onto other pages, the number of labeled examples in each category increases. As the size of these training sets increases, it becomes reasonable to calculate statistics about the data. For example, the number of fields assigned each of the possible labels in the category could be heavily skewed. This information could translate into \textit{a priori} probabilities that an unlabeled field is in a given category, and could be used to preview, initialize, or label fields in subsequent pages.
5.6.3 Weighted Classification

For a more sophisticated classification system, a weighted distance metric could be used. By this method all the instances in the training set could vote and participate in the final classification. This type of system is typically more robust against outliers and noise. Weights could be based simply on the word morph cost. Individual instances could also be multiplied or devalued, where the devaluation is learned during the interactive labeling. For example, if a labeled example in the training set causes a mismatch in the preview window, that training instance could be devalued as it participates in future labeling events.

5.7 Machine Consensus

There are several other handwriting recognition algorithms that could be used together to improve recognition accuracy. In Lund’s work regarding OCR output, it is shown that in an ensembled approach, even a classifier with relatively low accuracy can contribute and improve the overall result. [17] Further, it may be the case that one algorithm handles a certain handwriting type, or a certain document collection better than another. This could be learned, and then the best tool for the job could be employed.

5.8 Automatic Collection Clustering

In general the documents in a collection may not have a convenient label that can be used to group similar handwriting together. However, we think such grouping may be possible automatically using unsupervised clustering. Doug Kennard, author of the word morphing algorithm, has done work to show that this may be possible. [14] As more and more data is produced from a variety of sources, a corpus of training sets could be developed where each training set contains the fields in a category that match closest with another and furthest from other handwriting. For a novel document collection, the interactive learning could be bootstrapped by the existing training set that
most closely matches the data. We could then diminish the old training data and gradually replace it with the new training data.

5.9 Alternative Handwriting Recognition

Using domain constraints (“son,” “daughter,” etc.) and simpler word-spotting or ordered feature-matching for candidate glyph selection may be sufficient compared to the more computationally expensive word warping. For example, research could be done into using HOG or SIFT features left to right for word glyph discrimination.

5.10 Collaborative Framework

Sometimes the most time-consuming aspect of indexing for a volunteer is when they are trying to index a word or name with which they are not familiar. To help the volunteer in such a situation, a collaborative framework could be built with visual consensus across multiple asynchronous indexers, each with local domain expertise. An indexer could flag an unfamiliar word glyph, triggering an alert in the global community where anyone could help the volunteer decide what the label should be.

5.11 Automatic Category Discovery

Many documents have machine printed text whether in a table header or in a form that denotes what information is to be expected. For example, the header might say “Name,” or “Place of Birth.” If these could be recognized automatically, it would save the time spent preparing the document collection for annotation. This could potentially be done using a machine print glyph recognition system [6]. It could also be used to pre-generate potential labels and save keystrokes.
5.12 Minimum Count for Training Set Matches

Some volunteer indexers found that when only one or two labels were presented in the training set matches window, the gains afforded was not worth the cost of being interrupted in the flow of indexing.
Chapter 6

Conclusion

Intelligent Indexing provides a general, scalable, collaborative approach to indexing and transcription of non-machine-readable documents that exploits visual consensus and group labeling while harnessing human recognition and domain expertise. One of the main contributions of this thesis is the idea of previewing and selecting candidate matches while maintaining the context of the page. Results show that Intelligent Indexing reduces significantly (see Section 4.3.1) the time required to index census records, while maintaining comparable accuracy. While the software developed demonstrates “proof of concept” for Intelligent Indexing, we propose this as more of a strategy than a specific solution — an illustration of how the indexer and software can be tightly coupled in a cooperative, symbiotic framework, training each other and harnessing the best of both. More work would be required to fully integrate these ideas into an industrial setting. Finally, we have discussed possible improvements to Intelligent Indexing that can further enhance recommendation accuracy, make more use of available information, adapt the system to data categories with larger vocabularies, allow Intelligent Indexing to be used on a variety of devices, and improve the user experience.
Appendix A

Experiment Instructions

The following email was sent to Intelligent Indexing volunteers.

Thank you for volunteering to participate as a tester for Intelligent Indexing. I anticipate your participation will take approximately 45 minutes. All you have to do is run the program and follow the instructions. You will need to have an internet connection while you are running the program. After you have done the test, please write me an email letting me know you are done and briefly answer the following questions:

What did you like about Intelligent Indexing?
Which indexing style was more enjoyable (Basic or Intelligent)?
What could be done to improve the Intelligent Indexing experience?

You can download the program for your operating system of choice from the following links.
You should just have to uncompress the folder and it will be ready to go.

LINUX: https://dl.dropboxusercontent.com/u/8327936/IntelligentIndexing.tar.gz To run the program, use the shell script
MAC: https://dl.dropboxusercontent.com/u/8327936/IntelligentIndexing.dmg
WINDOWS: https://dl.dropboxusercontent.com/u/8327936/IntelligentIndexing.zip

If you have any questions, or issues downloading or running the program, let me know. You can email me or call me at ...

Thank you for being willing to be one of our select group of early testers. Since it is still in early testing, please don’t share this email or the download links with others.
Appendix B

Ground Truth

Ground truth data was provided by FamilySearch Indexing. The data was produced by volunteers using the FamilySearch Indexing program. Each field was indexed by two independent indexers, and the discrepancies between these two indexes was arbitrated by a third volunteer. The data was provided in a single xml file for the census. This file was parsed to connect each individual label with its corresponding field. While we made no effort to verify the accuracy of the ground truth, our experience has been that the quality of the data is very good.

For the “Gender” category and “Marital Status” categories, the census is abbreviated but the ground truth was not. Because there may not have been consistency in how the pages were indexed, we truncated everything after the first letter in both the ground truth and the indexed results before making comparisons between the two. For example, if one indexer transcribed gender as “Male” and the other as “M,” we compared only the first letters (“M”). Additionally, we reduced all characters to lower case before comparing to the ground truth. For example, we want “Male” and “male” to be equivalent.

Figure B.1 is an example page from the 1920 census, and the ground truth for this page is shown in B.2.

B.1 Document Examples
Figure B.1: Example page from the 1920 Utah census.
Figure B.2: Census page with ground truth data overlaid over fields.
Figure B.3: Document 00166, the first document in the basic indexing timed evaluation.
Figure B.4: Document 00167, the second document in the basic indexing timed evaluation.
Figure B.5: Document 00168, the third document in the basic indexing timed evaluation.
Figure B.6: Document 00213, an example of the documents used in the Intelligent Indexing timed evaluation.
Figure B.7: Document 00426, an example of the documents used in the Intelligent Indexing timed evaluation.
Figure B.8: Document 00786, an example of the documents used in the Intelligent Indexing timed evaluation.
Appendix C

Results Data

C.1 Qualitative Data

Many of the participants specifically said that they preferred column-based indexing over row-based indexing. There was a higher learning curve for Intelligent Indexing, but in spite of the learning curve there was no participant who preferred the basic indexing system. Some respondents said that Intelligent Indexing was less fatiguing. Here are a couple extended quotes made about the system by volunteers.

“This was awesome! I feel like it saved me TONS of time! I really like how it went down in columns, instead of across the rows like the other indexing. I feel like that, in and of itself, was easier to understand and made it faster! I was a little confused at first at how it worked, but I figured it out. I liked how it gave you options instead of just automatically filling it, and you having to go back and fix some things. I liked how it was easy to use and understand. I also liked the y/n option, that was neat.”

“Intelligent [Indexing], it was a bit harder to grasp at first, but the overall experience was much more enjoyable and not as monotonous. The basic way became very difficult because it was continually putting in the same things. The intelligent was more of a game to see how well the computer could get the same images.”
C.2 Quantitative Data

This section contains the table of volunteer results (see Figure C.1) and a chart that shows all of the mistakes made in both the Intelligent Indexing results and basic indexing results (Figures C.2 and C.8).

In the table of results, Automation (%) is calculated as the number of fields automatically indexed divided by the total number of fields indexed. The Time Gained (sec) column is computed as the sum of the 3 Intelligent Indexing page times subtracted from the sum of the 3 basic indexing page times.
<table>
<thead>
<tr>
<th>Participant</th>
<th>Basic Indexing</th>
<th></th>
<th>Intelligent Indexing</th>
<th></th>
<th></th>
<th>Time Gained (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Time (sec)</td>
<td>Accuracy (%)</td>
<td>Time (sec)</td>
<td>Accuracy (%)</td>
<td>Automation (%)</td>
<td></td>
</tr>
<tr>
<td>participant 0</td>
<td>303.35</td>
<td>100</td>
<td>492.04</td>
<td>99</td>
<td>61</td>
<td>-108.11</td>
</tr>
<tr>
<td></td>
<td>344.54</td>
<td>99</td>
<td>256.63</td>
<td>99</td>
<td>71.33</td>
<td></td>
</tr>
<tr>
<td></td>
<td>300.35</td>
<td>99</td>
<td>296.8</td>
<td>100</td>
<td>65</td>
<td></td>
</tr>
<tr>
<td>participant 1</td>
<td>284.87</td>
<td>100</td>
<td>282.41</td>
<td>100</td>
<td>62.67</td>
<td>20.25</td>
</tr>
<tr>
<td></td>
<td>310.42</td>
<td>100</td>
<td>282.41</td>
<td>100</td>
<td>62.67</td>
<td></td>
</tr>
<tr>
<td>participant 2</td>
<td>1108.73</td>
<td>100</td>
<td>715.53</td>
<td>99.67</td>
<td>62.33</td>
<td>1193.65</td>
</tr>
<tr>
<td></td>
<td>1027.4</td>
<td>100</td>
<td>595.92</td>
<td>99.67</td>
<td>65.33</td>
<td></td>
</tr>
<tr>
<td></td>
<td>747.69</td>
<td>98.33</td>
<td>377.72</td>
<td>100</td>
<td>76.33</td>
<td></td>
</tr>
<tr>
<td>participant 3</td>
<td>390.21</td>
<td>99.67</td>
<td>261.58</td>
<td>99.67</td>
<td>60</td>
<td>455.13</td>
</tr>
<tr>
<td></td>
<td>467.92</td>
<td>100</td>
<td>320.22</td>
<td>91.67</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td></td>
<td>358.3</td>
<td>99.67</td>
<td>175.5</td>
<td>99.67</td>
<td>70</td>
<td></td>
</tr>
<tr>
<td>participant 4</td>
<td>496.11</td>
<td>99.33</td>
<td>278.46</td>
<td>98.33</td>
<td>62.67</td>
<td>526.11</td>
</tr>
<tr>
<td></td>
<td>484.49</td>
<td>99.33</td>
<td>339.53</td>
<td>99.33</td>
<td>56</td>
<td></td>
</tr>
<tr>
<td></td>
<td>413.95</td>
<td>98.33</td>
<td>250.45</td>
<td>98.67</td>
<td>62.67</td>
<td></td>
</tr>
<tr>
<td>participant 5</td>
<td>463.34</td>
<td>95.33</td>
<td>443.56</td>
<td>98.33</td>
<td>50.33</td>
<td>34.94</td>
</tr>
<tr>
<td></td>
<td>455.92</td>
<td>99</td>
<td>448.31</td>
<td>97.67</td>
<td>46</td>
<td></td>
</tr>
<tr>
<td></td>
<td>392.52</td>
<td>99</td>
<td>384.97</td>
<td>95.33</td>
<td>60.33</td>
<td></td>
</tr>
<tr>
<td>participant 6</td>
<td>394.84</td>
<td>100</td>
<td>233.06</td>
<td>99.67</td>
<td>55</td>
<td>331.35</td>
</tr>
<tr>
<td></td>
<td>354.69</td>
<td>100</td>
<td>239.04</td>
<td>97.33</td>
<td>55.67</td>
<td></td>
</tr>
<tr>
<td></td>
<td>295.02</td>
<td>98.33</td>
<td>242.1</td>
<td>98.2</td>
<td>56.31</td>
<td></td>
</tr>
<tr>
<td>participant 7</td>
<td>630.27</td>
<td>100</td>
<td>474.5</td>
<td>98.67</td>
<td>50</td>
<td>349.54</td>
</tr>
<tr>
<td></td>
<td>532.91</td>
<td>99.32</td>
<td>419.25</td>
<td>98</td>
<td>44.67</td>
<td></td>
</tr>
<tr>
<td></td>
<td>461.58</td>
<td>98.98</td>
<td>381.47</td>
<td>99.33</td>
<td>44.33</td>
<td></td>
</tr>
<tr>
<td>participant 8</td>
<td>373.56</td>
<td>97.33</td>
<td>302.69</td>
<td>92</td>
<td>54</td>
<td>335.66</td>
</tr>
<tr>
<td></td>
<td>424.16</td>
<td>97.33</td>
<td>295.2</td>
<td>96</td>
<td>57.67</td>
<td></td>
</tr>
<tr>
<td></td>
<td>345.03</td>
<td>94</td>
<td>205.2</td>
<td>98</td>
<td>64.33</td>
<td></td>
</tr>
<tr>
<td>participant 9</td>
<td>263.04</td>
<td>99.33</td>
<td>314.44</td>
<td>94</td>
<td>66.67</td>
<td>377.29</td>
</tr>
<tr>
<td></td>
<td>433.94</td>
<td>98</td>
<td>229.51</td>
<td>99.67</td>
<td>60.33</td>
<td></td>
</tr>
<tr>
<td></td>
<td>370.59</td>
<td>97.33</td>
<td>146.33</td>
<td>100</td>
<td>72.67</td>
<td></td>
</tr>
<tr>
<td>AVERAGE</td>
<td>450.018</td>
<td>98.853666667</td>
<td>332.824333333</td>
<td>98.118333333</td>
<td>59.176666667</td>
<td>351.581</td>
</tr>
</tbody>
</table>

Figure C.1: Table of results.
## Intelligent Indexing

### Manually Indexed

<table>
<thead>
<tr>
<th>RELATIONSHIP TO HEAD</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Daughter</td>
<td>daugher: 11</td>
</tr>
<tr>
<td>Grandmother Grandma</td>
<td>grandmother: 1</td>
</tr>
<tr>
<td>Niece</td>
<td>neice: 2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SEX CODE</th>
<th></th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>MARITAL STATUS</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Divorced</td>
<td>w: 1</td>
</tr>
<tr>
<td>Single</td>
<td>-: 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>BIRTH PLACE</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Bohemia</td>
<td>bolivia: 1</td>
</tr>
<tr>
<td>Canada</td>
<td>england: 3</td>
</tr>
<tr>
<td></td>
<td>utah: 1</td>
</tr>
<tr>
<td>Connecticut</td>
<td>connectict: 1</td>
</tr>
<tr>
<td>Idaho</td>
<td>idah': 1</td>
</tr>
<tr>
<td>Japan</td>
<td>japam: 2</td>
</tr>
</tbody>
</table>

Figure C.2: Intelligent Indexing errors, page 1.
<table>
<thead>
<tr>
<th>Korea Oc</th>
<th>korea: 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syria</td>
<td>seria: 1</td>
</tr>
<tr>
<td><strong>FTHR BIRTH PLACE</strong></td>
<td></td>
</tr>
<tr>
<td>Alabama</td>
<td>oklahoma: 1</td>
</tr>
<tr>
<td>Bohemia</td>
<td>bolivia: 1</td>
</tr>
<tr>
<td>Connecticut</td>
<td>conneticut: 1</td>
</tr>
<tr>
<td>Denmark</td>
<td>sweden: 2</td>
</tr>
<tr>
<td>England</td>
<td>isle of man: 1</td>
</tr>
<tr>
<td>Illinois</td>
<td>kansaa: 1</td>
</tr>
<tr>
<td>Ireland</td>
<td>denmark: 1</td>
</tr>
<tr>
<td>Isle Of Man</td>
<td>lake of: 1</td>
</tr>
<tr>
<td>Korea Oc</td>
<td>korea: 2</td>
</tr>
<tr>
<td>Louisiana</td>
<td>louisana: 1</td>
</tr>
<tr>
<td>Nebraska</td>
<td>nebraksa: 1</td>
</tr>
<tr>
<td>New Hampshire</td>
<td>new hampshire: 1</td>
</tr>
</tbody>
</table>

Figure C.3: Intelligent Indexing errors, page 2.
<table>
<thead>
<tr>
<th>Country</th>
<th>Errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scotland</td>
<td>scoland: 1</td>
</tr>
<tr>
<td>Sweden</td>
<td>denmark: 1</td>
</tr>
<tr>
<td>Syria</td>
<td>seria: 1</td>
</tr>
<tr>
<td>Tennessee</td>
<td>tenn: 1</td>
</tr>
<tr>
<td>United States</td>
<td>at sea: 1</td>
</tr>
<tr>
<td></td>
<td>un us: 1</td>
</tr>
<tr>
<td></td>
<td>utah: 2</td>
</tr>
<tr>
<td></td>
<td>w. s.: 1</td>
</tr>
<tr>
<td>Utah</td>
<td>u: 6</td>
</tr>
<tr>
<td></td>
<td>uth: 1</td>
</tr>
<tr>
<td></td>
<td>wales: 1</td>
</tr>
<tr>
<td>MTHR BIRTH PLACE</td>
<td></td>
</tr>
<tr>
<td>Bohemia</td>
<td>bolevia: 1</td>
</tr>
<tr>
<td>Colorado</td>
<td>sweden: 3</td>
</tr>
<tr>
<td>Connecticut</td>
<td>connicut: 1</td>
</tr>
<tr>
<td>Denmark</td>
<td>sweden: 1</td>
</tr>
<tr>
<td>England</td>
<td>engalnd: 1</td>
</tr>
<tr>
<td>Idaho</td>
<td></td>
</tr>
</tbody>
</table>

Figure C.4: Intelligent Indexing errors, page 3.
<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Indiana</td>
<td>canada: 1</td>
</tr>
<tr>
<td></td>
<td>england: 1</td>
</tr>
<tr>
<td></td>
<td>utah: 2</td>
</tr>
<tr>
<td>Ireland</td>
<td>sweden: 1</td>
</tr>
<tr>
<td>Korea Oc</td>
<td>korea: 2</td>
</tr>
<tr>
<td>Rhode Island</td>
<td>providence: 1</td>
</tr>
<tr>
<td>Scotland</td>
<td>scotland: 1</td>
</tr>
<tr>
<td>Sweden</td>
<td>denmark: 2</td>
</tr>
<tr>
<td>Syria</td>
<td>seria: 1</td>
</tr>
<tr>
<td>United States</td>
<td>un us: 1</td>
</tr>
<tr>
<td></td>
<td>utah: 1</td>
</tr>
<tr>
<td>Utah</td>
<td>u: 1</td>
</tr>
<tr>
<td></td>
<td>uah: 1</td>
</tr>
<tr>
<td></td>
<td>united states: 3</td>
</tr>
<tr>
<td></td>
<td>uta: 3</td>
</tr>
<tr>
<td>Wyoming</td>
<td>wyoming?: 1</td>
</tr>
</tbody>
</table>

**Automatically Indexed**

**RELATIONSHIP TO HEAD**

Figure C.5: Intelligent Indexing errors, page 4.
| Head          | daughter: 1  
|              | son: 1       
| Niece        | niece: 1     
| Son          | wife: 3      
| SEX CODE     |              
| Female       | m: 6         
| MARITAL STATUS |            
| Divorced     | m: 1         
| Single       | m: 2         
| Widow        | m: 1         
| Widowwed     | m: 1         
| BIRTH PLACE  |              
| Idaho        | utah: 2      
| FTHR BIRTH PLACE |         
| Iowa         | utah: 1      
| Nebraska     | nebraksa: 1  
| Utah         | u: 19        

Figure C.6: Intelligent Indexing errors, page 5.
<table>
<thead>
<tr>
<th>MTHR BIRTH PLACE</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Wales</td>
<td>u: 1</td>
</tr>
<tr>
<td></td>
<td>utah: 1</td>
</tr>
<tr>
<td>Colorado</td>
<td>sweden: 2</td>
</tr>
<tr>
<td>Idaho</td>
<td>utah: 2</td>
</tr>
<tr>
<td>Sweden</td>
<td>denmark: 2</td>
</tr>
<tr>
<td>Utah</td>
<td>u: 2</td>
</tr>
<tr>
<td></td>
<td>united states: 2</td>
</tr>
<tr>
<td></td>
<td>uta: 14</td>
</tr>
<tr>
<td>Wales</td>
<td>utah: 1</td>
</tr>
</tbody>
</table>

Figure C.7: Intelligent Indexing errors, page 6.
## Basic Indexing

<table>
<thead>
<tr>
<th>RELATIONSHIP TO HEAD</th>
<th>SEX CODE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adopted Daughter</td>
<td>Male</td>
</tr>
<tr>
<td>ad daughter: 2</td>
<td>f: 11</td>
</tr>
<tr>
<td>ad. daughter: 4</td>
<td></td>
</tr>
<tr>
<td>daughter: 2</td>
<td>head 1: 1</td>
</tr>
<tr>
<td>granddaughter: 1</td>
<td></td>
</tr>
<tr>
<td>Adopted Son</td>
<td></td>
</tr>
<tr>
<td>ad son: 2</td>
<td></td>
</tr>
<tr>
<td>ad. son: 4</td>
<td></td>
</tr>
<tr>
<td>grandson: 1</td>
<td></td>
</tr>
<tr>
<td>son: 2</td>
<td></td>
</tr>
<tr>
<td>Brother-in-law</td>
<td></td>
</tr>
<tr>
<td>dont know: 1</td>
<td></td>
</tr>
<tr>
<td>Daughter</td>
<td></td>
</tr>
<tr>
<td>father: 1</td>
<td></td>
</tr>
<tr>
<td>Hired Man</td>
<td></td>
</tr>
<tr>
<td>head: 1</td>
<td></td>
</tr>
<tr>
<td>Sister</td>
<td></td>
</tr>
<tr>
<td>son: 1</td>
<td></td>
</tr>
<tr>
<td>Wife</td>
<td></td>
</tr>
<tr>
<td>wie: 1</td>
<td></td>
</tr>
</tbody>
</table>

Figure C.8: Basic indexing errors, page 1.
<table>
<thead>
<tr>
<th>MARITAL STATUS</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Married</td>
<td>≤:1</td>
</tr>
<tr>
<td></td>
<td>nm:1</td>
</tr>
<tr>
<td></td>
<td>s:1</td>
</tr>
<tr>
<td>Single</td>
<td>d:1</td>
</tr>
<tr>
<td></td>
<td>m:1</td>
</tr>
<tr>
<td>Widowed</td>
<td>m:8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>BIRTH PLACE</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Canada</td>
<td>can:1</td>
</tr>
<tr>
<td>Idaho</td>
<td>idao:1</td>
</tr>
<tr>
<td>Oklahoma</td>
<td>ohloahoma:1</td>
</tr>
<tr>
<td>Sweden</td>
<td>swden:1</td>
</tr>
<tr>
<td></td>
<td>sweeden:1</td>
</tr>
<tr>
<td>Switzerland</td>
<td>switerland:1</td>
</tr>
<tr>
<td>Utah</td>
<td>'utah:1</td>
</tr>
<tr>
<td></td>
<td>u:1</td>
</tr>
</tbody>
</table>

Figure C.9: Basic indexing errors, page 2.
<table>
<thead>
<tr>
<th>FTHR BIRTH PLACE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Canada</td>
</tr>
<tr>
<td>can: 1</td>
</tr>
<tr>
<td>Illinois</td>
</tr>
<tr>
<td>helinsia: 1</td>
</tr>
<tr>
<td>illionis: 1</td>
</tr>
<tr>
<td>illiosis: 1</td>
</tr>
<tr>
<td>Indiana</td>
</tr>
<tr>
<td>illionis: 1</td>
</tr>
<tr>
<td>louisiana: 1</td>
</tr>
<tr>
<td>Iowa</td>
</tr>
<tr>
<td>illinois: 1</td>
</tr>
<tr>
<td>Massachusetts</td>
</tr>
<tr>
<td>massachucetts: 1</td>
</tr>
<tr>
<td>massachucetts: 1</td>
</tr>
<tr>
<td>Michigan</td>
</tr>
<tr>
<td>michagan: 1</td>
</tr>
<tr>
<td>South Dakota</td>
</tr>
<tr>
<td>south dekoda: 1</td>
</tr>
<tr>
<td>Sweden</td>
</tr>
<tr>
<td>sweeden: 1</td>
</tr>
<tr>
<td>Switzerland</td>
</tr>
<tr>
<td>switerland: 3</td>
</tr>
<tr>
<td>Utah</td>
</tr>
<tr>
<td>u: 7</td>
</tr>
<tr>
<td>Wales</td>
</tr>
<tr>
<td>w: 1</td>
</tr>
</tbody>
</table>

Figure C.10: Basic indexing errors, page 3.
<table>
<thead>
<tr>
<th>MTHR BIRTH PLACE</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Iowa</td>
<td>sorba: 1</td>
</tr>
<tr>
<td>Michigan</td>
<td>mighigan: 1</td>
</tr>
<tr>
<td>New Zealand</td>
<td>new england: 1</td>
</tr>
<tr>
<td>Pennsylvania</td>
<td>pennselvania: 2</td>
</tr>
<tr>
<td>Sweden</td>
<td>sweden: 1</td>
</tr>
<tr>
<td>Switzerland</td>
<td>switerland: 8</td>
</tr>
<tr>
<td></td>
<td>swizerland: 1</td>
</tr>
<tr>
<td>Utah</td>
<td>idaho: 1</td>
</tr>
<tr>
<td></td>
<td>u: 7</td>
</tr>
<tr>
<td></td>
<td>wales: 1</td>
</tr>
</tbody>
</table>

Figure C.11: Basic indexing errors, page 4.
References


[10] Raposo J. Stone J. and Hart B. One of these things is not like the others. URL http://muppet.wikia.com/wiki/One_of_These_Things.


