5-2011

Information Extraction in an Optical Character Recognition Context

Ramon Pereda
University of Nevada, Las Vegas

INFORMATION EXTRACTION IN AN OPTICAL

CHARACTER RECOGNITION CONTEXT

by

Ramón Pereda

Bachelor of Science in Mathematics
Bachelor of Science in Computer Science
University of New Orleans
1991

Master of Science
University of Arizona
1993

Master of Science
National Technological University
1997

Master of Science
University of Texas
1999

A dissertation submitted in partial fulfillment of
the requirements for the

Doctor of Philosophy in Computer Science
School of Computer Science
Howard R. Hughes College of Engineering

Graduate College
University of Nevada, Las Vegas
May 2011

UNLV
UNIVERSITY OF NEVADA LAS VEGAS
THE GRADUATE COLLEGE

We recommend the dissertation prepared under our supervision by

Ramón Pereda

entitled

Information Extraction in an Optical Character Recognition Context

be accepted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Computer Science
College of Engineering

Kazem Taghva, Committee Chair

Thomas Nartker, Committee Member

Laxmi Gewali, Committee Member

Ajoy Datta, Committee Member

Ashok Singh, Graduate Faculty Representative

Ronald Smith, Ph. D., Vice President for Research and Graduate Studies
and Dean of the Graduate College

May 2011

ABSTRACT

Information Extraction in an Optical Character Recognition Context

by

Ramón Pereda

Dr. Kazem Taghva, Examination Committee Chair
Professor of Computer Science
University of Nevada, Las Vegas

In this dissertation, we investigate the effectiveness of information extraction in the presence of Optical Character Recognition (OCR). It is well known that the OCR errors have no effects on general retrieval tasks. This is mainly due to the redundancy of information in textual documents. Our work shows that information extraction task is significantly influenced by OCR errors. Intuitively, this is due to the fact that extraction algorithms rely on a small window of text surrounding the objects to be extracted.

We show that extraction methodologies based on the Hidden Markov Models are not robust enough to deal with extraction in this noisy environment. We also show that both precise shallow parsing and fuzzy shallow parsing can be used to increase the recall at the price of a significant drop in the precision.

Most of our experimental work deals with the extraction of dates of birth and extraction of postal addresses. Both of these specific extractions are part of general methods of identification of privacy information in textual documents. Privacy information is particularly important when large collections of documents are posted on the Internet.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

This dissertation presents an evaluation of information retrieval (IR) systems and information extraction (IE) systems in the presence of OCR errors. While the IR systems not designed with OCR errors in mind perform robustly, the IE systems do not. The IE systems need to take additional measures to counter-act the OCR errors and do so with some measured success. Chapter 2 gives an overview of IR and IE. The major technical steps in creating OCRed documents are explained in Chapter 3. Chapter 4 summarizes key research results of the Information Science Research Institute (ISRI) regarding the effects of OCR on unmodified IR systems. The methods and OCR effects of **Hidden Markov Models** are explained and experimental results described in chapters 5 and 6. In chapters 7 and 8, the methods and OCR effects using **Precise Shallow Parsing** are presented. Chapters 9 and 10 do the same as the previous two, but for **Fuzzy Shallow Parsing**. Chapter 11 gives a graphical summary of the IE results and draws some conclusions.

CHAPTER 2

INFORMATION RETRIEVAL AND EXTRACTION

Information Retrieval

Information Retrieval (IR) is predominately the science of searching for text, though sometimes it includes audio recordings, images, and video. The focus of this research is on text. IR has overlapping usage with data retrieval, document retrieval, and text retrieval. Baeza-Yates and Ribeiro-Neto (1999), claim “despite its maturity, until recently, IR was seen as a narrow area of interest mainly to librarians and information experts”. The Web brought IR to the forefront.

IR systems deal with natural language text which is not always well-formed and often semantically ambiguous. Small errors in an IR systems are likely to go unnoticed and be forgiven. The user of an IR system is trying to get information about a subject or topic. That intent is expressed in a query. The IR system must return documents that are relevant to the query. Often the list of potential documents is so large that an exhaustive review is not practical; so, the list of returned documents must be ranked in order of relevance.

Preprocessing on the documents is a central theme to most IR systems. A data structure called an index is typically built from the documents to speed up searching. If we wants to find all documents that contain a phrase or word, we could sequentially search all the documents but an index allows for a much quicker search. A process called stemming is used to reduce words to their grammatical roots. For example, “fishing”, “fished”, “fish”, and “fisher” all reduce to the root word “fish”. Extremely common words are not distinctive enough for search collections of any significant size. These common words are called stopwords. Examples are “a”, “the”, “should”, “would”, “but”, and “because”. These words are often removed from consideration during the search process. Other times, they might be allowed for full phrase searching as in “to be or not to be”. Identifying the key phrases in a document is far from a solved problem but central to many IR systems. These are but a few of the many document processing steps that make up a success IR systems.

Information Extraction

Information extraction (IE) structures the text within documents into semantically useful information. Human language is marvelously complex and so typical IE systems make incremental improvements towards understanding the information within documents. The simplest piece of information is an attribute-value pair, such as the **company-name** label and a particular company. An example attribute-value pair would be **company-name***(“Joe’s Bike Shop”).* Named-entity extraction is the technical term for identifying these phrases in text with a useful label for the instance. The next level of abstraction is a relation, such as ***wrote****(author, book).* **Wrote** is the name of the relation, and *author* and *book* are the fields. This is conceptually the same relation notion used in a relational database, though relations might be stored using a completely different type of database. Particular instances of a relation are sometimes thought of as facts. So, **wrote***(“Charles Darwin”, “Origin of Species”)* is a fact that might be extracted from a encyclopedia entry. Some information can be readily synthesized given a collection of facts. For example, given the attribute values for **key-sentence**, these could be ordered and used as a crude but useful summary of a document. The relation **friend**(*person1, person2*) could be used to generate a social graph of a group of people. There are many useful and sophisticated models for building systems from relations. However, the extracting of relation instances remains the hard problem of IE.

Documents are largely invisible to many information systems. Consider viewing online classified ads as a large collection of small documents. Reading 50 ads might take half an hour. The useful attributes and relations are buried inside of complex but not random text. Compare that task to the task of online shopping at Amazon.com or better yet Zappos.com. Shopping with a database is a much more efficient task. In seconds, you can find out if there is a size nine, extra wide, low-profile running shoe. Detailed searching inside of classified ads such as those in eBay or Craig’s List is not as scalable of a task.

Tim Berners-Lee, an inventor of the world-wide web, is leading a project called the Semantic Web (Bizer, Heath, & Berners-Lee, 2009). The Web started out a web of documents. These documents are intended for human consumption. The Web of Data is a growing part of the Web that has billions of assertions, or bits of data, linked together. These links are intended to be reliable and machine readable. The Semantic Web project is establishing best practices for publishing and connecting data on the web. IR systems, aka search engines, will adapt to the evolving web. IE systems will automatically sift out the data of documents, allowing the data and not just the document to be published.

CHAPTER 3

SCANNING, OCR, AND DIGITIZING

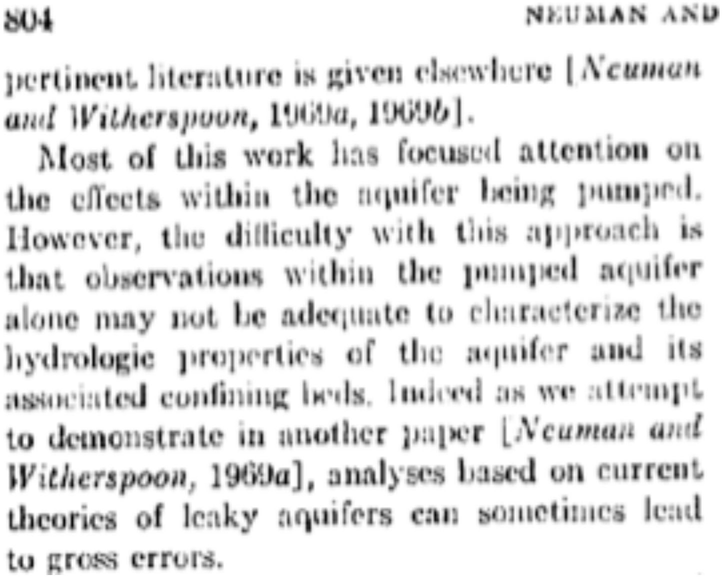
The process of converting a physical printed item into an electronic document is commonly referred to as *digitizing*. This process is more commonly done in reverse in the form of *printing* an electronic document. Conceptually, the digitizing can be divided into three stages: page *scanning*, block *zoning*, and text *recognition*.

Scanning means taking a special digital picture of a sheet of paper. This picture is taken with an optical scanner, which senses variations in light intensity used to construct a bit-map image. This image is often black and white with 1's for representing black pixels and 0's for representing white pixels.

Block zoning is the process of dividing up the pages into different logical areas (blocks), that are typically rectangular. These blocks must be ordered to preserve the anticipated reading order. As a simple example, if the page is formatted in two columns, the page must be divided into two blocks with the left one ordered before the right one for English text.

Text recognition is the process of converting text blocks into computer-encode characters, no different than if they had been hand-typed into an electronic document. Typically, the portion of the image identifying a single character is individually segmented and translated into its computer-encoded character equivalent. For state-of-the-art OCR systems, the accurate recognition of English Latin-script text from high quality scanning is comparable to what humans can do. The state-of-the-art has not advanced much in the last decade, but that art is also somewhat of a black art in that the best software applications are commercial and closed to outside technical inspection. The best open source solution for studying detailed OCR systems is perhaps Tesseract (2010), which was given to Google by ISRI.

Many errors that are generated by OCR systems can be traced back to low quality scanning or deteriorated printed materials. Sometimes that original physical paper is no longer available to re-do the scanning. Below are two examples of poor quality scans and one example of a reasonable scan.

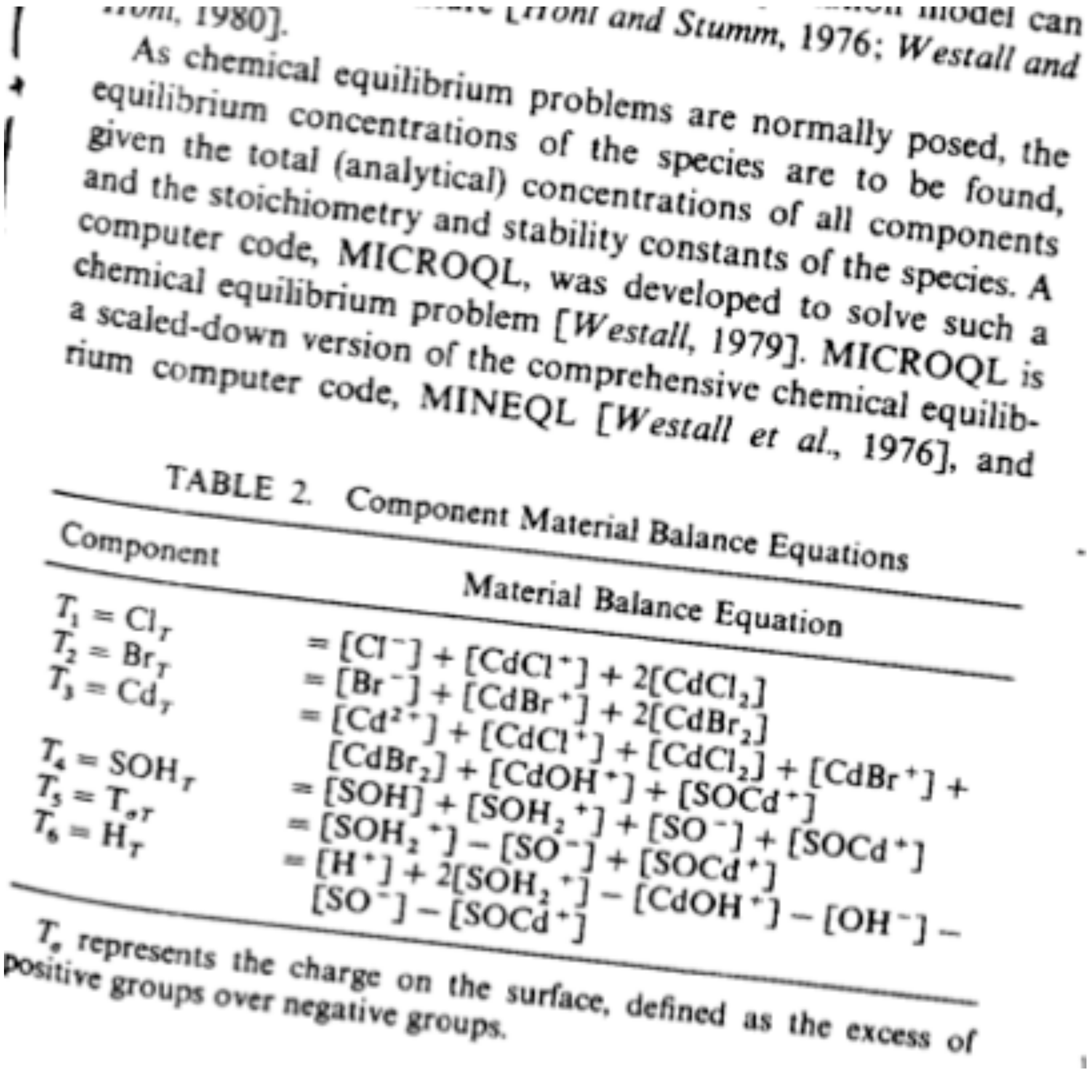


804 NEUMAN AND

pertinent literature is given elsewhere [*Neuman and Witherspoon,* 1969*a*, 1969*b*].

Most of this work has focused attention on the effects within the aquifer being pumped. However, the difficulty with this approach is that observations within the pumped aquifer alone may not be adequate to characterize the hydrologic properties of the aquifer and its associated confining beds. Indeed as we attempt to demonstrate in another paper [*Neuman and Witherspoon,* 1969*a*], analyses based on current theories of leaky aquifers can sometimes lead to gross errors.

Figure 3.1 Deteriorated printed page. Notice the broken characters.



model can

oni, 1980]. [Hohl and Stumm, 1976; Westall and

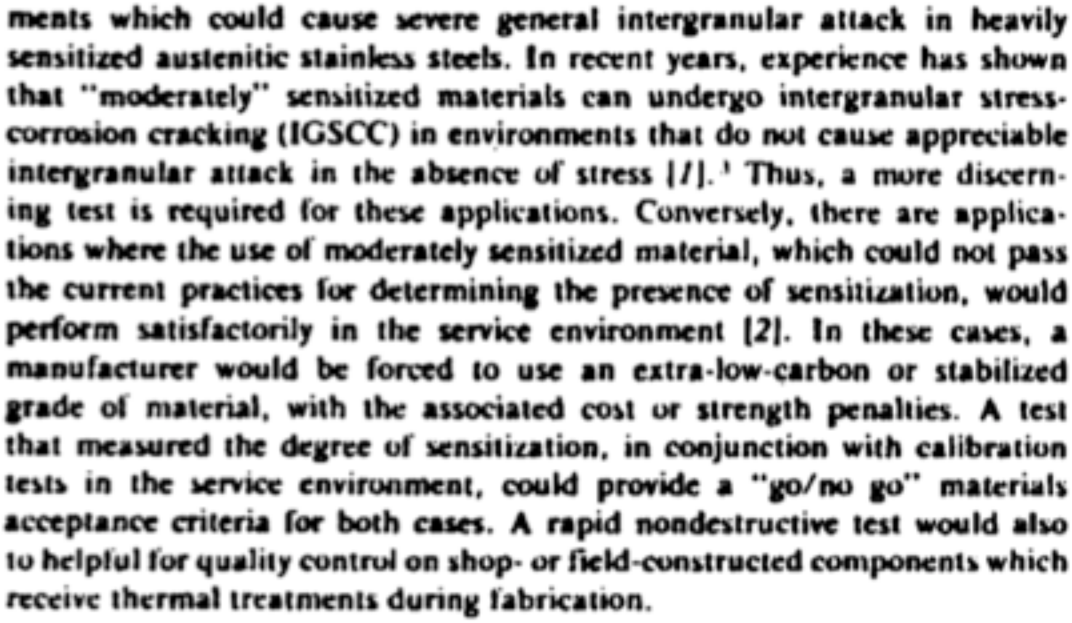
As chemical equilibrium problems are normally posed, the equilibrium concentrations of the species are to be found, given the total (analytical) concentrations of all components and the stoichiometry and stability constants of the species. A computer code, MICROQL, was developed to solve such a chemical equilibrium problem [*Westall*, 1979]. MICROQL is a scaled-down version of the comprehensive chemical equilibrium computer code, MINEQL [*Westall et al.*, 1976], and

TABLE 2. Component Material Balance Equations

Component	Material Balance Equation
$T_1 = \mathrm{Cl}_T$	$= [\mathrm{Cl}^-] + [\mathrm{CdCl}^+] + 2[\mathrm{CdCl}_2]$
$T_2 = \mathrm{Br}_T$	$= [\mathrm{Br}^-] + [\mathrm{CdBr}^+] + 2[\mathrm{CdBr}_2]$
$T_3 = \mathrm{Cd}_T$	$= [\mathrm{Cd}^{2+}] + [\mathrm{CdCl}^+] + [\mathrm{CdCl}_2] + [\mathrm{CdBr}^+] +$ $[\mathrm{CdBr}_2] + [\mathrm{CdOH}^+] + [\mathrm{SOCd}^+]$
$T_4 = \mathrm{SOH}_T$	$= [\mathrm{SOH}] + [\mathrm{SOH}_2{}^+] + [\mathrm{SO}^-] + [\mathrm{SOCd}^+]$
$T_5 = \mathrm{T}_{\sigma T}$	$= [\mathrm{SOH}_2{}^+] - [\mathrm{SO}^-] + [\mathrm{SOCd}^+]$
$T_6 = \mathrm{H}_T$	$= [\mathrm{H}^+] + 2[\mathrm{SOH}_2{}^+] - [\mathrm{CdOH}^+] - [\mathrm{OH}^-] -$ $[\mathrm{SO}^-] - [\mathrm{SOCd}^+]$

T_σ represents the charge on the surface, defined as the excess of positive groups over negative groups.

Figure 3.2 Poorly scanned page. Notice the page is not aligned and the left side of the page is bent back.



ments which could cause severe general intergranular attack in heavily sensitized austenitic stainless steels. In recent years, experience has shown that "moderately" sensitized materials can undergo intergranular stress-corrosion cracking (IGSCC) in environments that do not cause appreciable intergranular attack in the absence of stress [1].[3] Thus, a more discerning test is required for these applications. Conversely, there are applications where the use of moderately sensitized material, which could not pass the current practices for determining the presence of sensitization, would perform satisfactorily in the service environment [2]. In these cases, a manufacturer would be forced to use an extra-low-carbon or stabilized grade of material, with the associated cost or strength penalties. A test that measured the degree of sensitization, in conjunction with calibration tests in the service environment, could provide a "go/no go" materials acceptance criteria for both cases. A rapid nondestructive test would also to helpful for quality control on shop- or field-constructed components which receive thermal treatments during fabrication.

Figure 3.3 Typical scanned page.

CHAPTER 4

EFFECTS OF OCR ERRORS ON RETRIEVAL

Introduction

In this chapter, we review three information retrieval (IR) systems working with OCRed document collections. The metrics used to compare the result set for the OCRed collections with the result set for the corrected collections are different for each IR system. Part of this is due to the fact that the probabilistic and vector space models systems produce a ranking of the results; the boolean system produces an unordered result set. The most detailed level of granularity is used in the metrics for probabilistic system, which is tested on three document collections of varying OCR quality. For the vector space model, only a summarized and condense version of the results is presented. However, there were ten variation of the vector space model's key parameters used in the study. All three of these studies are covered in the paper by Taghva, Borsack, and Condit (1996b).

Effects Of OCR Errors On Boolean IR

A document collection from the Department of Energy containing 204 documents was given to the researchers (Taghva, Borsack, Condit, & Erva, 1994). These scientific documents contained formulas, graphs, photos, and maps. There were sixteen major subject areas. A level of 99.8% character accuracy was reported for the collection.

The collection was loaded into the BASIS Plus information retrival system (Delphi Consulting Group, 1991). It was originally developed by Information Dimensions Inc. of Dublin, Ohio, which was acquired by Open Text Corporation in 1998. The boolean logic with an inverted file model has been (Salton, 1989) and still is the most widely used technology as seen in the Google search engine (Brin & Page, 1998). Querying this type of information retrieval system is so popular it has become an English verb: googling.

There were 205 unique search terms for the seventy-one queries. The average number of terms per query was five. If we count the number of queries that return exactly the same result set documents, we have 63. That means 88.8% of the queries return identical results.

If we take a closer look at the results sets, and count the number of correct documents retrieved across all the queries, 617 out of a possible 632, the results are even closer. That means 97.6% of documents are retrieved. We consider a 95% or better recall statistically the same as the 100% recall for the clean text.

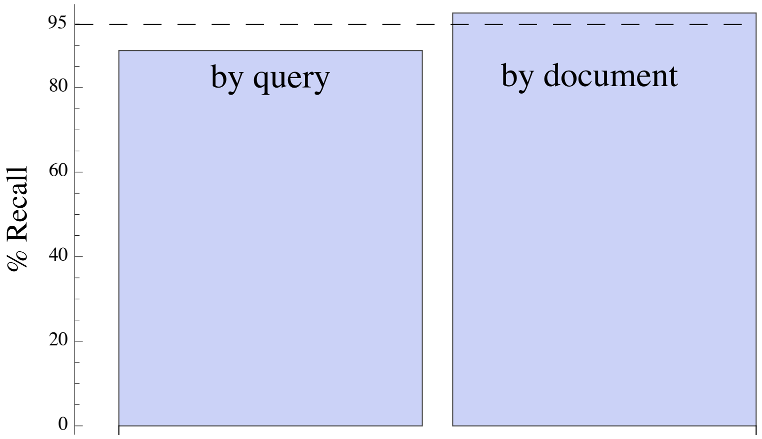


Figure 4.1 Boolean Model: recall by query and by document

All the statistical studies in this dissertation assume a level of significance (α) of 5%. Choosing the level of significance is somewhat arbitrary; but, for many IR and IE studies, a level of 5% is the conventional choice. A more formal approach would involved a measuring the signal to noise ratio and considering sample size to determine the most precise level of confidence.

Effects Of OCR Errors On Probabilistic IR

Another document collection from the Department of Energy containing this time 674 documents was given to the researchers. These documents were recognized by three different OCR devices that varied in their level of character accuracy: the highest accuracy (98.14%), an intermediate accuracy (97.06%), and a lowest level of accuracy (94.63%). These three document collections were separately loaded into a probabilistic information retrieval system. This system, called INQUERY, was developed at the University of Massachusetts (Callan, Croft, and Harding, 1992).

The probabilistic approach uses inference networks to model query and document content based on concepts. Roughly, concepts correspond to terms. Concept associations are based on document-term frequency and collection-term frequency. The formula below is used to assign concept i to document j:

$$k + \left[(1-k) \times \frac{\log(t_{ij})}{\log(\max tf_j)} \times \frac{\log \frac{c}{f_i}}{\log(c)} \right]$$

where

$c \equiv$ collection size

$f_i \equiv$ number of documents in which concept i occurs

$t_{ij} \equiv$ frequency of concept i in document j

$\max tf_j \equiv$ maximum concept frequency in document j

The parameter k serves to bias the initial belief of term relevance and can be adjusted by the user to fit the collection. For this study, the default was k = 0.4 was used.

Each query produced a ranked list of documents for both the correct data and the OCR data. We compute the precision at 10 standard recall levels points. For example, to compute the precision at recall level 10%, we see how far down the ranked results set we need to go to achieve 10% recall. Suppose there a total of 100 relevant documents, and the top 20 documents contain 10 relevant documents. That means the top 20 documents have a 10% recall and a 50% precision because only half of the documents are relevant. The table below shows the precision and recalls results for the four document collections.

Table 4.1 Precision at standard recall points

Recall	Correct Data	OCR Best	OCR Middle	OCR Worst
10	53.5	53.1	52.9	54.0
20	46.0	44.8	45.7	47.6
30	38.6	38.9	40.8	39.7
40	33.8	33.8	35.3	35.6
50	30.4	30.5	31.0	31.9
60	23.3	24.6	24.6	24.7
70	17.8	17.3	17.5	18.5
80	14.7	13.5	13.2	14.1
90	12.7	11.1	10.8	11.6
100	11.8	10.5	10.2	10.6
Average	28.3	27.8	28.2	28.8
% Change		-1.6	-0.2	2.0

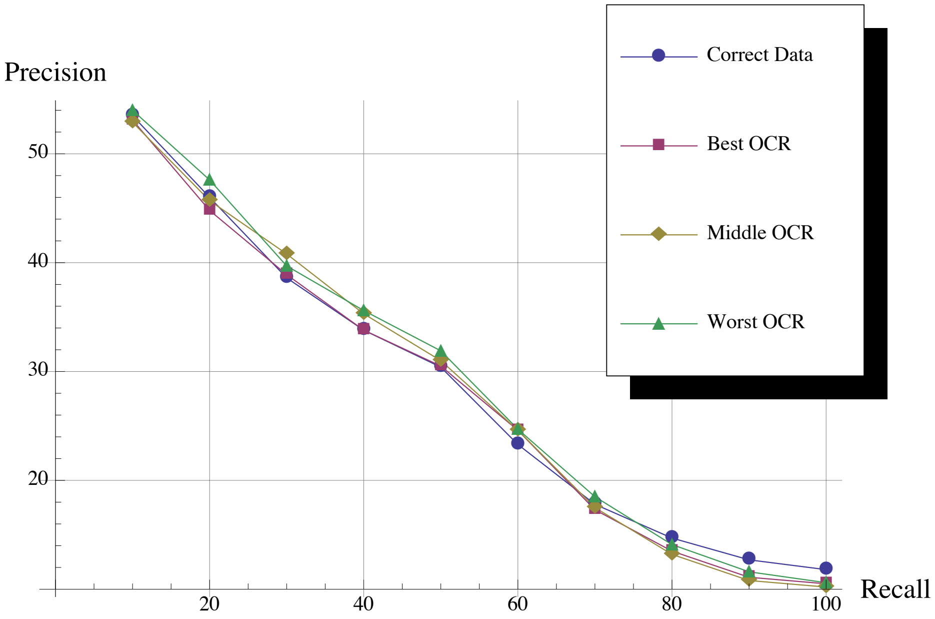


Figure 4.2 Precision-Recall curve at 10 standard recall points

All four curves are nearly identical, and the differences so small so as to consider them insignificant from an information retrieval perspective. If the percent change between the correct data and any of the OCR collections for the average precision, was greater than 5%, then we would considered the difference to be significant.

Effects Of OCR Errors On Vector Space Model IR

Another document collection from the Department of Energy containing this time 674 documents was given to the researchers. Two version of this single collection were generated: one an OCR generated collection with 80-90% level of character accuracy, and another that was a manually corrected version of these same documents. Roughly, the same set of queries as were used in the study of the Probabilistic IR system were used for this study.

In Vector Space Model, every document (or query) is represented by a corresponding vector:

$D = \left(d_1,\cdots,d_t\right)$ where each weight d_i is non-zero if the term appears in the document (or query). An inner product can provide a scalar value representing the similarity between two documents. The SMART IR systems uses a three part weighting scheme, with each weight having a document and a query component (Salton & Buckley, 1988):

1. **Term Frequency Component:** considers term frequency with respect to a single document, $a_D \times a_Q$
2. **Collection Frequency Component:** considers term frequency with respect to the entire collection. $b_D \times b_Q$
3. **Vector Normalization Component:** treats long and short documents with some equality. $c_D \times c_Q$

The overall dot product is

$$D \bullet Q = \left(c_D \times c_Q\right)\sum_{\text{terms}}\left(a_D \times a_Q\right)\times\left(b_D \times b_Q\right)\times d_i \times q_i$$

Before explaining the different options for each of the three weights, some definitions are needed.

- *tf* term frequency, total number of occurrence of a term in a given document
- *cf* collection frequency, total number of occurrences of term in the collection
- *df* document frequency, total number of documents with a given term

Below are three table with each of the options for the three components of the weighting scheme. Within the SMART system, the user is free to choose any of these combinations, but some have proven to be superior for collections with particular idiosyncrasies (Salton & Buckley, 1998).

There are a total of $(3 \times 4 \times 3)^2 = 1296$ possible weighting combinations. A carefully selected ten of those combinations were chosen for this study. Unfortunately, the history of the selection criteria is not readily available. The SMART system has a concise notation for labeling the particular weighting scheme: xxx.xxx. The three characters to the left of the period represent the weighting combination used for the document, and the three to the right, for the query. For example, atc.atn applies augmented normalization term frequency (the first a), inverse document frequency (the first t), and cosine normalization (the second a) to the document vector, and augmented normalization term frequency (the second a), inverse document frequency (second t), and no normalization (the n) to the query vector.

Table 4.2 Options for a weights

Abbreviation	Description	Formula
N	none	1
A	augmented normalized	$0.5+0.5\left(\frac{tf}{\max tf}\right)$
L	natural logarithm	$\ln(tf)+1$
S	square	tf^2

Table 4.3 Options for b weights

Abbreviation	Description	Formula
N	none	1
T	inverse doc frequency	$\log\left(\frac{\# docs}{cf}\right)$
P	probabilistic	$\log\left(\frac{\# docs - cf}{cf}\right)$
S	square	$\left(\log\left(\frac{\# docs}{df}\right)\right)^2$

Table 4.4 Options for c weights

Abbreviation	Description	Formula
N	none	1
T	sum	$\frac{tf}{\sum weights}$
C	cosine	$\frac{tf}{\sum weights^2}$

Table 4.5 Average Precision for Correct and OCR Collections

Weighting	Correct Data	OCR Data	% Difference
nnn.atn	0.2457	0.2497	1.63
ann.atn	0.3222	0.3308	2.67
lnn.atn	0.3311	0.3421	3.32
snn.atn	0.2053	0.2006	-2.29
ntn.atn	0.2898	0.2881	-0.59
npn.atn	0.3110	0.3153	1.38
nsn.atn	0.3305	0.3269	-1.09
nns.atn	0.2913	0.2873	-1.37
nnc.atn	0.3500	0.3381	-3.40
atc.atn	0.2228	0.2235	0.31

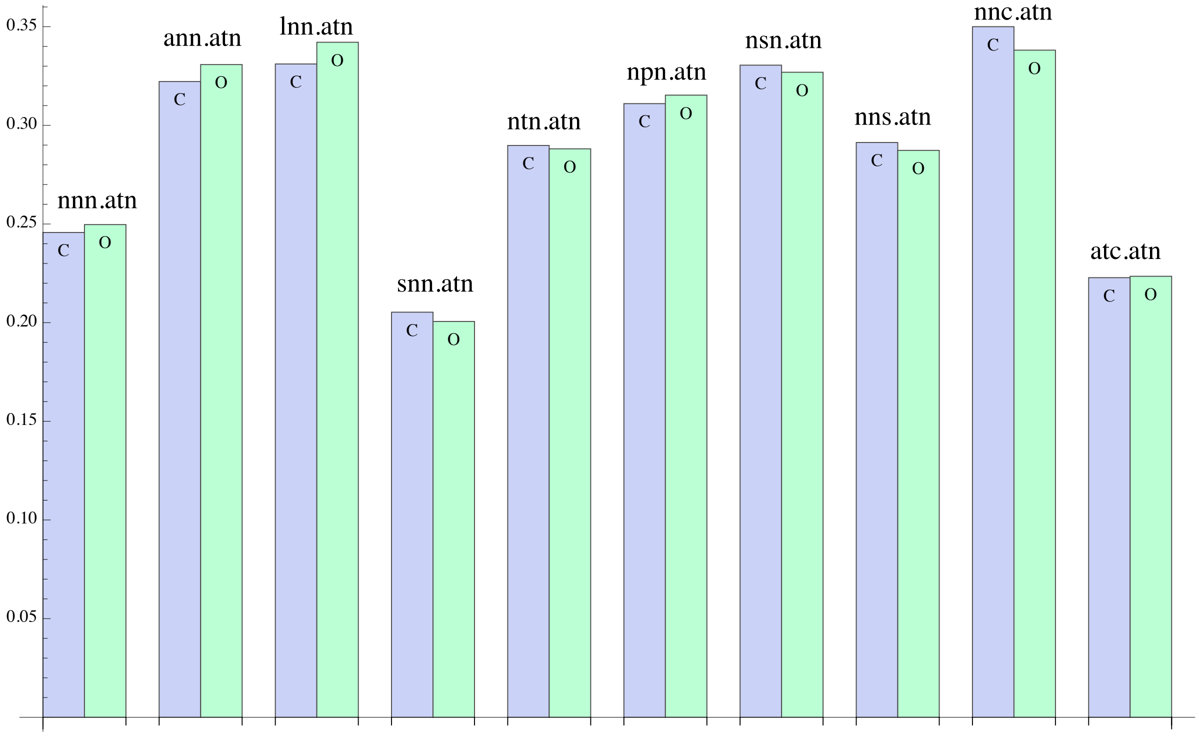


Figure 4.3 Average Precision for Correct (c) and OCR (o) Collections

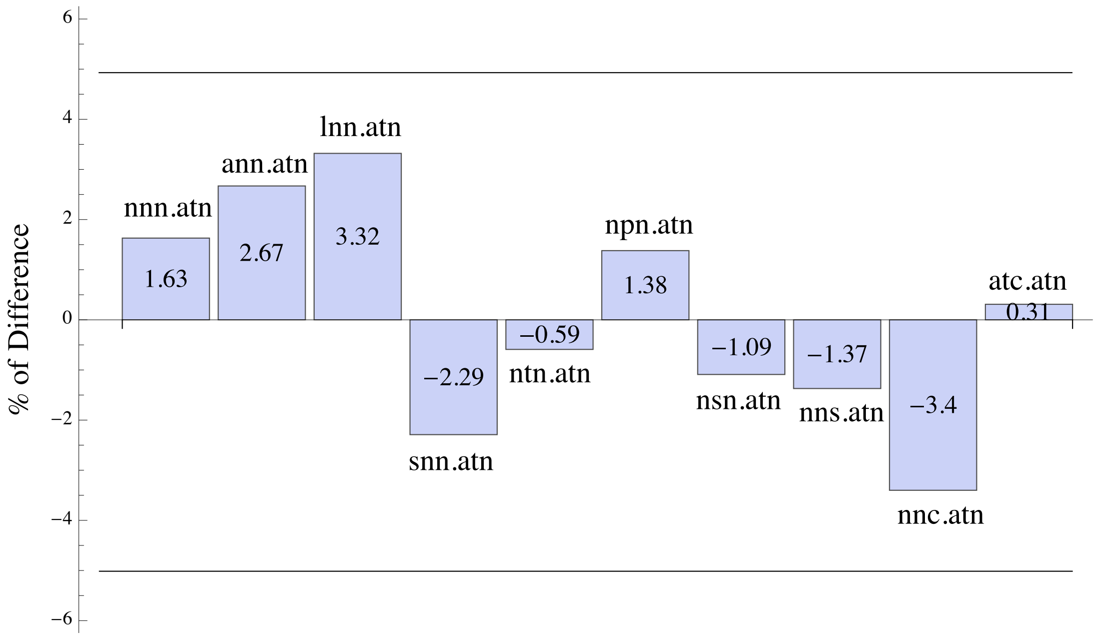


Figure 4.4 % Change in Average Precision for Correct-OCR

The order of the different schemes has no special meaning. If any of the bars had been above or below the 5% marks, then we would have considered the differences significant.

CHAPTER 5

METHODS FOR INFORMATION EXTRACTION USING HIDDEN MARKOV MODELS

Introduction

In this section we describe the development and effectiveness of a Hidden Markov Model (HMM) based extraction program to identify potentially private addresses. We found that HMMs provide high recall and precision for this task. However, we also discovered that optical character recognition (OCR) errors degraded the performance of the program. This last result is somewhat surprising given that document retrieval has been shown, on average, not to be so affected (Taghva, Coombs, Pereda, & Nartker, 2005). We also discovered that for our task, the smoothing technique called shrinkage did not improve our results.

We will first describe the problem our HMM is designed to solve in the next section. We discuss previous attempts to extract information such as addresses using HMMs. Presented thereafter is a standard definition of HMMs and the details of our implementation of the address-finding HMM. We describe how the HMM was trained. We next discuss some problems resulting from errors in the OCR processing of training documents. Finally, we describe the construction and outcomes of experiments measuring the effectiveness of several variations on our address-finding HMM for various types of documents.

Applications for Address Extraction

United States Department of Justice (1974) established a code of fair information practices that governs the collection, maintenance, use, and dissemination of personally identifiable information about individuals. According to the Privacy Act of 1974, the United States Government cannot make public private information about American citizens. However, to comply with the Freedom of Information Act, the Federal Government must make many of its documents and records available to the public. These documents, therefore, must be reviewed to determine if they contain private information. Since the number of documents to be reviewed is very large, and human review is costly and time-consuming, it is clearly advantageous to seek automatic methods for identifying likely private information in documents.

For certain types of private information, such as social security numbers, a simple regular expression matching technique proved effective for the majority of cases. However, the prospects of such a solution for identifying addresses seemed unlikely as we reviewed the data and noted the wide variation in address formats. In addition, many documents had been OCRed and some noise had been introduced into the target addresses. Since Hidden Markov Models have been applied to problems similar to ours with some success, we decided to explore the use of an HMM to identify personal addresses (Bikel, Miller, & Weischedel, 1997; Leek, 1997).

Our task is to identify a set of documents such that the likelihood of them containing private information is extremely low. In other words we want to locate the “easy” examples of non-private documents so that they can be released in accordance with the Freedom of Information Act.

Documents that we place in the “private” category will very likely still contain documents without private information. However, these more difficult cases will be subject to a manual review by human experts. Our automatic process aims to reduce the number of documents requiring human review but does no eliminate the need for such a review completely.

Thus, as we will see, the worst mistake for us is a false negative: a document containing private information in which our HMM finds none. We can accept, within reason, a higher rate of false positives if doing so decreases false negatives. We can accept loses in precision to boost recall.

Hidden Markov Models

A Hidden Markov Model is a finite state automaton with probabilistic transitions and symbol emissions. An HMM consists of

- A set of states, $S = \{s_1 \mathrm{K}\ s_n\}$
- An emission vocabulary, $V = \{w_1 \mathrm{K}\ w_n\}$
- Each state is associated with a probability distribution over emission symbols where the probability that a state s emits symbol w is given by $P(w \mid s)$.
- Each state is further associated with a probability distribution over the set of possible outgoing transitions. The probability of moving from state s_i to s_j is given by $P(s_i,s_j)$.

- Finally, a subset of the states are considered start states, and to each of these is associated an “initial” probability that the state will be a start state.

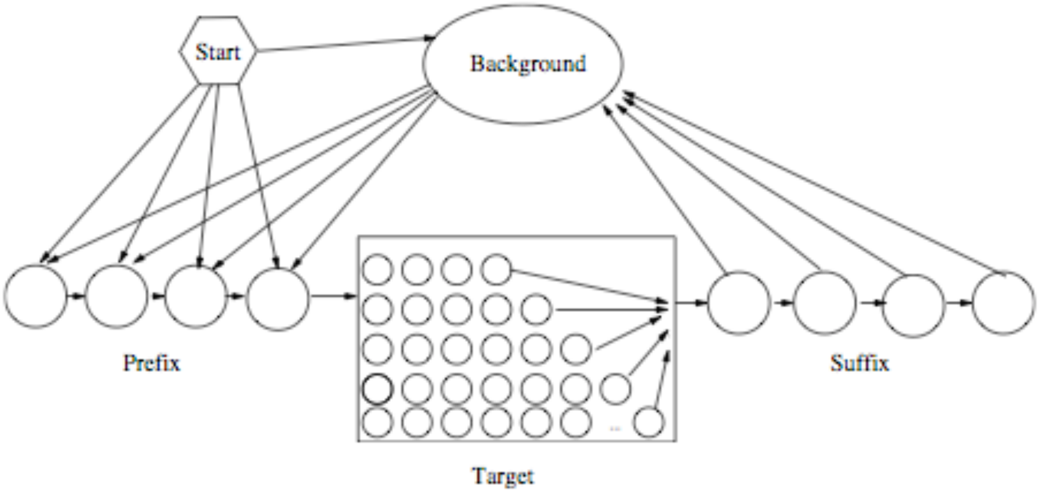
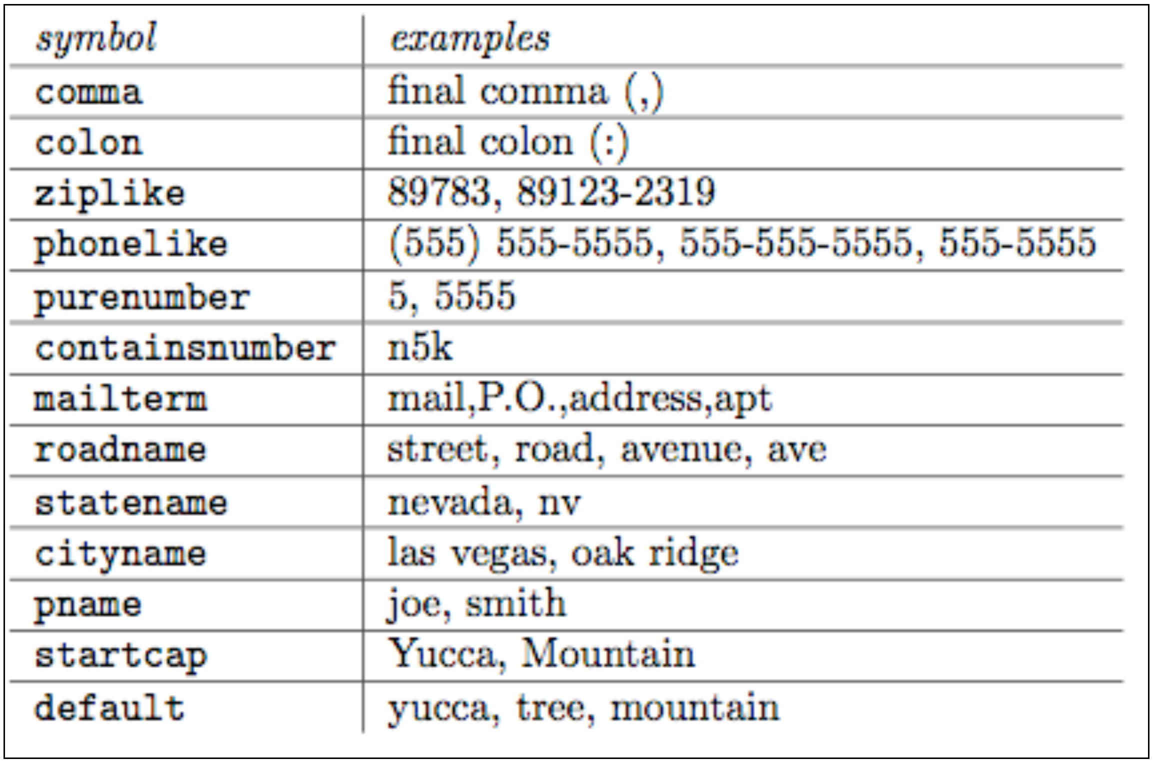


Figure 5.1 Hidden Markov Model for Addresses

Following the model of Freitag and McCallum (1999), our HMM has four types of states. *Background* states represent symbols that are not part of an address nor part of the “context” of one. We defined the “context” to include the four terms in the document prior and the four terms subsequent to the address. The prior four terms correspond to what we call the four *prefix* states of an address, and the following four terms in the context are four *suffix* states. *Target* states represent the parts of the address. Figure 1 provides an illustration of the structure of our HMM. More specifically, figure 1 illustrates the structure of our “clean” HMM as discussed more in chapter 8.



symbol	*examples*
comma	final comma (,)
colon	final colon (:)
ziplike	89783, 89123-2319
phonelike	(555) 555-5555, 555-555-5555, 555-5555
purenumber	5, 5555
containsnumber	n5k
mailterm	mail,P.O.,address,apt
roadname	street, road, avenue, ave
statename	nevada, nv
cityname	las vegas, oak ridge
pname	joe, smith
startcap	Yucca, Mountain
default	yucca, tree, mountain

Figure 5.2 HMM Hidden Symbols used for Address Extraction

Figure 5.2 displays how we defined the emission vocabulary for the HMM. As in Bikel (1997), vocabulary identification had an ordering which is reflected in the list in our figure. This means that a term is first scanned to determine if is ends in a comma. If so, the comma is removed and the comma itself is represented in the HMM with the distinct symbol comma. Next, the remainder of the term is checked to determine if it is a five or nine digit number with an optional hyphen after the fifth digit. In such a case the term is represented in the HMM with the symbol ziplike. The symbol colon is only added if a term ends in a colon prior to the removal of a final comma. If it is not such a number, then it is checked if it is phonelike, and so on.

Terms were placed in this taxonomy in two ways. Either a straightforward regular expression match was used, as in the case of ziplike and phonelike, or a term was matched against a database of known objects. For example, a very short list of city names most relevant to the documents was generated. This list took into consideration that most of the documents concerned government employees and contractors from government offices and businesses dealing with nuclear power.

After a document is converted to symbols by this taxonimizing process, addresses are discovered by applying the well-known Viterbi algorithm to the resulting string of symbols. As explained in Rabiner (1993), the Viterbi algorithm uses dynamic programming techniques to determine the most likely state sequence for the document. Of interest to us are those subsequences consisting of target states since they represent the most likely addresses.

Training Hidden Markov Models

Training for the HMM was based on tagged documents which were marked by human experts with the Information Science Research Institute's (ISRI) META-Marker tool (Taghva, 2004). The topology of transitions as illustrated in figure 5.1 was predetermined except for the lengths of the target sequences, which were determined by the marked data. Figure 5.1 is somewhat simplified for aesthetic purposes. Transitions from suffix states to any of the prefix states was allowed, and a final target state could transition to a prefix state if addresses appeared close together in training data.

Transition probabilities were estimated by Maximum Likelihood:

$$P(s_i, s_j) = \frac{\text{number of transitions from } s_i \text{ to } s_j}{\text{total number of transitions out of } s_i}$$

Initial probabilities representing the beginning of the document were set by hand. The initial probability was divided uniformly over the background state, all prefix states, and the first target state.

The emission probability table is estimated with Maximum Likelihood supplemented by smoothing. Smoothing is required because Maximum Likelihood estimation will sometimes assign a zero probability to unseen emission-state combinations.

Prior to smoothing, emission probabilities are estimated by:

$$P(w \mid s)_{ml} = \frac{\text{number of times symbol } w \text{ is emitted at state } s}{\text{total number of symbols emitted by state } s}$$

We used *absolute discounting* to smooth emission probabilities. Absolute discounting consists of subtracting a small amount of probability p from all symbols assigned a non-zero probability at a state s. Probability p is then distributed equally over symbols given zero probability by the Maximum Likelihood estimate.

If v is the number of symbols assigned non-zero probability at a state s and N is the total number of symbols, emission probabilities are calculated by

$$P(w \mid s) = \begin{cases} P(w \mid s)_{ml} - p & \text{if } P(w \mid s)_{ml} - p > 0 \\ \dfrac{vp}{N - v} & \text{otherwise} \end{cases}$$

Freitag (1999) reports that there is no known solution for determining the optimal value for p. We followed Borkar (2001) in using

$$p = \frac{1}{T_s + v}$$

where T_s is the total number of symbols emitted by state s, i.e., the denominator of $P(w \mid s)_{ml}$.

In addition to absolute discounting, Freitag and McCallum (1999) propose applying a more elaborate smoothing technique called *shrinkage*. According to this approach, various alternative distributions are defined for the emission probability $P(w \mid s)$ for each state s. The *default* distribution is absolute discounting. Further, each probability can "shrink" to its *parent* distribution. For example, the target parent is estimated by dividing the number of times a symbol is emitted by any target state by the total number of symbols emitted by *any* target state. The non-target *ancestor* shrinks to the common ancestor state of all background, prefix, and suffix states. The *context grandparent* shrinks prefix and suffix states to a state common to all prefixes and suffixes. Finally each distribution can shrink to the *uniform* distribution.

Table 5.1 Types of Shrinkages Possible For Each State

state type	**default**	**parent**	**context grandparent**	**non-target ancestor**	**uniform**
background	✓	✓		✓	✓
target	✓	✓			✓
suffix	✓	✓	✓	✓	✓
prefix	✓	✓	✓	✓	✓

For each symbol-state combination, the final emission probability $\hat{P}(w \mid s)$ is determined by a weighted sum of the estimates from the table above. For example, a given suffix state will have a default probability, a suffix parent, a context grandparent, a non-target ancestor, and the uniform distribution probabilities associated with it, and each of these will be assigned a weight $\{\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5\}$. We thus define

$$\hat{P}(w \mid s) = \sum_{i=1}^{k} \lambda^i P(w \mid s^i)$$

In the case of a suffix state, for example, $\{P(w \mid s^1), P(w \mid s^2), P(w \mid s^3), P(w \mid s^4), P(w \mid s^5)\}$ would represent its default, suffix parent, context grandparent, non-target ancestor, and uniform probabilities. The value k is the number of alternative probabilities for a given type of state. For a suffix state, $k = 5$.

Optimal values for the λ weights are discovered using Expectation Maximization. Following Freitag (1999), we initialize the λ's to $\frac{1}{k}$. Holding out each symbol in turn in a hold-out set H_j, we perform the following two steps until the values of all λ's no longer change:

E-step: Determine how much each emission state predicts the symbols from a held-out set H_j for each state s:

$$\beta^i = \sum_{w \in H_j} \frac{\lambda^i P(w \mid s^i)}{\sum_m \lambda^m P(w \mid s^m)}$$

M-step: Derive new λ's by normalizing the β's:

$$\lambda^i = \frac{\beta^i}{\sum_m \beta^m}$$

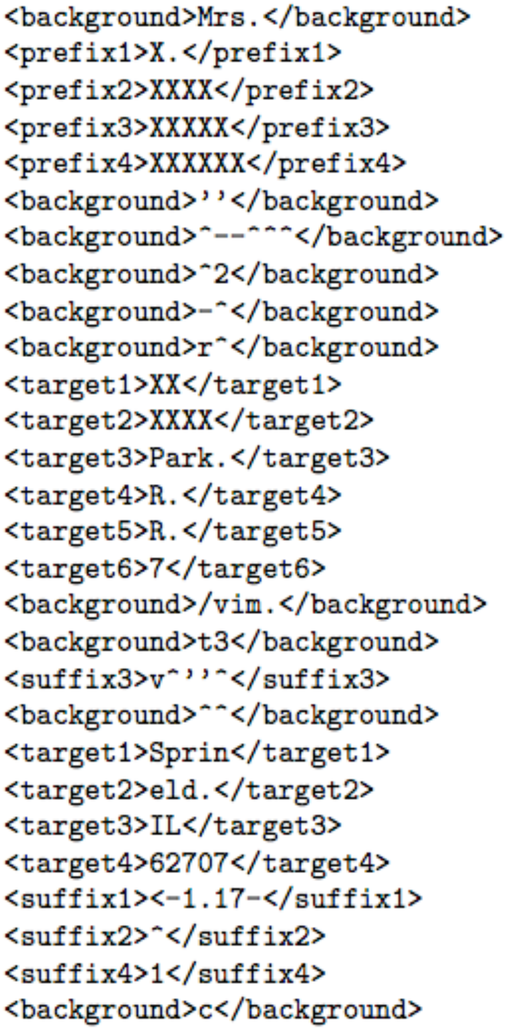
Finally, the output of our HMM consists of sequences of strings which are likely addresses. In fact, the HMM will sometimes discover non-addresses, and the HMM has no concept of a "private" address. For these reasons, filters had to be introduced to remove obvious non-addresses and known "public" addresses.

CHAPTER 6

EFFECTS OF OCR ERRORS ON INFORMATION EXTRACTION USING HIDDEN MARKOV MODELS

OCR Related Difficulties

Several difficulties occurred in OCR documents which affected the training for the HMM. Roughly, they can be divided three categories: text ordering not matching human visual ordering, information loss during OCR, and handwritten addresses.



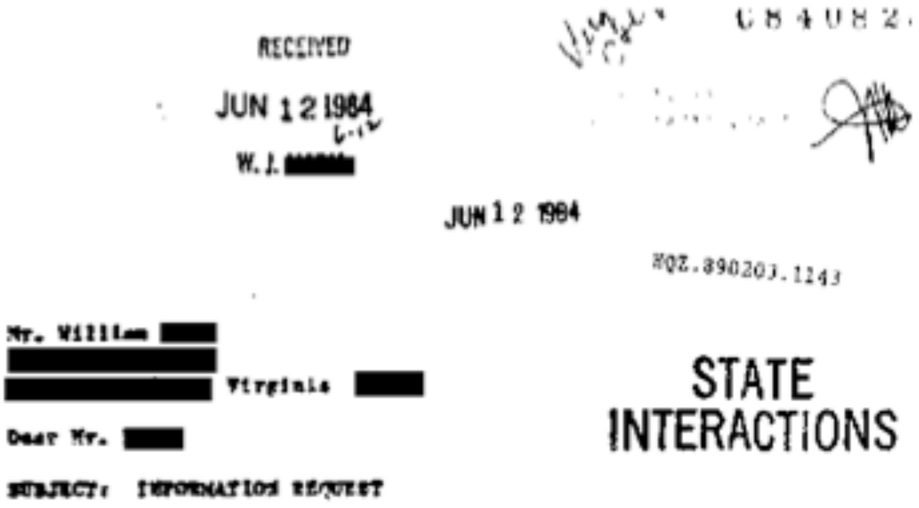
```
<background>Mrs.</background>
<prefix1>X.</prefix1>
<prefix2>XXXX</prefix2>
<prefix3>XXXXX</prefix3>
<prefix4>XXXXXX</prefix4>
<background>''</background>
<background>^--^^^</background>
<background>^2</background>
<background>-^</background>
<background>r^</background>
<target1>XX</target1>
<target2>XXXX</target2>
<target3>Park.</target3>
<target4>R.</target4>
<target5>R.</target5>
<target6>7</target6>
<background>/vim.</background>
<background>t3</background>
<suffix3>v^''^</suffix3>
<background>^^</background>
<target1>Sprin</target1>
<target2>eld.</target2>
<target3>IL</target3>
<target4>62707</target4>
<suffix1><-1.17-</suffix1>
<suffix2>^</suffix2>
<suffix4>1</suffix4>
<background>c</background>
```

Figure 6.1 Tagging of HQZ.19880629.6150

OCR RELATED DIFFICULTIES

Several difficulties occurred in OCR documents which affected the training for the HMM. Three major difficulties are detailed below.

While the META-marker tool allowed experts to mark zones on the document image which were derived from OCR information, the ordering of the text in the OCR output did not always correspond to the ordering inferred by the human expert viewing the image. Figure 6.2 shows a redacted private address in the context of handwritten text. Figure 6.1 reveals how the OCRed text was tagged. (The figures are in presented out of order due to formatting requirements of this document. Conceptually, the processing proceeds from image, to OCR, and to tagging, which corresponds to figures 6.2, 6.3, and 6.1.) Prefix-tagged terms were separated from target-tagged terms by background-tagged terms and the suffix-tagged terms were taken out of order. In fact, the term tagged as “suffix3” was placed in the midst of background-tagged terms, and a single address was tagged as two distinct addresses. Obviously, there is only a tenuous correspondence between the tagged address and the “ideal” topology of figure 5.1. “X” is used in figure 6.1 to redact private information.



RECEIVED

JUN 12 1984

W. J. [redacted]

C 8 4 0 8 2.

JUN 12 1984

HQZ.890203.1143

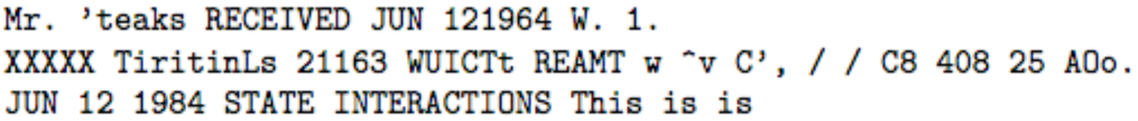
Mr. William [redacted]
[redacted]
[redacted] Virginia [redacted]

Dear Mr. [redacted]

SUBJECT: INFORMATION REQUEST

STATE
INTERACTIONS

Figure 6.2 Address in HQZ.19890203.1143

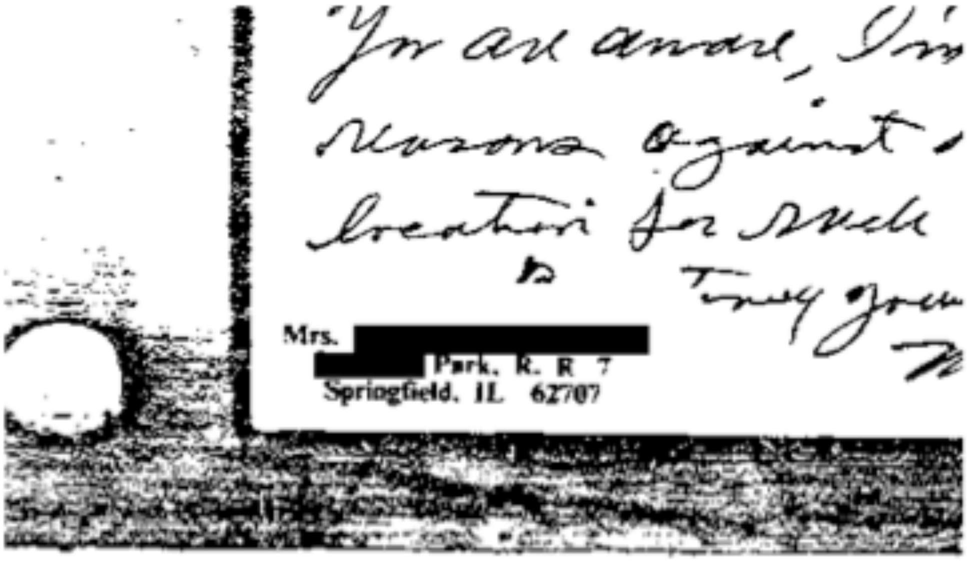


```
Mr. 'teaks RECEIVED JUN 121964 W. 1.
XXXXX TiritinLs 21163 WUICTt REAMT w ^v C', / / C8 408 25 AOo.
JUN 12 1984 STATE INTERACTIONS This is is
```

Figure 6.3 Resulting OCR of HQZ.19890203.1143

In some documents, addresses viewable in the document images were either partially or completely lost in the OCR process. Also, useful context information would occasionally be lost as well. Figure 6.3 and its corresponding OCR in figure 6.1 provide an example of lost information.

In some documents handwritten addresses appeared which were not recognized by the OCR. This can be seen in figure 6.4. Several documents were simply removed from our training and testing sets because these problems recognizing handwriting appear to us to require a different approach entirely.



You are aware, I'm
reasons against
creation for such
to Tiny you

Mrs. [redacted]
[redacted] Park, R. R 7
Springfield, IL 62707

Figure 6.4 Address in HQZ.19880629.6150

HMM Experiments

The first two problems noted in the previous section brings up the question whether the tagging should be "cleaned up" to reflect our preconceived ideas about how the HMM should be structured, or whether it should be left "as is" to reflect the reality of the data.

We decided to compare the performance of an HMM trained on tagged documents which contained "incorrect" tags with an HMM trained only on documents where the tags fit our assumed topology illustrated in figure 5.1.

Human experts identified 251 documents and emails in a large collection of documents from various government agencies which contain addresses subject to the privacy act. Addresses in 187 of these documents were tagged with the META-marker tool. Documents with handwritten addresses were removed from the 187 and 89 were randomly selected for training what we call the default HMM. Some of these 89 documents contain "incorrect" tagging due to the OCR difficulties described in the previous section.

A second training set was derived from the first by removing all examples which contained "incorrect" tagging. The HMM based on this training set was called clean.

A test set of 614 documents and emails was constructed from (1) a random sample from the 251 documents containing private addresses, with hand-written examples removed, (2) a random sample of emails from a large (more than 20,000) collection of government emails, and (3) a random sample from 2,000 government documents.

Three experiments were performed. In the first, the default HMM was run on the test set. In the second, the clean HMM was run against the same data. Finally, the emission table for the clean HMM was optimized using shrinkage.

The standard performance measures of precision, recall, and F1 were calculated for each experiment. Let TP be the number of true positives, that is, the number of documents which both experts and the HMM agreed contain private addresses. Let FN be the number of false negatives, i.e., the number of documents which experts said contain privacy, but the HMM marked as not having privacy. We then define recall as

$$recall = \frac{TP}{TP + FN}$$

Letting FP signify the number of false positives, i.e., those documents which the HMM marked as containing privacy but which experts decided does not, precision is defined as

$$precision = \frac{TP}{TP + FP}$$

The harmonic mean of precision and recall is called the F1 measure, defined as:

$$F1 = \frac{2}{\frac{1}{precision} + \frac{1}{recall}}$$

The results of our experiments are presented below.

Table 6.1 Precision, Recall, and F1 Score Results

EXPERIMENT	TP	FP	FN	TN	PRECISION	RECALL	F1
DEFAULT	104	4	9	495	0.963	0.920	0.941
CLEAN	108	16	5	483	0.871	0.956	0.911
SHRINKAGE	108	28	5	471	0.794	0.956	0.867

The value TN represents the number of true negatives, which are documents which both the experts and the HMM agreed did not contain private addresses.

CHAPTER 7

METHODS FOR INFORMATION EXTRACTION USING PRECISE SHALLOW PARSING

Introduction

Federal, state, and local governments maintain records containing very specific personal information about individuals. Hospitals also have records of this sort of information. Personally Identifiable Information (PII), as it is called, refers to any information that identifies or can be used to identify, contact, or locate the person to whom such information pertains. This includes information that is used in a way that is personally identifiable, including linking it with identifiable information from other sources, or from which other personally identifiable information can easily be derived, including, but not limited to, name, address, phone number, fax number, email address, financial profiles, patients' records, social security number, and credit card information.

Among many projects dealing with patients’ records, the National Institutes of Health (NIH) is interested in applications of data mining to the vast amount of text. This text must not include any identifiers, for example name, postcode, date of birth, or National Health Service (NHS) number. Typically the identifiers must be coded so that the information cannot directly identify an individual, but a 'key' is available to re-link data with identifying information. Identification of PII is necessary to avoid accidental release of un-coded data. Knowing that the number of records that contain PII is substantial and human review is costly. It is clearly advantageous to seek automatic methods for identifying PII.

In this chapter, we describe the development and effectiveness of a pattern-based extraction program to identify potentially private date of birth information. We found that pattern-based extraction provides high recall and precision for this task. However, we also discovered that optical character recognition (OCR) errors degraded the performance of the program. We also discovered some techniques for making our extraction program resistant to OCR errors.

In the following sections, we will first describe the problem our information extraction program is designed to solve and then we describe the detailed design of our solution.

Extracting Date Of Birth Information

The Department of Energy (DOE) is publishing many once private documents concerning the development of the Yucca Mountain Project. For most documents, nobody knew if they contained private information. The DOE is required to publish this information in order to comply with the Freedom of Information Act. However, these documents must be "scrubbed", or redacted, of personal information before publishing. This is in accordance with the Privacy Act of 1974, which states that the United States Government cannot make public private information about American citizens. Since the number of pages to be reviewed is in the millions, human review is costly and time consuming. We are developing automatic methods to identify the documents that do not contain privacy information with very high probability. These non-private documents will be published without further human review. The remaining "potentially private" documents would be sent on for human review. For the document collection from the DOE, the amount of human review would be reduced by a factor of ten roughly, estimated from the ratio of private to non-private documents and the precision of our extraction system. There will likely be many potentially private documents that will turn out not to be private after careful human review.

Our method of evaluation for finding date of birth will mirror the standard method in most information extraction experiments. The abbreviation DOB stands for “date of birth”. We use:

$$\text{recall} = \frac{\text{number of correct DOB hits}}{\text{total number of DOB's inside the document collection}}$$

and

$$\text{precision} = \frac{\text{number of correct DOB hits}}{\text{total number of DOB's reported by the program}}$$

We also look at the results on a more coarse grain by computing recall and precision on a per document basis as shown in the equations below:

$$\text{recall} = \frac{\text{number of correct DOBdocument hits}}{\text{total number of documents containing DOBinformation}}$$

and

$$\text{precision} = \frac{\text{number of correct DOB document hits}}{\text{total number of DOB documents flagged by program}}$$

When it is deemed that further human review is necessary, the reviewer will examine the entire document not just simple windows of text with potential DOB information. Precision and recall on a per document basis more closely models the effectiveness of our human-and-machine review process.

Finding Extraction Patterns

The first problem we tackled was that of extracting all the dates which come in many different formats. A crude UNIX grep(1) of lines with digits was performed to reduce the amount of text under review for finding all formats. It quickly became evident that many of the dates in the documents were much more recent than the birth dates, such as the dating of a letter. By filtering the dates by years before 1990, we reduced the number or dates to consider by three orders of magnitude and removed no known birth dates from the training data. Next, the text to the left and right of each date was trimmed and each occurrence of a digit was replaced by the single letter d. What remained were roughly the nine patterns for all the observed date formats. Below is an example of each pattern:

- 1968-3-15
- January 15, 1968
- 3/15/1968
- 3-15-1968
- 4th of July, 1968
- 15 September, 1968
- 09-18-21
- 09/18/21
- l5-JUN-53

While testing data was being prepared with ground truth information, we decided to mine the web using Google for extraction patterns. We searched for "George Washington" and pulled out snippets that contained his birthday as well as his name.

Below is a sample list:

- George Washington (born February 22,1732),
- George Washington Tuesday, February 22;
- George Washington's Birthday is February 22.
- GEORGE WASHINGTON, b. February 22,
- George Washington was actually born February 22, 1732
- I learned about George Washington. He was born February 22, 1732.
- George Washington ~Date of Birth~ February 22, 1732
- George Washington, the first president of the United States, was born on February 22, 1732.

Each occurrence of George Washington was replaced by aName and each occurrence of a date was replaced by aDate. The resulting lines were sorted by frequency of occurrences. Below is a sample of the top extraction patterns:

- aName was born on aDate
- aName's Birthday aDate
- aName (aDate)
- aName, born on aDate
- aName, (aDate-aDate2)
- aName, born aDate
- aName: aDate
- aName's aDate birthday
- aName was born aDate
- On aDate, aName was born.
- aName's Birthday (on aDate)

We then proceeded to implement the following algorithm to make use of the extraction patterns. First, we extracted a window of text centered about each date. Dates after 1990 were discarded as not possible birth dates of employees. The size of the window was estimated to be 20 words to the left and right. Using the Microsoft Live search engine, we tracked the number of hits for each phrase with no more than 40 words between George Washington and his birth date. This search engine's proximity feature does allow the terms name and birth date to occur in either order. The chart below shows that the number of hits levels off after about 20 words.

Note that the data is slightly noisy. If it were completely accurate, the number of hits should be strictly non-decreasing, and it is not. Approximate searching can be kludged together using Google's asterisk feature inside of quoted strings. For example, “George Washington * * * * February 22, 1832” should return the documents where the two terms are exactly separated by four words. Unfortunately, this feature used to work but now returns no results.

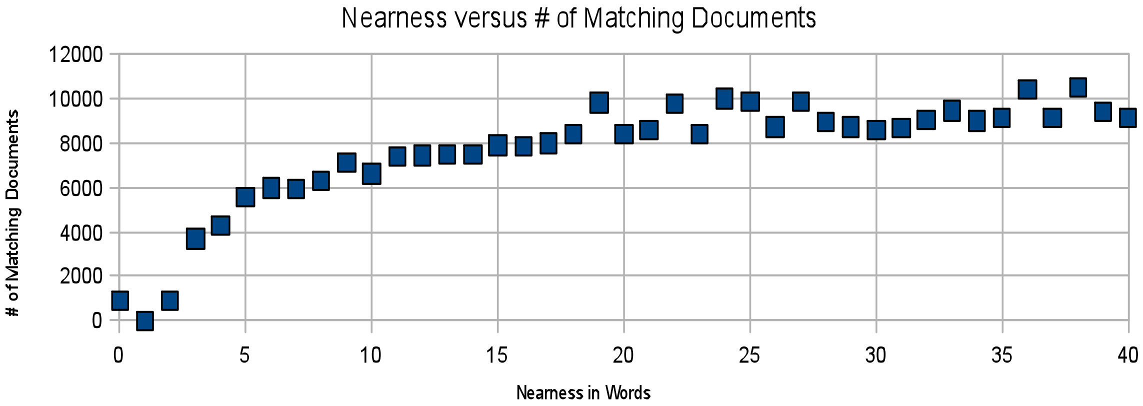


Figure 7.1 Hits versus Span between name and birth date

Spans of about 20 words are the maximum between name and birth date that refer to each other. If you search for windows longer than 20 words, you'll find that your search crosses sentence boundaries. We later simplified the window size calculations to be 300 characters to the left and right of the date. For each window, we attempted to match an extraction pattern. If we found a match, we stopped trying other extraction patterns and reported a found birth date. We found that many of the missed extraction patterns were based on failed matches of personal names. Correctly matching personal names proved to be just as difficult a task as finding birth dates if not more so.

At this point we decided to relax the matching of personal names requirement. After all, for redaction purposes only the date needs to be blotted out, and not the name. We then also relaxed the extraction patterns to simply be that the window contains at least one of the following keywords: 'D.O.B.', 'DOB', a word starting with 'birth', or the word 'born'.

CHAPTER 8

EFFECTS OF OCR ERRORS ON INFORMATION EXTRACTION USING PRECISE SHALLOW PARSING

Blind Testing of DOB Extractor Program

The extractor program was developed in two successive versions. For the first version, we built an extractor using web documents for well known people and their birth dates. Still these documents do not provide examples of all the possible text patterns for expressing DOB that we will see in the Yucca Mountain data. The focus here was making sure we did not miss any birth dates, in other words no false positives. For the second version, we added to the extractor program after studying a sample of 3l documents and emails containing Privacy Act relevant date of birth information. The focus here was relaxing the constraints of the extraction patterns without introducing any false positives. Human experts, different from the developers of the program, made the relevancy judgments. All the data was actual business data from the Yucca Mountain Project delivered to the Information Science Research Institute. A much larger and unseen data set consisting of 1075 documents was used for the blind testing.

Precision and Recall Results

Table 8.1 Precision and Recall Results on 1075 Clean Text Documents

Per-Document		Per-Hit	
tp: 67	fp: 3	tp: 215	fp: 32
fn: 9	tn: 997	fn: 91	tn: 997
precision: 95.71%	recall: 88.16%	precision: 87.04%	recall: 70.26%

Table 8.2 Precision and Recall Results on 1075 OCRed Documents

Per-Document		Per-Hit	
tp: 64	fp: 3	tp: 180	fp: 1173
fn: 12	tn: 997	fn: 128	tn: 997
precision: 95.52%	recall: 84.12%	precision: 13.30%	recall: 58.44%

The per-hit precision dropped in OCRed documents due to the misrecognition of a table of dates for a table of birth dates. Tables of dates are typically recognized as "all or none", meaning the extraction works the same for each date. Tables are somewhat of an oddity that should probably be tackled as its own separate problem. Fortunately, they make up only 12% of dates in the documents, so we can still get useful results without much custom tailoring to tables.

Making the Patterns Robust to OCR Errors

About one-third of the extraction errors on the OCR collection were due to misrecognized words such as "date of birth" being OCRed into "duty of birth", and “BIRTH” being OCRed into “9RTH”. If we consider the top 100 of all OCR-confusions, and apply just the top one or two confusions to each keyword, we typically only generate one or two additional keywords. This makes sense given that our keywords are rather short, at most eight characters. If we add these OCR-generated words to the keywords list, the extraction process would survive the OCR errors that occur inside of keywords.

Another third of the extraction errors occurred when a date was not recognized at all because it has at least one OCR error. Applying the same recovery technique as above would not work because of the very large number of strings that can represent slightly OCR-corrupted dates. We identified the dates using a very large (500 characters long) regular expression that allowed for dozens of different formats. We can relax the conditions on this regular expression using what is known as approximate (fuzzy) regexp. Fuzzy regexp was first implement by Wu and Manber (1992) in the agrep program. Laurikari (2009) has released a free implementation called TRE which allows for regular expressions of any length and more detailed specification of the fuzziness. The fuzziness is specified using edit-distance measure (also known as the Levenshtein distance) where characters can be inserted, deleted, or substituted in the searched text in order to get an exact match on the regular expression. For example, the “bom” can be transformed into “born” using one delete of the m, and two insertions, one for r and another for n. Assuming equal weighting, we can specify an edit distance of three in order allow for this one OCR error during matches.

The last thirds of the OCR errors were due to zoning distortions. This is partially due to tables. Consider a department’s employee list in two column format with field labels like name and date of birth. On the left we have actual names and on the right specific dates. If the OCR engine zones the list into separate columns, the window of text near "date of birth" would now become many characters away, and perhaps outside of the 300 character limit. If we increase the window size to that of a typical whole page, we would avoid the zoning distortions. However, we do increase the likelihood of finding other non-birthdate dates in the window. To what extent we've increase false negatives, would have to have to be measured. In all likelihood, that would cause almost any document with a table of dates and names to be labeled as containing date of birth information. Zoning problems have been previously identified as significantly degrading information extraction of privacy data in general, not just date of birth information (Taghva, Beckley, & Coombs, 2006).

Another approach to handling degraded OCR searching is described by Harding, Croft, and Weir (1997). This is a specialized form of query expansion based on using 2-grams, 3-grams, and 5-grams. Using only partial words inside of extraction rules represented by the n-grams would potentially increase our false positives. So for example, we could make “bor” one of the keywords needed near a date to trigger an extraction. Similar to what Harding did, we can corrupt whole keywords using character recognition confusions typical of OCR systems, including the inclusion and deletion of spaces in words or phrases. Using n-grams would be ideal in the most severely corrupted text where there might be more than one corruption within a window. However, this technique would not work for extracting non-specific dates in a very large number of formats. The amount of actual strings represented by corrupted possible dates is very large. Estimating 50 different formats, 365 different values, 20 different OCR variants gives nearly a half millions strings. Searching for this many strings in parallel requires a very different implementation that those focused on a few patterns with a few errors. This approach is still very feasible. In another unrelated project, we have implemented and performance tested an Aho-Corasick (1974) automaton in Java on a million strings, each about 20 characters in length, and found the pattern matching to be O/I bound.

CHAPTER 9

METHODS FOR INFORMATION EXTRACTION USING FUZZY SHALLOW PARSING

Introduction

Federal agencies, state governments, and local agencies are regularly making records available to the public via the Internet or other media. All these records must be scanned for Personally Identifiable Information (PII) prior to public release. The Licensing Support Network is a document repository built by Nuclear Regulatory Commission (2010) for public review. The LSN is a large document repository of over forty million pages that will provide information to the proceedings for licensing Yucca Mountain's nuclear waste repository. A typical example of PII is the occurrences of individuals and their dates of birth. Other examples are individuals' addresses, phone numbers, fax numbers, email addresses, and credit card information.

Manual review for PII of large collections is not only expensive but also impractical. Knowing that the number of documents to be reviewed for PII is substantial and human review is costly, it is clearly advantageous to seek automatic methods for identifying PII in documents. Since most records are retained in paper form, an automatic solution begins with conversion to an electronic format using OCR. Subsequently, the process would then apply various IE and/or categorization techniques to identify records with PII. Finally, these documents must then be redacted prior to public release.

In this chapter, we describe the development and effectiveness of a fuzzy pattern-based extraction program for identification of Date of Birth PII (Pereda & Taghva, 2011). We found that fuzzy extraction provides high recall for this task. However, we also discovered that fuzzy matching produces a high recall at the expense of a low precision.

We will first describe the problem our information extraction (IE) program is designed to solve in this chapter. We then describe the detailed design of our solution. The next chapter explains the collecting process of the clean and OCRed text. This is followed up the blind testing along with the actual precision and recall numbers.

Extraction Date of Birth Information

Many of the Personally Identifiable Information (PII) patterns are of the form E1 R E2, where E1 and E2 are entities and R is the relationship between these entities. In the Date of Birth example, the relationship R establishes that the entity E2 is the date of birth for the entity E1. The study of patterns of this sort can lead to a partial solution to the PII problem. There are many known techniques for extracting pattern of this form. The most popular methods are a combination of Hidden Markov Model (HMM), language pattern processing, and ad-hoc algorithms (Taghva, Coombs, Pereda, & Nartker, 2005; Brin, 1998; Harding, 1997). Most of the extraction techniques rely on tokens such as “born” or “birth” to establish the relationship. These tokens could have been corrupted during the conversion process using

OCR. Consequently, one has to modify the process to overcome some of the difficulties associated with the OCR errors.

Our method of evaluation for finding date of birth will mirror the standard method in most information extraction experiments. The abbreviation DOB stands for “date of birth”. We use:

$$recall = \frac{\text{number of correct DOB hits}}{\text{total number of DOB's inside the document collection}}$$

and

$$precision = \frac{\text{number of correct DOB hits}}{\text{total number of DOB's reported by the program}}$$

Hits were measured on a line-by-line basis. We choose our document collection so it would contain a high concentration of DOB. In practice, when filtering DOB information it will be much more sparse. There will be fewer DOB hits per line and per document. All but two of the fifty, or 96%, test documents contained DOB information. The density of DOB information for LSN documents will typically be less than 10% of the documents (Taghva, Coombs, Pereda, and Nartker 2005).

Fuzzy Matching and High Recall

The objective of this experiment was to achieve a high recall of over 95% with a reasonable precision of around 30%. We constructed two fuzzy patterns using agrep (Wu & Manber, 1992; Laurikari, 2009) to identify dates and birth indicators. Agrep stands for approximate regular expression pars-

parser. The first pattern is a dash with a number on the left and right, each within 40 characters of the dash. The intention is to capture patterns of the following forms, where the first three are from the clean documents and the second three are from the OCR documents:

1. 1870 -1924
2. (c. 581 - November, 644)
3. (ca. 570/571 Mecca[مَكَّة]/[مَكْة] – June 8, 632)
4. t3lt4ltsl @a. 5701571 Mecca[eS. llli&'] - June 8,632
5. Lnuis Pasteur (l 822- I 895)
6. (t9zg - 1953)

The second pattern is a number with the token “birth” or “born” with 100 characters of the number. The tokens “birth” or “born” can be as much as an edit distance of 2 away from “birth” or “born”. So “bom” is an edit distance of 2 away from born. The intension is to capture patterns of the form:

1. born: Jan 01, 1751
2. he was born -on Feb. 11, 1947
3. Born: August l6, 1769
4. Mendel was bom in Hyncice, Moravia on22 Julyll22in what
5. tsirth: c1400 in Mainz, Germany
6. b, 18 November 1787; d. 10 JulY 1851

Here are the two patterns:

Pattern 1, clean and OCR documents:

[0-9]+.{0,40}[-].{0,40}[0-9]+

Date ranges surrounded with parenthesis are a subset of the date ranges without parenthesis. For this reason, we dropped the parenthesis from this pattern to cover both cases. Both cases occur about equally often for birth dates. The precision is higher when the parentheses are present. Because our main priority for this study is high recall, this is an acceptable simplifying assumption.

Pattern 1 is used on both the clean and OCR documents. For pattern 2 a, the date and keyword can occur in either order. For pattern 2 b, the date must immediately follow the “b.”.

For pattern 2, clean documents:

Pattern 2 a: [0-9]+ AND \b(birth|born)\b

Pattern 2 b: \b[.] [0-9]+

For pattern 2, OCR documents:

Pattern 2 a: [0-9]+ AND \b(birth|born){-1+2#2~2}\b

Pattern 2 b: \b[.,] [0-9]+’

The patterns are specified using POSIX compliant regular expressions. A string of digits is specified as [0-9]+. A comma or period are specified in ‘[.,]’. These two characters were found to OCR confused for one another in the training data. The ‘\b’ denotes a word boundary. All matching was set to be case-insensitive.

Appending approx-settings to the subpattern sets the approximate matching settings for a subpattern. Limits for the number of errors are specified. The count-limits were used to set limits for the number of insertions (+), deletions (-), substitutions (#), and total number of errors (~). For the approx-setting {-1+2#2~2}, the matching allows at most one deletion, at most two insertions, and at most two substitutions. The total number of errors allowed is two. When speaking more broadly, an edit distance of 2 is a rough approximation of the settings used here.

Consider the misrecognized 'bom' for 'born'. Deleting the 'r' and replacing the 'n' with 'm' can transform one word into the other. Because there were two edit operations to repair the error, the edit distance is 2, which is within the limit of 2 for this approx-setting. So '(born){-1+2#2~2}' will match 'bom'. Born is a very short keyword so we can't allow more errors without exponentially increasing the number of possible matches, many of which would lead to false positives. These setting were arrived at with experimentation with matching of word list generated from training data. Using the same approx-setting, 'Birth' will also match 'tsirth' because one word can be transformed into the other with one deletion and one substitution. This was an actual error that was not seen in the training data, but was in the blind testing data.

In some traditional information exaction, dates are matched using a very complex pattern tuned to large collection of clean data. This was the approached used in (Pereda & Taghva, 2010). Producing and tuning this extraction pattern is very specific to the data set and takes much time to fine tune. Someone doing information extraction on OCRed dates would quickly give up using this kind of patterns because dates can have such a wide variety of OCR errors compared to a fixed string of characters for clean dates.

For this study, the assumption that a date always contains at least one number provides a practical matching heuristic. It is conceivable that the numbers were spelled out but that never occurred once in the training data. This heuristic is very useful but not very innovative by itself. When used in combination with approximate regular expressions broken into parts that allow for errors, it is an important piece of the puzzle to IE with in an OCR context.

CHAPTER 10

EFFECTS OF OCR ERRORS ON INFORMATION EXTRACTION USING FUZZY SHALLOW PARSING

Collecting Documents With DOB Information

For this study, documents on the web about people were needed. A popular list of the one hundred most influential people in history is detailed in Hart (1992). Each person's named was googled and the first hit was considered as a candidate document. If the URL had a domain name previously seen then that URL was disqualified and the next document in the search query results was considered. This allowed a more random sampling of sites from the web. If we didn't do this disqualification process, websites like Wikipedia would account for many of the documents and they all have too similar a writing style and formatting.

The first page of each document was printed on a laser printer and scanned using a scanner, model CanoScan LiDE 200 from Cannon. The software package ScanSoft OmniPage SE OCR, provided with the scanner, was used for converting the scanned images into text files. The default scan resolution of 300 dots per inch with grey scale was used for the OCR software. These setting, software, and hardware generated a much higher error rate than state-of-the-art OCR systems. Regardless, the errors are representative of many OCR collections in part because the initial image quality is sometimes very poor and leads to high error rate even with the state-of-the-art OCR software with optimal settings. While the error rate is much higher for the CanoScan system that a state-of-the-art system, the specific individual errors are of the same nature. At the end of this process, one hundred single page OCRed documents were created as a text files.

The generation of the clean text documents was a little less straightforward. The files were saved as text using the web browser Firefox's save as text feature. Some files were in UTF-8, ASCII, and ISO-8859 file formats. Some files had illegal byte sequences, which were manually removed. In some cases where the save text failed, the text was saved using cut and paste from the browser into a text editor. Not all of the HTML was removed from the text using Firefox. In particular, there are vestiges left over from hyperlinks and embedded pictures. It is pretty clear when date of birth matches happened on one of these lines, which happens less than five percent of the time. These matches were not considered matches during the statistics collection of precision and recall.

The line terminators were a mix of Window, Unix, and Mac formats. These were all normalized. This made viewing and verification efforts easier. Files were then manually trimmed to the first page, and parts of columns not visible on the first page removed. In some files, the HTML had no line breaks for the text and line breaks were added. The lines breaks were not matched up with the OCR text line-for-line, as this would take much manual data entry work.

All date of birth matches in both the OCR and clean text were assured to occur on a single line. This was done by manually editing of about five percent of the matches. In a more commercial implementation as done in (Taghva, Coombs Pereda & Nartker, 2005), windows of text centered about key phrases would be used to isolate potential matches rather than examining the text line at a time. Brin (1998) used a similar simplification of ignoring multi-line matches in his relation extraction study. No matches occurred across three or more lines in the original text, so at most a single newline was removed.

Because a dash was used a key matching character, all instances of dashes were normalized to ASCII short dash. Even within the same document before normalization, dashes were encoded with different Unicode sequences yet they might look nearly identical when visually displayed.

Blind Testing on Both Clean and OCRed Text

The DOB extractor was developed in two successive versions. For the first version, we built an extractor using the first 50 clean text documents. Because these web pages were carefully sampled from the entire Web, they represent a broad range of examples of all the possible text patterns for expressing DOB. The focus here was making sure we did not miss any birth dates, in other words no false negatives. For the second version, we added approximate matching limits for keyword based extraction patterns. The focus here was relaxing the constraints of the extraction patterns without introducing so many false positives that the precision drops below 30%. Again only the first 50 documents were used for developing these extracting patterns and arriving at approximate matching limits.

Once the different extraction patterns were developed, they were not allowed to be modified during blind testing. The testing was done the unseen last 50 clean documents and then last 50 OCRed documents. All the relevancy judgments were made on a line-by-line basis. There were roughly 10K lines of clean text and 3K lines of OCR text across the 50 unseen documents. This difference in number of lines is due to the bloating the text when using Firefox's save as text feature. Whitespace resembles the HTML source more than it resembles the whitespace of the HTML rendered.

Precision And Recall Results

Table 10.1
Precision and Recall Results on 50 Unseen Clean Text Documents

Pattern 1		**Pattern 2**	
tp: 37	fp: 16	tp: 41	fp: 1
fn: 0	tn: 10442	fn: 0	tn: 10453
precision: 69.81%	recall: 100.00%	precision: 97.62%	recall: 100.00%

Table 10.2
Precision and Recall Results on 50 Unseen OCR Text Documents

Pattern 1		**Pattern 2**	
tp: 35	fp: 29	tp: 36	fp: 109
fn: 0	tn: 3109	fn: 3	tn: 3025
precision: 54.69%	recall: 100.00%	precision: 24.83%	recall: 92.31%

For pattern 1, the recall remains 100% in the presence of OCR errors. The precision dropped 15.12%. The dash in the date range is never misrecognized in the training or unseen documents. For this pattern, there was never a date that was so poorly recognized that there was not even a single digit. A string of digits makes a good proxy for a date when combined with just a few more details for filtering out many of the false positives. We used the same exact pattern on the clean text and the OCR text. The precision is below 70% even on clean text.

For pattern 2, the recall drops 7.69% in an OCR context. The precision dropped 72.79%. On the clean text we use exact matching of keywords. On the OCR text, we use approximate string matching. We don't exactly predict the OCR errors; we just know that most of the OCR corrupt words will fall near the clean version of the word. We measure distance between OCRed and clean words using the Levenshtein distance, also known at the edit distance.

There are three false negatives in three separate documents that prevented a 100% recall on pattern 2 in the OCR text documents. The first one is attributed to a very small HTML font that caused the space character to be unrecognized.

Clean Text:
Jean-Jacques Rousseau was born to Isaac Rousseau and Suzanne Bernard in Geneva on June 28, 1712
OCR Text: JacquesRousseaurvasborntolsaacRousseauandSuzauneBernardinGenevaon.lune23. 1712.

The second false negative was due to an unrecognized date because not a single digit was recognized correctly.

Clean Text:

Vladimir Lenin was born Vladimir Ilich Ulyanov on April 10, 1870

OCR Text:

Vladimir l,enin was born Vladimir Ilich Ulyanov on April ro, rBTo

For both the two errors above, they could have just as easily have effected pattern 1 because both patterns 1 and 2 use the same date recognition strategy. The last error occurred because the pattern in “b. at Tauresium in Illrium, May 11, 483;” where the date doesn’t not immediately follow the “b.” was not seen in the training data. A larger training data set would have captured this extraction pattern.

Given that this is a relatively small document collection, any changes in precision or recall greater than 5% must be considered significant. Only recall for the pattern 1 remained unchanged with the introduction of OCR errors. There is an inverse relationship between recall and precision. A high recall is being maintained at the cost of a lower precision.

CHAPTER 11

SUMMARY AND CONCLUSION

HMM Information Extraction

Our testing indicates that shrinkage does not significantly improve an HMMs performance for this task. In fact, there was some degradation.

More interesting is the relative "success" of the default HMM. This may suggest that for OCR texts the structure of HMMs should be more flexible than that of the Freitag-McCallum HMM (figure 5.1). For example, perhaps there should be an additional "error state" accessible to all states with a low transition probability that would recognize addresses with "out of order" elements such as in the tagged data of figure 6.1.

For our task, however, the "clean" HMM is preferable to the others. As discussed in at the starts of chapters 5, 7, and 9, correctly finding documents with private information is, to a certain degree, more important than mistakenly marking non-private documents as private. In other words, recall to a certain degree is more important than precision. Human review can make up for the lacking precision without too heavy a cost.

Although Taghva, Borsack and Condit (1996a, 1996b) show that OCR errors do not affect the average effectiveness of information retrieval, some studies have indicated that noisy data can degrade information extraction tasks such as text summarization as reported by Jing, Lopresti, and Shih (2003). Our test results indicate that noisy data will also degrade tasks such as searching a text for segments belonging to some specific category, such as addresses.

There seems to be some "common sense" support for this conclusion. When retrieving OCR documents, a search engine will still have a large proportion of the text available to it due to the high accuracy of OCR engines. However, information extraction requires that specific, localized patterns be found. If the very objects being sought have been lost to the OCR process, extraction will fail. We plan to further study the effects of OCR noise on IE tasks by comparing the performance of HMMs on clean and noisy text collections in various IE tasks.

We also hope to further investigate the variation of certain parameters of HMMs. For example, Freitag suggests a version of absolute discounting different from the one proposed by Borkar, Deshmukh, & Sarawagi (2001), which we used. We would like to determine in later experiments which is superior.

IE using Precise Shallow Parsing

In our applications, the extractors look for establishing a relationship between two entities, namely a person and a date. A date in isolation is not private information. Nor does it become private if it is identified as a birth date with an identifier such as date of birth. A date becomes private information only when it is correctly associated with a person. The date-of-birth identification is an example of relational extraction of the form E1 R E2, where E*i* and R denote an entity and its relation, respectively. Other examples of relational extractions are authors and books are outline in Brin (1998) or acronyms and their definitions by Taghva and Gilbreth (1999).

There are many known techniques for extracting relational information from text. The most popular methods are a combination of Hidden Markov Model, dictionary look up, natural language pattern processing, and ad-hoc algorithms (Taghva, Borsack, and Condit, 1994 &1999; Taghva, Coombs, Pereda, and Nartker 2005). Our approach is an example of an ad-hoc algorithm. Our work implies that identification of documents containing PII can be done using IE techniques for date of birth information on OCRed document collections. The success rate is comparable to that of clean text document collection. With a remaining 5% not recalled, there is still room for improvement. The recall can be improved at the cost of decreased precision by allowing for recognition of page sized tables, and by considering larger training document sets. On a per-hit basis, the OCR errors reduce recall to nearly 50% and precision to nearly 10%, making it not very useful. This is a very challenging problem if one wishes to perform an automatic redaction process.

IE using Fuzzy Shallow Parsing

Again, the objective of this experiment was to achieve a high recall of over 95% with a reasonable precision of around 30%. For pattern 1, this was achieved with 54.69% precision and 100.00% recall. For pattern 2, this was nearly achieved with a 24.83% precision and 92.31% recall. We were within 5.17% of this goal for precision and 2.69% for recall. With some more experimentation with the approx-settings, there is some possibility of improving the precision of pattern 2. The 2 out of the 3 recall errors, or false negatives, could just as easily have occurred during the testing of pattern 1. These errors were due to a very noisy OCR results where dates are unrecognizable when relying on digit characters. This need was not foreseen in the training document set. The date extraction pattern could be improved by expanding the date pattern to include dates without numbers but with the names of months.

The degree of OCR corruption in the unseen OCR documents was extreme. For example, in the date “April ro, rBTo” there are 6 errors in the span of just 14 characters. In a state-of-the-art OCR document repository, such as the LSN, the density of errors per token would be less than half that error rate. In these situations, fuzzy information extraction will meet up with fewer unrecoverable errors, and prove more robust than shown here.

In future studies, we will develop a systematic approach to setting the edit distance error counts for a given pattern. The length of the patterns, the make up of the words within the document set, and the degree of OCR corruption within the collection will play a critical role in determining the optimal error counts. Characterizing the attainable levels of precision and recall will guide IE in an OCR context projects.

OCR Effects on IE

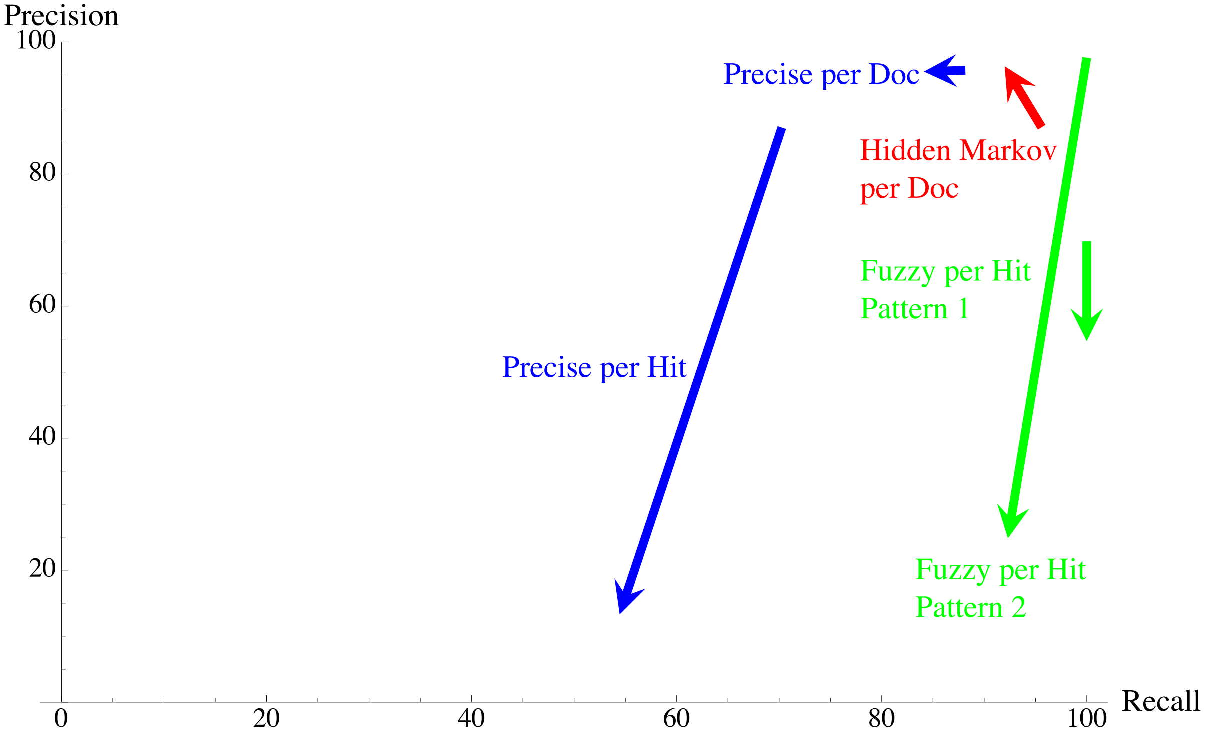


Figure 11.2 Precision Recall Plot of OCR Effects

The above plot summarizes three studies on information extraction in an OCR context. The starting point of each arrow marks the precision and recall on clean text; the end point marks the precision and recall after OCR errors. The first study in red is the on the IE system based on Hidden Markov Models. There is a small decrease in precision but also a surprising small increase in precision. Theoretically, perhaps the HMM generalized better with the noisy OCR training data. This was a complex system and did not easily lead itself to detailed internal analysis of the errors. This could also be a case of cargo cult science (Feyman, 1974 & 1985) where the expected results appear in the data. The blue arrows represent precise shallow parsing. The small blue arrows shows a small decrease in recall after the introduction of OCR errors. The large blue arrow shows a large decrease in precision and a moderate decrease in recall. Here the precision and recall is properly measured per hit, rather than per document. Both the green arrows measure fuzzy shallow parsing IE measured per hit. Pattern 1 matches date of birth patterns with a keyword; pattern 2 matches date of birth patterns with a date range. In one case, we see no decrease in recall and in the other only a slight decrease in recall. In one case, there is a slight decrease in precision and the other a moderate. Roughly, for the shallow parsing techniques, it seems we can trade-off precision for recall when adapting IE systems to OCR errors.

The code for the HMM extractor was written as part of a team of researchers. The code for the shallow parsing programs was written by this author and was, of course, certainly his best effort. The precise shallow parsing extractor was evaluated independently of the author. The results of the three studies are intended to give an empirical gauge of the quality of information extraction results on a large variety documents that have been OCRed.

Conclusions

We've shown that formal extraction methodologies based on the Hidden Markov Models are not robust enough to deal with extraction in an OCR environment. We also shown that both precise shallow parsing and fuzzy shallow parsing can be used to increase the recall at the price of significant drop in the precision. These result are in stark contrast to IR system that function well without modification in an OCR environment.

The Hidden Markov Model system was built by a team that had as many as ten employees. The team included two dedicated data entry workers, a half dozen programmers, two professors, and more. The system was the emperor's new software, like his clothes. The HMM could have been tested more objectively. The shallow parsing system was tested in a very adversarial manner because it was not as sophisticated, and fared well. Now, one can dare say that the emperor has no clothes. The hidden structure of this particular HMM turned out to be the directly observable in the linear sequence of lexical tokens. The shallow parsing techniques were much simpler solutions that worked: solved a real problem on real data. Sometimes problem solving and appearing sophisticated are competing goals. Cargo cult science is an easy mistake to make with Hidden Markov Models.

BIBLIOGRAPHY

Aho, A. V. & Corasick, M. J. (1975). Efficient string matching: An aid to bibliographic search. *Communications of the ACM* 18 (6), 333–340.

Baeza-Yates, R. A., & Ribeiro-Neto, B. (1999). *Modern Information Retrieval.* Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.

Bikel, D., Miller, S., & Weischedel, R. (1997). Nymble: a high–performance learning name-finder. *Proceedings of Applied Natural Language Processing-97*, 194–201.

Bizer, C., Heath, T. & Berners-Lee, T. (2009). Linked Data – The Story So Far. *International Journal on Semantic Web and Information Systems (IJSWIS)*, Volume 5(3), 1– 22

Borkar, V., Deshmukh, K., and Sarawagi, S. (2001). Automatic segmentation of text into structured records. *ACM SIGMOD 2001*, 175–186.

Brin, S. (1998). Extracting patterns and relations from the world wide web. WebDB at 6th International Conference on Extending Database Technology, EDBT'98, 172–183. New York: Springer-Verlag.

Brin, S. and Page, L. (1998). The Anatomy of a Large-Scale Hypertextual Web Search Engine. *Computer Networks*, 30(1-7), 107–117.

Callan, P., Croft, B. and Harding, M. (1992) The INQUERY retrieval system. *Proceedings of the 3rd International Conference on Database and Expert Systems Applications,* 78–83.

Delphi Consulting Group. (1991). *Text Retrieval Systems, A Market and Technology Assessment*, Vol. 1-3.

Feyman, R.P. (1974). Cargo Cult Science. *Engineering and Science*, Volume 37:7, June 1974, pp. 10–13. www.lhup.edu/~DSIMANEK/cargocul.htm.

Feyman, R.P., Leighton, R. & Hutchings, E. (1985). *Surely You're Joking, Mr. Feynman!: Adventures of a Curious Character.* W W Norton, pp. 338–346.

Freitag, D. & McCallum, A. (1999). Information extraction with HMMs and shrinkage. *Proceedings AAAI-99 Workshop Machine Learning and Information Extraction*, 31–36.

Harding, M., Croft, W. B., & Weir, C. (1997). Probabilistic retrieval of OCR degraded text using N-grams. *European Conference on Digital Libraries*, 345–359.

Hart, M. H. (1992). *The 100: A Ranking of the Most Influential Person in History.* Carol Publishing Group/Citadel Press, New York, vii–x.

Leek, T. (1997). Information extraction using hidden markov models. Master's thesis, UC San Diego.

Laurikari, V. (2009). TRE – a lightweight, robust, and efficient POSIX compliant regexp matching library with approximate (fuzzy) matching. Retrieved from http://www.laurikari.net/tre/.

Jing, H., Lopresti, D., & Shih, C. (2003). Summarizing noisy documents. *Proceedings of SDIUT 2003*, 111–119.

Navarro, G. (2001). A guided tour to approximate string matching. *ACM Computing Surveys* 33 (1), 31–88.

Nuclear Regulatory Commission. (2010). The Licensing Support Network – a document repository. Retrieved from http://www.lsnnet.gov/.

Open Text. (1999). Open Text introduces livelink activator for basis. http://www.opentext.com/news/pr.html?id=1055.

Tesseract. (2010). Tesseract (software) an open-source OCR system. Retrieved from http://en.wikipedia.org/wiki/Tesseract_(software).

Rabiner, L. & B. Juang, B. (1993). *Fundamentals of Speech Recognition*. Prentice Hall.

Salton, G. (1989). *Automatic Text Processing*. Reading, MA: Addison-Wesley.

Salton, G & Buckley, C. (1988). Term-Weighting Approaches in Automatic Text Retrieval. *Information Processing and Management, 24*, 513–523.

Pereda, R. & Taghva, K. (2010). Date of birth extraction using precise shallow parsing. *Proceeding IS&T/SPIE 2010 International Symposium on Electronic Imaging Science and Technology*, 7534–05.

Pereda, R. & Taghva, K. (2011). Fuzzy Information Extraction on OCR Text. *Eighth International Conference on Information Technology: New Generations*, in press.

Taghva, K., Cartright, M., & Borsack, J. (2004). An efficient tool for xml data preparation. *Proceedings of Information Systems: Next Generation*.

Taghva, K., Beckley, R. & Coombs, J. (2006) The effects of OCR error on the extraction of private information. *7th IAPR Workshop on Document Analysis Systems* (DAS 2006), volume 3872 of Lecture Notes in Computer Science, 348–357.

Taghva, K., Borsack, J., & Condit, A. (1994). Results of applying probabilistic IR to OCR text. *ACM SIGIR Conference on Research and Development in Information Retrieval*, 202–211.

Taghva, K., Borsack, J., & Condit, A. (1999). The effectiveness of thesauri-aided retrieval. *Proceeding of IS&T/SPIE 1999 International Symposium On Electronic Imaging Science and Technology*, 202–211.

Taghva, K., Borsack, J., & Condit, A. (1996a). Effects of OCR errors on ranking and feedback using the vector space model. *Inf. Proc. and Management*, 32(3), 317–327.

Taghva, K., Borsack, J., & Condit, A. (1996b). Evaluation of model-based retrieval effectiveness with OCR text. *ACM Transactions on Information Systems 14*, 64–93.

Taghva, K., Borsack, J., Condit, A., & Erva, S. (1994). The effects of noisy data on text retrieval. *Journal of the American Society for Information Science 45*, 50–58.

Taghva, K., Coombs J., Pereda, R., & Nartker, T. (2005). Address extraction using hidden markov models. *Proceeding IS&T/SPIE 2005 International Symposium on Electronic Imaging Science and Technology*, 119–126.

Taghva, K. & Gilbreth, J. (1999). Finding acronyms and their definitions. *International Journal on Document Analysis and Recognition 1(4)*, 191–198.

United States Department of Justice. (1974). *The Privacy Act of 1974*. Retrieved from http://www.justice.gov/opcl/privacyact1974.htm.

Wu, S. & Manber, U.. (1992). Agrep – a fast approximate pattern matching tool. *Proceeding of USENIX Technical Conference*, 153–162.

VITA

Graduate College
University of Nevada, Las Vegas

Ray Pereda
Email: raypereda@gmail.com

Degrees:

Bachelor of Science, Mathematics, 1991
Bachelor of Science, Computer Science, Department Honors, 1991
Minors in Philosophy and Linguistics
University of New Orleans, University Honors

Master of Science, Computer Science, 1993
University of Arizona

Master of Science, Complex Adaptive Systems, 1997
National Technological University

Master of Science, Computer Science, 1999
University of Texas

Dissertation Title:

Information Extraction in an Optical Character Recognition Context

Dissertation Committee:

Chairperson, Kazem Taghva, Ph.D.
Committee Member, Thomas Nartker, Ph.D.
Committee Member, Laxmi Gewali, Ph.D.
Committee Member, Ajoy Datta, Ph.D.
Committee Member, Ashok Singh, Ph.D.