1-1-2007

Document boundary determination using structural and lexical analysis

Marc-Allen Cartright
University of Nevada, Las Vegas

DOCUMENT BOUNDARY DETERMINATION USING STRUCTURAL AND LEXICAL ANALYSIS

by

Marc-Allen Cartright

Bachelor of Science
Stanford University
2002

A thesis submitted in partial fulfillment
of the requirements for the

Master of Science Degree in Computer Science
School of Computer Science
Howard R. Hughes College of Engineering

Graduate College
University of Nevada, Las Vegas
August 2007

UNLV
UNIVERSITY OF NEVADA LAS VEGAS

Thesis Approval

The Graduate College
University of Nevada, Las Vegas

JUNE 5TH , 2007

The Thesis prepared by

MARC-ALLEN CARTRIGHT

Entitled

DOCUMENT BOUNDARY DETERMINATION USING STRUCTURAL AND LEXICAL ANALYSIS

is approved in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

Examination Committee Chair

Dean of the Graduate College

Examination Committee Member

Examination Committee Member

Graduate College Faculty Representative

1017-53

ABSTRACT

Document Boundary Determination Using Structural and Lexical Analysis

by

Marc-Allen Cartright

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

A method of sequentially presented document determination using parallel analyses from various facets of structural document understanding and information retrieval is proposed in this thesis. Specifically, the method presented here intends to serve as a trainable system when determining where one document ends and another begins. Content analysis methods include use of the Vector Space Model, as well as targeted analysis of content on the margins of document fragments. Structural analysis for this implementation has been limited to simple and ubiquitous entities, such as software-generated zones, simple format-specific lines, and the appearance of page numbers. Analysis focuses on change in similarity between comparisons, with the emphasis placed on the fact that the extremities of documents tend to contain significant structural and lexical changes that can be observed and quantified. We combine the various features using nonlinear approximation (neural network) and experimentally test the usefulness of the combinations.

TABLE OF CONTENTS

ABSTRACT iii

LIST OF TABLES vi

LIST OF FIGURES vii

ACKNOWLEDGMENTS viii

CHAPTER 1 INTRODUCTION 1

CHAPTER 2 RELATED WORK 4
The Lexical Approach 5
The Structural Approach 6
The Crossroads 7

CHAPTER 3 DATA DESCRIPTION AND METHODOLOGY 9
Formal Description of Data 9
Methodology 12

CHAPTER 4 ANALYSIS OF RESULTS 32
Performance Metrics 32
Experimental Data 33
Experimental Results 35

CHAPTER 5 CONCLUSIONS AND FURTHER THOUGHTS 40

BIBLIOGRAPHY 42

APPENDIX A VARIOUS EQUATIONS AND METRICS 45
Document Representation 45
TF*IDF Model 45
Term Count Model 45
Vector Cosine 46

Precision and Recall 46

APPENDIX B EXAMPLE OF THE THRESHOLDING METHOD 48

APPENDIX C SUPPLEMENTAL DATA 50
Alternative Network Performances 50
Baseline Data 51

VITA 56

LIST OF TABLES

1 Attribute values and possible values. 15
2 Attribute values and associated translations. 26
3 Our $K = 10$ subsample sets. 34
4 Baseline performance for 6-fold removed training set. 35
5 Best-performing network. 35
6 $\Delta - Learn$ network performance. 37
7 The 8-10-4-1 ($Hidden+$) configuration performance. 38
8 Short example of statistical errors 47
9 An example of boundaries and their corresponding vector cosine values. . . 48
10 Threshold approximation. 49
11 Performance of the 8-3-1 configuration. 51
12 Performance of the 8-5-1 configuration. 52
13 Performance of the 8-10-1 configuration. 52
14 Baseline Performance for 1-fold removed training set. 52
15 Baseline Performance for 2-fold removed training set. 53
16 Baseline Performance for 3-fold removed training set. 53
17 Baseline Performance for 4-fold removed training set. 53
18 Baseline Performance for 5-fold removed training set. 54
19 Baseline Performance for 6-fold removed training set. 54
20 Baseline Performance for 7-fold removed training set. 54
21 Baseline Performance for 8-fold removed training set. 55
22 Baseline Performance for 9-fold removed training set. 55
23 Baseline Performance for 10-fold removed training set. 55

LIST OF FIGURES

1 Coordinate layout for pages. 11
2 How a single problem instance is presented to the classifier. 14
3 Geometric representation of documents. 16
4 Example of eligible header annd footer data. 18
5 Finding center points and mid line for zone partitioning. 20
6 Two pages being compared for zone count differences. 22
7 Lines captured by the Hough Transform. 25
8 Determining the error from our threshold. 27
9 A example neural network classifier with one hidden layer. 29

ACKNOWLEDGMENTS

I would like to acknowledge and thank the patience and support provided by the wonderful members of the Information Science Research Institute. Professors Kazem Taghva and Tom Nartker both allowed me to progress in my Master's degree throughout my time at ISRI. I can also credit them with providing the seeds of my research topic, even if it was not always pursued with the utmost firm and vigor.

I have learned an enormous amount during my time with my colleagues there, and am thoroughly grateful for providing me with a place to work and study without having to log on the campus cluster. Many thanks to all of you.

CHAPTER 1

INTRODUCTION

In recent years, the ability to digitize physical documents has increased dramatically. It is becoming more commonplace for large organizations to place as much of their paper documents online as possible. Scanning documents into image format and making then available online has largely been addressed, however more often than not, owners of those docments want to be able to organize them and search them efficiently as well. The data must not only be scanned, but this legacy data must also have meta data included to provide information for organizational and retrieval purposes. The process of annotating this legacy data, while yielding great benefits in the areas of retrieval and searching, is generally tedious and riddled with errors. Typically human intervention is needed in multiple steps of the process. The most common digitization process involves scanning an entire collection in one large batch. This process still requires significant preparation by human hands. Just to prepare a collection to be scanned requires that someone to remove all physical bindings between pages, and then the boundaries between documents must be manually determined by visually scanning each of the pages in the collection. This particular task has come to be known as document boundary determination. While humans typically excel at one instance of such a task, we quickly degenerate into boredom when faced with the same scenario many times. Eventually a human reviewer will introduce errors into the process, even more so as the process continues. A solution to both the typical bored reviewer and the process as a whole, would be to delegate

such a repetitive task to a computer.

Such a solution is clearly not that simple; otherwise we would have automated this process years ago. Computers are good at discrete, repeatable tasks, such as calculations. However even a task like boundary determination between two documents, which we would consider laughably simple, involves a developed level of understanding about document content and structure that computers are currently unable to achieve. Documents almost never fall into discrete categories, nor are they confined to a standard for formatting or content. They often contain images that cause the document to further deviate from any established pattern of formatting or content. Images also convey information that is subsequently omitted from the printed text. Even adults, who generally are very good at this task, rely on years of learned reasoning skills to make the correct determination.

So, what chance does a simple computer have against solving a problem that seems to require years of training and higher-level reasoning? Fortunately, some progress has been made, even if it has been in a fashion analogous to "spoon-feeding" the machines the relevant understanding we use to determine document boundaries. The most successful approaches to the problem to date have shown to be quite effective (over 95% accuracy for Collins-Thompson and Nickolov [9]), however these approaches tend to focus solely on content analysis or structural analysis only, and completely disregard the host of information contained in the other approach. We believe that a practical solution to this problem should combine both types of features, to take advantage of as much information as possible in each document. Ideally, we would rely on aspects of the data in question that tend to be universal, which would mitigate the need for specialized niche systems. Such a task will require a large amount of evolution before it becomes ubiquitious; but it seems that the path to realistically automate such tasks lies through all available avenues.

In this thesis we present and test a hybrid analysis method that could serve to automate of the process of document boundary determination. The analysis makes use of both classical information retrieval features as well as structurally relevant features. The features are then used collectively as inputs to a neural network classification system that produces a final boundary determination prediction. The classifier is trained on a set of examples and then we test its performance.

The remainder of the thesis is organized as follows. Chapter 2 presents a review of current research and literature on the topic of document understanding, in both content-driven and structurally-driven methods. Chapter 3 creates a formal presentation of the problem, and then discusses the methodologies implemented in the analysis of the data. Chapter 4 contains experimental results followed by a critique of the results. We conclude the thesis with a discussion of expansion of the concept and possible future work in Chapter 5.

CHAPTER 2

RELATED WORK

Automatic document boundary determination falls in the realm of document understanding; we need our system to have enough information to correctly determine when one document ends, and when the next one begins. However, the information retrieval community is just now getting to the point of considering automatic boundary determination an issue that can be addressed. A multitude of innovation and research has taken place over the years, but most techniques are only tangentially pertinent to this problem. A historical review will provide more understanding of the problem, and what challenges we can expect in facing it.

The large-scale issue of document understanding has been well understod for years [28]; by early 1978, the International Association for Pattern Recognition (IAPR) had been established to provide a centralized organization for researchers concerned with pattern recognition problems to frequently meet and present their research. A host of various conferences and journals have grown out of this organization, among them the International Conference on Document Analysis and Recognition (ICDAR). This conference now frequently convenes (the 9^{th} conference convenes in September of 2007) to specifically address the problem of document understanding.

Approaches to document understanding fall mainly into two categories: lexical analysis and structural analysis. Lexical analysis uses the words and language used in a document to gain an understanding of the content of a document. Structural analysis uses layout, formatting, even font-size and style as the basis for document

understanding. Both approaches have developed concurrently in research, however they have also done so mostly in isolation of each other.

The Lexical Approach

Lexical analysis, also known as content-based analysis, best serves document understanding tasks that depend on knowing what the topic of discussion is in the documents, such as text categorization, information retrieval, and information extraction.

The Term Count Model was one of the earliest document representation models in use in information retrieval, but was prey to several frequently-occurring weaknesses. The successor to the Term Count Model, the Vector Space Model (VSM), was presented in [23] by Salton and remains one of the most popular methods of representation for information retrieval. VSM incorporates collection-wide information on the terms, something that was lacking representation in the earlier model. Another technique known as language models also arose as a highly-successful analysis method. Language models represent a document as a series of joint probabilities based on the occurrences of the terms found in the document. After some time, however, statistically it was shown that both the Vector Space Model and Language Models share a high degree of similarity in their representative power [12].

The advent of large-scale search engines created a need for richer metadata to aid in information retrieval. This consequently increased the need to be able to create the metadata necessary for many developed techniques to function efficiently. Information extraction (IE), the task of extracting structured data from an unstructured data source, has gained a large amount of momentum from this demand. The most effective methods have found success in the use of statistically-driven methods, more popularly known as machine learning. Systems are created that gather statistical information concerning the colletion of interest; in some instances, they also contain features

tailored for that specific collection as well. One of the proven methods for gaining document understanding involves Bayesian networks for classification [16, 26, 27].

Neural networks have also seen successful use in text classification. In [20], a neural network classifier was trained and used in identification of IRS forms. Jeschke and Lalmas [14] combined a neural network classifier with belief values generated using the Dempster-Shafer Theory of Evidence [25].

In 1990, Rabiner published a tutorial on Hidden Markov Models (HMMs) and their application in speech recognition [21]. The work quickly became a standard reference, and made using such a structure much more approachable to the IR community. T. Leek also showed a relatively simple yet highly effective application of HMMs in extracting medical information [17]. Bikel et al. targeted searching for names specifically using an HMM/n-Gram hybrid and achieved a moderately high amount of success [6]. Seymore, McCallum, and Rosenfeld [24] were able to even created a learing algorithm to determine the optimal structures for the HMMs they used over their target collections; they consequently used the generated Hidden Markov Model to extract information from their target collections. A similar technique was used to generate a back-off HMM using data from sparsely populated datasets [10].

The Structural Approach

Structural analysis in document understanding relies heavily on being able to correctly identify visual cues in the images of scanned documents. The problem was first viewed almost solely as a pattern recognition task, however the lack of standards in potential data made such approaches difficult, since they inherently had limited adaptability. As research continued, the consensus has been to adopt statistical methods to gather the information [5]. A variety of approaches have been developed over the years, some of the most popular being connected-component analysis [19], wavelet

decomposition [11, 1], smearing [7], and geometric transforms [13, 8]. Most of these approaches involve heavy reliance on mathematical processing to discover patterns in the images; consequently this approach has drawn some attention from electrical engineers who specialize in signal processing (the wavelet decomposition approach evolved in this manner). A rather comprehensive analysis on almost all perceivable structural features was conducted by Bagdanov [3], in which such elements as font style, size, and layout are painstakingly gathered and analyzed. He developed templates for classification using training examples in his datasets, all of which was driven heavily by probabilistic methods. In 2004, a group from Lehigh University recognized the growing need for commerically viable methods, and began the search for broad-spectrum robust methods of document image understanding [4].

The Crossroads

More recently, there have been some inroads specifically into automated document boundary determination. Thompson and Nickolov created a support vector machine-based system to determine boundaries in document batches [9]. Their approach decidedly revolves more heavily around the anaylsis of content than of the structure of the documents, however some features are derived from a structural approach. Their system achieved fairly high accuracy (upwards of 95%). Very recently, Xerox RCE has also taken an interest in researching the document boundary determination problem (termed "document separation")[1]. Several organizations now offer this process as part of a full-blown document digitization service, although none have yet claimed to be able to implement this process in an automated fashion.

This brings us to our current situation. The problem of automatic document

[1]http://www.xrce.xerox.com/internships/JMR.AlgoTextIntensiveDocSep.2007.html, as of June 2007

separation has existed since the first set of documents was scanned, but to even think of addressing the problem, we need to gather a varied body of knowledge about the documents of concern. A combinational approach to this problem is presented here; several pre-existing methods are used in conjunction to construct a mosaic of data. This collection of diverse data is then translated into values appropriate to feed into a trained binary classifier that is based on a neural network design. We choose a neural network design because the model has an inherent adaptability to new uses. Other uses of neural networks include image recognition [2] and even autonomous vehicle navigation [15]; our use of the model, while novel, is not the most extreme ontological leap when compared to such previous implementations. Ultimately, we hope to see the whole of the system perform the task better than any one of its individual components.

CHAPTER 3

DATA DESCRIPTION AND METHODOLOGY

In this chapter we explore the various features that will be analyzed for the experiment. To understand the methods employed, however, we must first establish a solid representation of exactly what we are analyzing. We begin by formalizing the data we wish to analyze, and then proceed to describe the process used in our attempt to address the problem. We continue by describing the chosen features that will be used, how they are gathered as well as interpreted to be meainingful, and finally how they are combined to make a judgment for a particular instance of our problem.

Formal Description of Data

The simplest description of our data is "a continuous collection of scanned pages", where "page" takes on the typical meaning of a standard piece of paper containing printed text and possibly other information, such as images or graphs. We allow ourselves a slight abuse of language, and note that the phrase "document fragment" and "page" may be used interchangably. The term "document" here assumes the generally accepted interpretation. Our first observation is that the collection of pages contains somewhere between one full document and as many documents as there are pages (each page is a distinct document). We also assume the pages in each document were scanned in the order in which they were originally set when the document was produced. In a stricter fashion, suppose we have N pages in our collection. Let p_{ij}

indicate the j^{th} page in the i^{th} document. We consider the collection to be ordered, so each page will also have a collection-wide ordinal, indicated as p^k where $1 \leq k \leq N$. The two ordinal systems will also be simultaneously applied to a particular page. Therefore, p_{ij}^k refers to the k^{th} page in the collection, as well as the j^{th} page in the i^{th} document. Notice that any p^k where $1 \leq k \leq N$ uniquely determines a specific p_{ij} in the collection. Note that this mapping is a bijection, so any p_{ij} uniquely determines a p^k as well. We will operate with two assumptions that create what we call document cohesion. The two assumptions made here are:

1. The ordering of pages within a document is preserved throughout the collection.

2. The pages of a document are continuous throughout the collection.

Symbolically, we may say it as

1. For any p_{ij}^k, p_{il}^m, if $j < l$, then $k < m$.

2. For any p_{ij}^k, p_{il}^m, and $j < l$, there does not exist a p_{rs}^t such that $r \neq i$ and $k < t < m$.

It would also serve to provide a spatially relevant understanding of each page in the collection, as we will be analyzing each page according to their visual presentation as well as their content. First, we define a Cartesian coordinate system on the image of each page. The upper-left corner of the page is defined as the origin of our coordinate system, with movement right being the increasing 'x' direction. Movement in the downward direction translates into movement in the increasing 'y' direction, as shown in Figure 1. Each pixel in the image translates into a unit in the coordinate space. Notice that each term can also be visually defined by its bounding box. The bounding box is the smallest rectangle in which all pixels in the term can be encompassed visually.

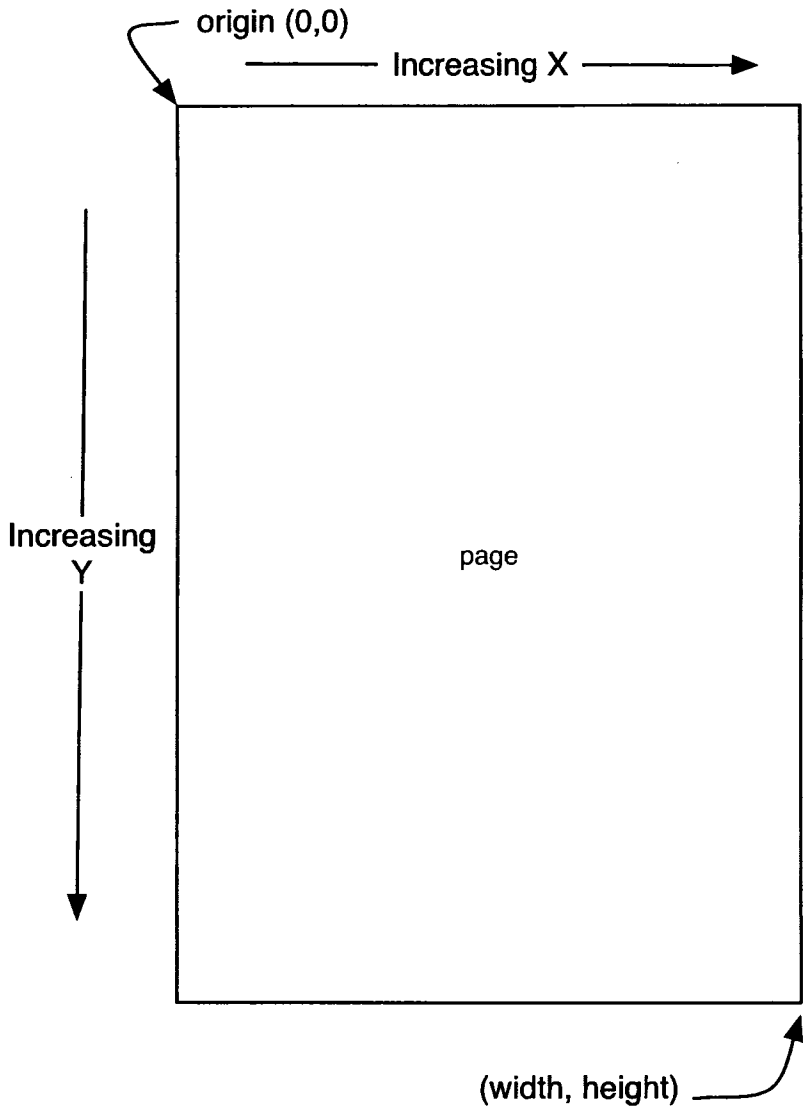

Figure 1: Coordinate layout for pages.

Now that we have a way of formally describing our data, we can also formally describe the problem we are trying to solve. For any given p^i and p^{i+1} in our collection, we would like to know if they belong to different documents. Let D_j refer to the j^{th} document in the collection, and let $b^{i,j}$ indicate the possible boundary between pages p^i and p^{i+1} :

$$b^{i,j} = \begin{cases} 1 & \text{if } p^i \in D_j \text{ and } p^{i+1} \in D_{j+1}, \\ 0 & \text{if } p^i, p^{i+1} \in D_j. \end{cases} \tag{3.1}$$

Note that with the assumptions made above, these are the only possible conditions, and therefore the only two values $b^{i,j}$ can assume. Let

$$\boldsymbol{B} = \{\forall i, 0 \leq i < N : b^{i,i+1} \text{ where } b^{i,i+1} = 1\} \tag{3.2}$$

The set $\boldsymbol{B}$ represents all of the actual document boundaries between all pages in our collection. Ideally, what we would like is to find all of the members of set $\boldsymbol{B}$. More realistically, what we want is to find as many members of set $\boldsymbol{B}$ as possible. This formulation is essentially a classification problem; we have a set of problem instances that fall into one of some number of categories. In our case, we have 2 categories: either our instance 1) belongs to the set of document boundaries $\boldsymbol{B}$, or 2) it does not. Now that we have fully defined our problem, we describe our approach to addressing it.

Methodology

Referring back to the set notation for $\boldsymbol{B}$ defined earlier, we want to find as many members of set $\boldsymbol{B}$ as possible, and we believe that in using information gained from various aspects of the data in question, we can perform this task better than if we simply use only one aspect. Therefore, our inputs consist of the various features, or "attributes", of each problem instance, where a single attribute is information gathered using a distinct technique, and provides information about the instance not available from the other techniques. Figure 2 visualizes the entire process. It is important to note that a "problem instance" actually refers to the attributes describing

a possible boundary between pages, and not the pages themselves. Each instance therefore requires two pages in the collection to be compared, producing a set of values that actually describe the degree of similarity or dissimilarity between them, depending on the specific attribute under comparison.

For a single problem instance, we perform the process of gathering, which is compiling all of the attributes for that particular instance. After that, we then translate each of the inputs into an appropriate value for an input to the classifier, which then classifies the instance. In short, the three processes may be thought of as functions:

1. Gathering: Problem Instance → Set of Attributes
2. Translation: One Attribute → One Input to Classifier
3. Classification: Set of Inputs → Classification of Problem Instance

Although Figure 2 describes how a single instance is classified, in the actual implementation, the gathering process is conducted for all problem instances as a preprocessing step, and during training/testing the translation and classification processes are performed on a per-instance basis. We continue by describing the gathering process for each of the attributes, followed by a description of any needed translation on each of the attributes, and finally with a description of the classifier used in the experiment.

Process: Gathering

The process of gathering the attributes which describe a problem instance involves various techniques, each of which provides an exclusive representation of the instance. Several of the attributes described are just variants of the same technique applied to different parts of a page (i.e., header and footer information gathering uses the same technique, but one is applied to the top of the page, the other the bottom).

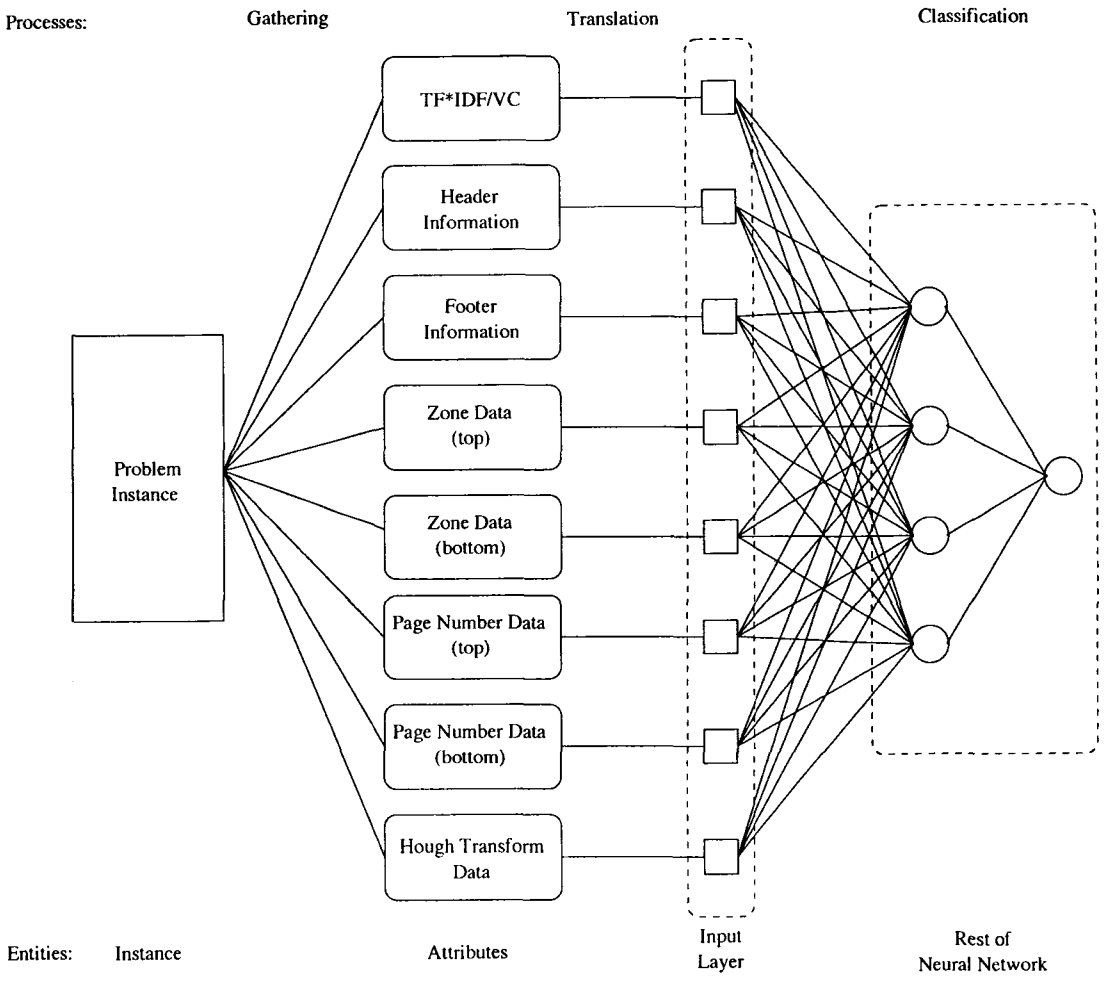

Figure 2: How a single problem instance is presented to the classifier.

These attributes will be described as one in the following sections, our explanation for why they are treated as distinct attributes will also be contained in that section.

To aid in the following discussion of attributes, we review the attributes in question, as well as provide shorthand labels to make it easier to refer to them as variables. Table 1 shows these values.

We use the symbols defined in Table 1 to act as placeholders for possible values that the attribute in question may assume.

Attribute	Symbol	Value Range
TF*IDF/VC	$attr_{TF}$	[0,1] $\subset \mathbf{R}$
Header/Footer	$attr_{HDR,FTR}$	[0,1] $\subset \mathbf{R}$
Page Numbers	$attr_{PN(TOP),PN(BOT)}$	0,1
Zones	$attr_{ZONE(TOP),ZONE(BOT)}$	positive integer
Hough Transform	$attr_{HOUGH}$	0,1

Table 1: Attribute values and possible values.

Gathering: TF*IDF/VC. The term "TF*IDF" is shorthand for term frequency * inverse document frequency, and describes a particular model used to represent the lexicon of a document. The TF*IDF model represents each unique term of the document in question as a value in $\mathbf{R}$, which represents the term weight of that term in that document. This term weight considers both frequency of the term in the document as well as the strength of the term as a discriminating factor across the entire collection.

The "VC" is short for vector cosine, which is the method used to compare two model instances. When the cosine calculation is made, it produces a scalar value in the range $[0, 1] \subset \mathbf{R}$ that represents the degree of similarity between two documents. 0 represents complete dissimilarity, whereas 1 indicates identical documents[1]. The calculation itself involves taking the dot product of the two vectors that represent the term weights of the two documents in question, and dividing them by the maximum possible product of the two documents. Figure 3 shows an example of 3 documents displayed in geometric space. Document is d_1 is more similar to document d_3 than document d_2, because the angle between them is smaller. A more detailed explanation of the TF*IDF model and the vector cosine operation can be found on page 45 .

[1]The location of the terms in the documents is not considered in the calculation. Therefore "My blue cat and her pink poodle are lost" and "My lost pink cat and her poodle are blue" would appear to be identical

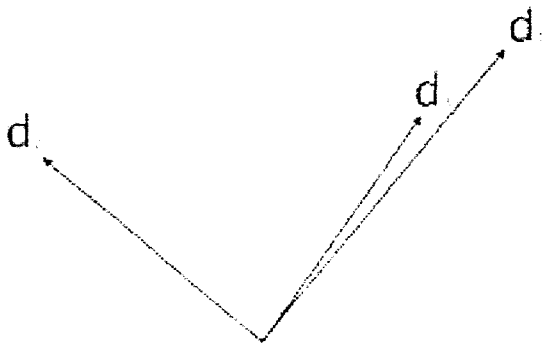

Figure 3: Geometric representation of documents.

Gathering: Header and Footer Information. Many documents contain valuable meta information contained along the top (header) and bottom (footer) margins of the pages. Naturally, this information seems appropriate to look for in our experiment. In [9], the header/footer data was optionally included in analysis in trying to determine document boundaries. The study concluded that the use of header and footer information in the analysis negatively affected their results; however the authors still concede that much of the error was due to unexpected instances in the data, and that header/footer analysis could still serve an important part in boundary determination. We agree with their claim, and choose to add this information for our experiments.

Margin information is determined by first determining a margin limit. We choose a certain percentage of the page to be eligible as margin information. For example, say we were to use 10% as the limit for a page of height 1000. We would first calculate the threshold for the top (header) and bottom (footer) 10% of the page. In this case the top threshold is $y = 100$, and the bottom threshold is $y = 900$. For each term, we calculate the center point of the bounding box of the term, and determine whether the y position of that center point is either above the top threshold (i.e., < 100) or below the bottom threshold (i.e., > 900). If it is, we add it to the list of terms considered part of the margin data. Figure 4 shows a sample page. The shaded sections are the areas of the page that are considered part of the margin data.

We use a different document representation model known as the Term Count Model (TCM) for comparisons of the margin data. The difference between the TCM and the TF*IDF models is the value used as the term weight in the table describing the document. Where the TF*IDF model considers collection-wide statistics in calculating the term weight, the TCM merely uses the number of occurrences of that term in the document. The TCM is widely known to be susceptible to several weaknesses, such as term spamming[2] as well as a bias towards longer documents, which tends to create stronger term weights. However a single page has a discrete size, and since we only consider a fraction of the terms in each comparison, we believe that these concerns are immaterial. Our comparison method is the vector cosine method that was introduced above. Although described in conjunction, the data collected for the header exists separately from the data collected for the footer. This was done to avoid aliasing the two attributes together. A minor example will illustrate the point.

Suppose we do not separate the attributes, and we unwittingly proceed to include a large corpus of training data that happens to have the title of each document printed

[2]Deliberately repeating a term to increase its relevance.

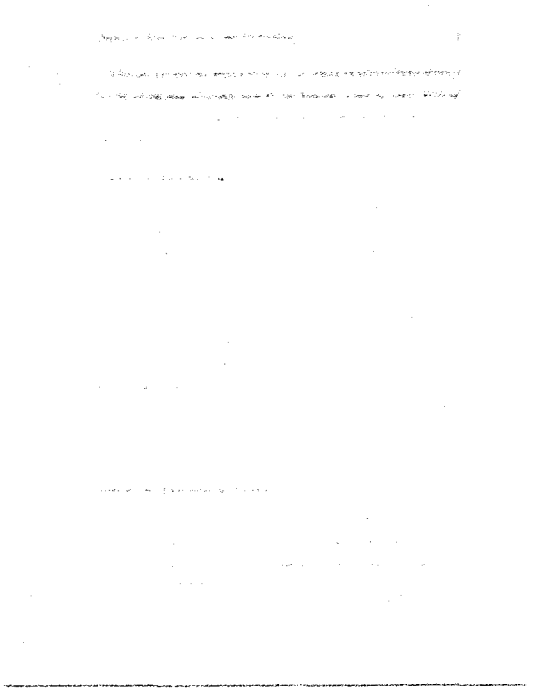

Figure 4: Example of eligible header annd footer data.

across the top of each page in the document, for every document in the collection. Upon training, our classifier will learn to heavily trust our combined header/footer attribute, since it has such a strong representation in the training data, and it can easily make the correct classification in virtually every training instance. Now, when we take the classifier on a test run, it comes across footer information that partially matches, but is not a perfect match. An example would be if the document title and the section were printed together at the bottom of the page. Since the match is not perfect, this attribute will mistakenly consider this instance a document boundary, and provide its now-overwhelming signal to the classifier, possibly now causing the

classifier to incorrectly identify the boundary instance. In order to avoid this scenario, we simply treat the header and footer as separate entities.

Gathering: Zone Data. The zone data we begin with is produced during the OCR process. OCR software has evolved to the point of easily being able to determine distinct zones within the layout of a page. Zone class (i.e., whether the zone contains an image, text, etc.) information was also available from the software, but the method used to assign classes was unavailable, and as such, we consider that information unreliable and omit it from consideration in this thesis.

Each zone is fundamentally a set of points that describe a rectangle in the image that encompasses some piece of information. This carries interest as a feature because it is typical that different documents use different layout styles, and therefore produce a distinct number of zones. No existing, straightfoward method was found that can compare the similarity of the layout between two pages of a document, so we use a very simple approach to implement this feature. For each zone on the page, we determine the center point of that zone (midpoint of length and height of the zone), and then determine whether that zone falls on the top half of the bottom half of the page. Figure 5 illustrates the division of the page. For the purposes of discussion, let $zone_B^i$ be the number of zones in the bottom half of page $page^i$, while $zone_T^i$ be the number of zones in the top half of that same page.

Our attribute values, $attr_{ZONE(TOP)}$ and $attr_{ZONE(BOT)}$, are produced as follows:

$$attr^i_{ZONE(TOP)} = |zone^i_T - zone^{i+1}_T| \tag{3.3}$$

$$attr^i_{ZONE(BOT)} = |zone^i_B - zone^{i+1}_B| \tag{3.4}$$

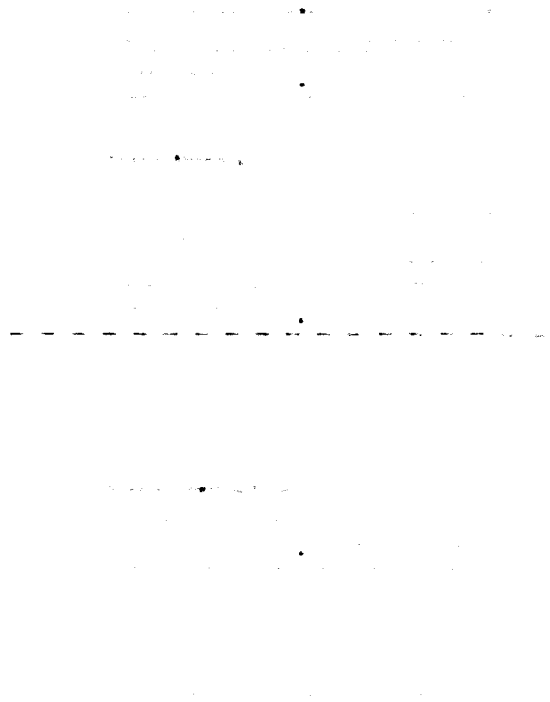

Figure 5: Finding center points and mid line for zone partitioning.

Take the absolute value of the difference between the number of zones from the two pages $page^i$ and $page^{i+1}$. Figure 6 shows two pages ready to be compared. This particular method also requires supplemental values for the attributes; the maximum number of zones between the two pages being compared. Although we will not use these quantities until section 3.2, where we describe translation of this attribute, it is easier to introduce these values in this context. We denote the maximum values as follows:

$$max(attr^i_{ZONE(TOP)}) = max(attr^i_{ZONE(TOP)}, attr^{i+1}_{ZONE(TOP)}) \tag{3.5}$$

$$max(attr^i_{ZONE(BOT)}) = max(attr^i_{ZONE(BOT)}, attr^{i+1}_{ZONE(BOT)}) \tag{3.6}$$

The left-hand quantities are shorthand for the full function, but we reserve the right to refer to this value later on in the processing, as it is produced for this attribute during the phase of the processing.

Like the header and footer attributes, the top and bottom zone attributes are considered separately to avoid a possible aliasing issue. While no evidence exists to substantiate this concern (since no existing comparison methods could be found), in terms of processing it is temporally and spatially trivial to treat the two separately, and only serves to provide finer resolution when analyzing our attributes for the problem instance.

Gathering: Page Number Data. Attempting to track page number information has also shown potential [18]. Such information, if it is present, is easy to obtain, and also tends to follow one of several patterns. This thesis uses several well-known patterns for recognizing potential page numbers. We only search data that has been gathered as part of the margin data, as described above. We also only collect a subset of

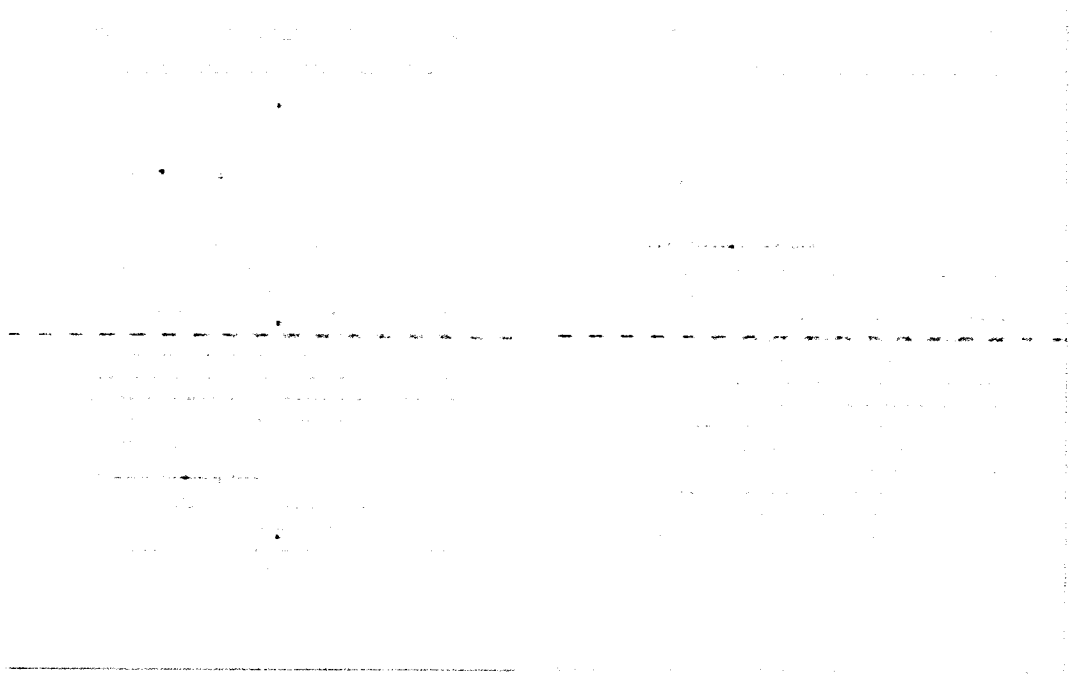

Figure 6: Two pages being compared for zone count differences.

the patterns presented in [18]. Only strings that can be recognized as integers or roman numerals will be captured as features in this thesis. This was done to keep the implementation and verification of this feature simple, as well as cut down on a lot of noise (early experiments showed the letter-based pattern to produce a huge amount of noise). Many possible strings can match the patterns, so all of them are compared in turn, and if any are considered a "match", the comparison is done and the attribute is set to 1. Suppose pat_i and pat_{i+1} are values of patterns found on pages i and $i+1$ respectively, then equation 3.7 shows the possible values for the feature.

$$attr_{PN} = \begin{cases} 1 & \text{if } pat_i < pat_{i+1} \\ 0 & \text{if } pat_i \leq pat_{i+1} \end{cases} \quad (3.7)$$

The value of $attr_{PN}$ is 1 when a pattern on page i is considered "less than" a pattern found on page $i+1$. If no patterns fit this criteria, $attr_{PN}$ is 0. The case where $attr_{PN} = 0$ also encompasses a situation where insufficient information exists (i.e., no patterns matched) on one or both of the pages, and consequently no comparison could be made. Therefore, the default value of this attribute is 0, which corresponds to not automatically assuming the two pages compared are part of the same document. Similar to the header/footer and zone attributes described above, we assume the page number information discovered at the top to be independent of the information found on the bottom, therefore they are represented as distinct attributes.

Gathering: Hough Transform Data. A Hough Transform [13], is a process of detecting pixel patterns in an image by parameterizing the pixels in a way that make then easy to analyze. The original transform allowed for the detection of lines on an

image using representation in a polar space[3]. Various extensions have grown out of this technique that allow for the detection of more complicated entities (i.e., circles, squares), however we use the original implementation to look for lines only. The original transform uses the equation

$$r = x\cos\theta + y\sin\theta \tag{3.8}$$

to describe lines in the image. All of the points in a line will fall into a sinusoidal curve in the Hough space defined by (r, θ), and the points where those curves intersect (when they are superimposed) represent lines in the image. The intersections in the Hough space are counted in bins, and the higher the count, the more points contribute to a line. Using this implementation, each page image produces a map of points in Hough space that represent all of the line fragments in the image. Most of the points in the space will have relatively low counts, which correspond to the smattering of pixels contributing to that bucket from characters intersecting that line. However the buckets corresponding to actual lines in the image have dramatically higher counts, and can be easily identified. Figure 7 shows a page where two lines are captured using the Hough Transform. These lines will appear as high-valued points in the transformed space.

The comparison of two pages to produce our attribute involves a few simple steps. First, we determine the maximum intensity (i.e., highest count) of the buckets for each image. For pages p^i and p^j, let us call them $intensity^i$ and $intensity^j$, respectively. We then proceed to find all buckets in the Hough maps of the pages that have an intensity that is equal to or greater than 90% of the maximum intensity found. These resultant sets of lines are then compared by angle and magnitude. If all of the lines in the first set match all of the lines in the second set, the attribute

[3]Polar coordinates are (r, θ), where r is the magnitude, and θ is the angle.

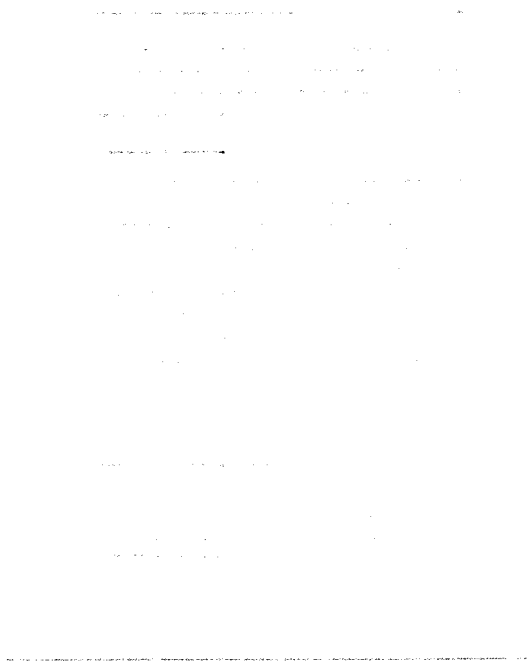

Figure 7: Lines captured by the Hough Transform.

assumes the value of 1, 0 otherwise.

Process: Translation

We must perform a translation step before using the attribute values as inputs to the classifier because signals to the network operate in an "on-off" manner, while several of our attributes produce values that do not fall into this binary definition. Therefore further work must be done to fit them into this space. Three distinct methods are used in the translation process. Table 2 indicates which method is used for which attribute. We proceed by now describing each of the methods in turn,

beginning with the easiest, inversion.

Attribute	Value Range	Translation Method
TF*IDF/VC	[0,1] $\subset \mathbf{R}$	Thresholding
Header/Footer	[0,1] $\subset \mathbf{R}$	Thresholding
Page Numbers	0,1	Inversion
Zones	positive integer	Thresholding (Pct)
Hough Transform	0,1	Inversion

Table 2: Attribute values and associated translations.

Translation: Inversion. The Hough Transform and the page number attributes both have binary value ranges; they can assume a value of 0 or 1. However both of these attributes produce the 1 value when they indicate similarity. Our input nodes must fire when the indication is that of dissimilarity, which means the signals for those two attribute classes are inverted. The actual implementation does not strictly invert the value, although the output obeys that property. For an attribute value $attr_v$ where $v \in \{HOUGH, PAGE\}$,

$$input_v = \begin{cases} 1 & \text{if } attr_v < 0.99, \\ 0 & \text{otherwise.} \end{cases} \tag{3.9}$$

Although this approach seems superfluous, the original translation implementation involved setting some adjustable value for the threshold of each attribute. Although this value is not experimentally determined at this time, we leave the implementation open to that possibility in the future.

Translation: Thresholding. The thresholding method involves determining an activation threshold for the attribute in question. We do this by experimentally determining the threshold with the lowest average error. We use the threshold as a linear

separator, and iterate through the training data, adjusting the threshold value by the average error, until the average error produced is higher than the previous iteration. After determining the optimal threshold value with respect to the training data, that threshold is used to create the "on-off" situation required for suitability as an input to the classifier. If a successive instance has an attribute value below the threshold, it is interpreted as not being similar enough, and the input corresponding to that attribute will not fire. However if the instance's attribute value is above the threshold, the associated input fires because the value indicates a high enough similarity to come from the same document. Figure 8 shows the error being determined from misclassifications using a threshold of 0.5.

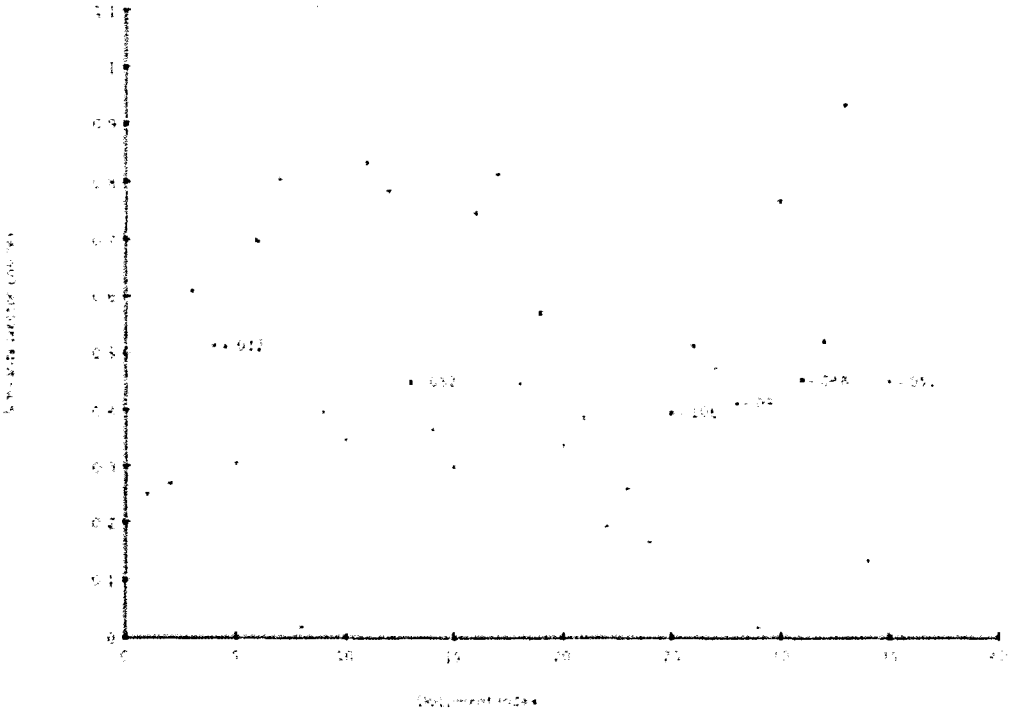

Figure 8: Determining the error from our threshold.

Translation: Thresholding (Percent). This method is a variation on the method described above. Recall that the value of $attr_{ZONE(TOP)}$ and $attr_{ZONE(BOT)}$ can be any value in $\mathbf{N_0}$, the set of whole numbers[4]. Also recall that the maximum of the two pages in question is produced during the proceeding processing phase (see page 19). Our value range is unbounded, which creates an issue when trying to determine a reasonable threshold. To circumvent this issue, instead of using the raw attribute value, we use the quantity $attr^i_{ZONE}/max(attr^i_{ZONE})$ as our value to threshold. We view this value as the percent of change in the amount of content between the two pages. The value for this quantity falls in the interval $[0, 1] \subset \boldsymbol{R}$, which we know is an acceptable range for the standard thresholding value described above.

Process: Classification

Our approach to this particular classification problem is to borrow a well-known technique from machine learning, known as a neural network. A neural network is an interconnected configuration of individual nodes; an individual node is known as a perceptron. Each node is also equipped with an activation function, which takes all incoming signals as input, and depending on the input, will "fire" under the correct conditions and propagate the signal forward from that node. The basic idea is to model the behavior of the human brain, where individual nodes fire due to some input, and pass along the signal to some set of nodes elsewhere in the network, until finally a set of output nodes receive the signal and produce some output. Each internodal connection in the network carries a weight, which indicates the strength of the signal as it is passed along that connection. The learning aspect arises from the ability to train the network using a set of examples. As the network trains, the weights between nodes can be adjusted to improve performance, until the network

[4]This formulation of the whole numbers includes 0.

reaches optimal performance[5]. An example network is shown in Figure 9.

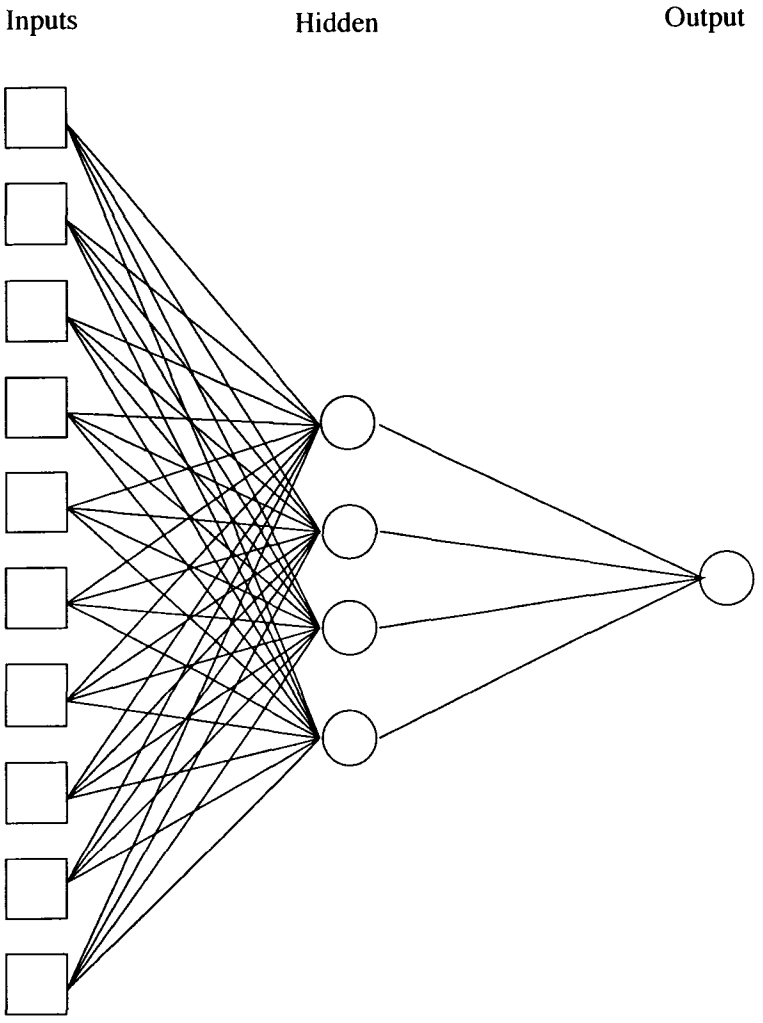

Figure 9: A example neural network classifier with one hidden layer.

Networks can be massively complicated, and even have connections where nodes "later" in the network can be connected as inputs to nodes "earlier" in the network. These types of networks are known as recurrent networks. However the most successful implementations have had much simpler designs. Similar to those other

[5]It is a known fact that such constructs can fall victim to reaching only local optima in a search space. Somtimes the same network configuration is retrained multiple times, with the starting weights randomized each time in an attempt to find the best among several optima.

implemenationst, only networks that feed in one direction, that is, with no loopback connections, are used in this experiment. Such a network is commonly known as a feed-forward network. Several configurations of this type of network were trained during the course of the experiment, so we quickly introduce some notation in order to ease discussion. We describe any network by indicating the number of nodes in each layer, with left-to-right syntax corresponding to an input-to-output nodes configuration. So, to describe a network with 10 input nodes followed by a hidden layer consisting of 8 nodes, another hidden layer of 4 nodes, and an output layer of 1 node, we label that network as a "10-8-4-1" network. Referring to Figure 9, the network shown is a 10-4-1 network. Clearly this notation can get unwieldy in the face of many-layered networks, however we intend to use no more than 2 hidden layers, and our output layer will always consist of 1 node, which will produce either a 1 or 0 as a prediction for the classification assignment to a particular problem instance.

The learning aspect of the network has several variations as well. The most well-known technique is called back-propagation. The idea is that after making a classification assignment, if the network is incorrect, an amount of adjustment is applied to each input connection to the output node(s). The amount of adjustment is dependent on how much each connection "contributed" to the incorrect assignment. The correction is then applied again through the hidden layers, affecting each of the previous layer's connections, until it reaches the connections coming from the input layer. This type of learning algorithm requires that the activation function at each of the non-input layer nodes be differentiable, so the correct amount of "blame" can be assigned to each input connection. In using the partial derivative of the function with respect to each input connection, we can then determine how to adjust that connection to make it "less incorrect" in future assignments with similar input. The partial derivative effectively creates a gradient which our network can "travel" along

to head towards a optimum in the search space. The network will train using the back-propagation method, and the activation function used is the sigmoid function[6], which is differentiable and continuous. Two parameters we can adjust before or during training are the learning rate and the bias. The learning rate is a multiplier that is applied to to every weight adjustment, and in some implementations it decreases as training progresses. This helps guard against the gradient ascent[7] constantly hopping around the optimum by slowly reducing the size of the steps taken for each training step. The other parameter, the bias, is usually implemented as a "quiet node" in that feeds into each non-input node in the network. The bias node is always active, and has an adjustable weight to each receiving node. The idea is that the bias provides a threshold the inputs to the nodes must overcome in order to "fire" the node. Otherwise, even a miniscule signal would get propagated to the next layer. The desired effect is to "fire" the node, therefore a suppressing quantity is needed to default the nodes to "off" until they receive a strong enough signal to fire. After training the network, we then deactivate the learning functionality, and merely run the network over fresh examples to test its performance.

[6] $\frac{1}{(1+e^{-x})}$

[7] or descent, depending on if your optima are maxima or minima. The description here uses maximization.

CHAPTER 4

ANALYSIS OF RESULTS

We begin this chapter by describing the generic performance metrics in use throughout the section. We then continue by describing in further detail the data used for this experiment, followed by a presentation of the baseline data. We then present the experimental results of the classifier, and conclude with an in-depth analysis of the performance of the classifier versus our baseline data.

Performance Metrics

Our classifier produces simple binary results, allowing us to use the standard statistical tools to analyze our performance. Each result from our classifier will fall into one of four categories: true positive, false positive, true negative, false negative. For the purposes of this analysis, the real-world values of an instance are $\{true, false\}$, where true corresponds to an actual boundary condition for that instance. The classification values will be $\{positive, negative\}$, where "positive" corresponds to an instance the classifier believed to be a boundary condition.

We can use these definitions to determine such quantities as precision and recall, which will be the standard metrics of the performance. Precision measures the proportion of boundary cases assigned a positive value that are also true cases. This measure, as it is defined here, is known in statistics as positive predictive value. The recall, also known as the sensitivity, measures how well a the classifier correctly identifies the true cases in the collection. The FP Rate is the false-positive rate, and it

is the complement of precision. Likewise, the FN Rate, or false-negative rate, is the measurement complement of the recall. A more complete definition of the metrics used is given starting on page 46.

Experimental Data

We select a random 500 documents from a larger collection of documents, for a total of 1137 pages, or 1136 possible boundary instances. This means that, over the entirety of the sample, there a only a few more negative examples than positive ones. All of the documents are taken from a collection of scientific and correspondence documents, so the format of the documents may vary widely, as well as the subject matter. The lengths of the documents are also restricted between 1 and 200 pages[1]. Admittedly, the size of the sample is small compared to typical training sets. The reason for this is that the effort ncessary to compile these examples proved to be much more time consuming than previously thought. As such, the sample gathering process was severely restricted due to time constraints.

To avoid overfitting to the data, we use K-fold cross validation, with $K = 10$. K-fold cross validation is a form of partitioning the data set into k smaller subsets. One of the subsets is left out for validation while the others are used for training. The process is repeated k times in total, each time using a different subset as the validation set. Either the results from the K separate runs can be combined, or the best estimation is then used in practical application. This helps avoid the overfitting to one particular attribute of the training data. Therefore, when a "3-fold removed set" is mentioned, we mean that the 3^{rd} subset of the partitioned data has been left out of training, and is used as the validation set.

[1]In case an eyebrow was raised, our sample did not contain any 200-page documents. 200 is just the limit imposed; the longest document that occurred in the collection was actually 39 pages.

K	# of samples
1	113
2	113
3	113
4	113
5	113
6	113
7	113
8	113
9	113
10	119
total	1136

Table 3: Our $K = 10$ subsample sets.

We begin by using a constant learning rate of 1.0 when training, with a bias of -1 to each of the non-input nodes. We train several different network configurations, to see if one configuration significantly outperforms any other. Each of the individual configuration-training set pairings is trained for 200 iterations. Each iteration trains the classifier over the entire training set.

Recall that our original hypothesis is that the combined performance of a set of unique attributes could outperform the individual attributes. In order to determine if this is true, we need to know the performance of the attributes as standalone classifiers. The performance of the individual inputs is displayed in Table 4. The 6-fold removed training set produced the best classifier, so we show the performance of the attributes after undergoing threshold adjustment for that subset. Displaying the individual input performances gives us a feel for how well the standalone parts can perform.

Attrbiute	Optimal Value	Precision	Recall	FP Rate	FN Rate
$attr_{TF}$	0.0366	0.8565	0.9184	0.1111	0.0816
$attr_{HDR}$	0.0041	0.6156	0.9371	0.4226	0.0629
$attr_{FTR}$	0.0012	0.5085	0.9744	0.6801	0.0256
$attr_{PN(TOP)}$	N/A	0.4194	1.0000	1.0000	0.0000
$attr_{PN(BOT)}$	N/A	0.4194	1.0000	1.0000	0.0000
$attr_{ZONE(TOP)}$	0.3396	0.5589	0.6970	0.3973	0.3030
$attr_{ZONE(BOT)}$	0.1659	0.4574	0.8625	0.7391	0.1375
$attr_{HOUGH}$	N/A	0.5667	0.9604	0.5303	0.0396

Table 4: Baseline performance for 6-fold removed training set.

Experimental Results

We train three different network configurations (8-3-1, 8-5-1, and 8-10-1) over our subsample sets. The best-performing network used the $k = 6$ training set, which means 6^{th} subsample was omitted for testing purposes. The results of the network with the best performance is shown in Table 5.

Configuration	Precision	Recall	FP Rate	FN Rate
8-10-1	1.0	0.610619469026549	0.0	0.389380530973451

Table 5: Best-performing network.

As can be seen from Table 5, our classifier seems to be truly right when it thinks it is right, but it is overly conservative. The precision value of 1.0 indicates that there were no false positives; every case assigned as a boundary was in fact a boundary situation. However the recall value of ≈ 0.61 indicates that the classifier was only able to identify about 61% of the real boundary cases. While this indicates some level of promise, let us investigate if there is anything we can do to improve the recall. We reused the original seed values for the 8-10-1 network and retrained using a

higher number of iterations (500). The results were identical to the classifier trained on only 200 iterations, so it is unlikely that our classifier suffers from lack of traning repetition on the data. One possibility could be that the classifier suffers from the hopping problem described on page 28. Our learning rate was not adjustable, and so our adjustments may have just been overwhelming in the latter training iterations. Another possibility may be that the target function our classifier is trying to approximate is more complicated than we originally surmised, and requires another layer of hidden nodes to be better approximated[2].

Both of the conditions described can be tested to see if they have any effect on our performance. In one case, we must control the network configuration and only vary the learning rate, in the other, we must perturb the network configuration. The former case we will call the $\Delta - Learn$ case, and the latter will be the $Hidden+$ case.

The $\Delta - Learn$ Case

The $\Delta - Learn$ situation is actually simple to test. We use the original, untrained network that seeded all 8-10-1 networks, and just tweak the training algorithm to slowly decrease the value of the learning rate towards zero. The gradual descent will have the learning rate begin at 1.0, and descend to 0.01 by the final iteration. To do this, we determine the size of the step necessary to descend the value by for each iteration:

$$\text{descent amount} = \frac{\text{total difference}}{\#\text{ of iterations}} = \frac{(1.0 - 0.01)}{199} \approx 0.00497 \qquad (4.1)$$

and we try both 0.5 and 0.99 as the "total difference" values, as shown in Table 6.

[2]Networks with 1 hidden layer can represent all continuous functions with enough hidden nodes in the layer. However with 2 hidden layers, it is possible to even represent discontinuous functions [22].

Total Difference	Precision	Recall	FP Rate	FN Rate
0.99	1.0	0.292035398230089	0.0	0.707964601769911
0.5	1.0	0.292035398230089	0.0	0.707964601769911

Table 6: $\Delta - Learn$ network performance.

After some experimentation, it is obvious that our original classifier did not suffer from hopping around the optimum. If anything, we apparently descreased the size of the learning far too early, which left our classifier stuck mid-ascent during training, which reflects itself as poorer performance than the original. The precision did not drop off, but the recall is significantly worse, meaning the $\Delta - Learn$ classifier was not able to identify as many of the actual boundary cases as the originally trained classifier.

The *Hidden+* Case

The *Hidden+* case is considerably harder to properly establish the control variables for. The nearest approximation to varying only the configuration would be to take the seed values for the 8-10-1 network and inject a new layer of random weights that would correspond to the new hidden layer. Even doing this does not insure the pre-existing weights will begin to converge in the same manner they did before; the new hidden layer will interpret the signals differently, as well as provide an extra layer of feedback in the training process. In keeping with the vein of the original configuration tests, we simply create a brand new configuration using randomized weights, and train it against all of our validation sets, like we did before. We keep the learning rate at a constant of 1 for this experiment, so we may keep some semblance of control over this case. The configuration we try is 8-10-4-1, and we train it on across all training sets, using 200 iterations per unique network.

A comparison between the results in Table 7 and Table 5 shows that adding

K	Precision	Recall	FP Rate	FN Rate
1	1.0	0.530973451327434	0.0	0.469026548672566
2	1.0	0.592920353982301	0.0	0.407079646017699
3	1.0	0.407079646017699	0.0	0.592920353982301
4	1.0	0.265486725663717	0.0	0.734513274336283
5	1.0	0.424778761061947	0.0	0.575221238938053
6	1.0	0.610619469026549	0.0	0.389380530973451
7	1.0	0.300884955752212	0.0	0.699115044247788
8	1.0	0.451327433628319	0.0	0.548672566371681
9	1.0	0.389380530973451	0.0	0.610619469026549
10	1.0	0.571428571428571	0.0	0.428571428571429

Table 7: The 8-10-4-1 (*Hidden+*) configuration performance.

an additional layer did not seem to affect the performance of the classifier at all. We did no worse, but we also did no better. It appears that whatever is lacking in performance is not visibly dependent on the learning rate parameter, nor on the depth of our network configuration.

Summary

Our experiment has met with moderate success. The classifier appears to be able to improve our precision over the individual attributes used as inputs, however the ability to identify all cases (recall) did not significantly improve. So, while the performance is arguably better, it is certainly no worse. We also attempted several variations in an attempt to imrpove the performance of the classifier further, but to no avail. One issue that was left unaddressed was the question of sample size. The sample size is notably smaller than a standard training set for a machine learning method, and the possibility exists that a significant increase in our training sample size will translate into improved performance. The cause of this would be that our feature set chosen as attributes did not encounter enough examples to come to a stable optimum during training; in other words the current sample set most likely does not

create an accurate enough representation of our search space. However, as stated earlier, compiling the data in order to feed into the classifier is a tedious, expensive, and slow process. While an increased sample size may improve performance, it is currently beyond the scope of this thesis to investigate such a hypothesis.

CHAPTER 5

CONCLUSIONS AND FURTHER THOUGHTS

In retrospect, it appears that the strength posited in the hypothesis (that using unique facets of data increases performance) also proved to be the largest liability in showing its efficacy. The effort required to gather the numerous attributes proved to be prohibitive towards generating a signifcantly large sample size. However even with a small sample set, we were able to produce a noticeable increase in performance, indicating that method may still prove to have merit.

The simplest solution to our sample size problem would be to throw more processing power at it. Many aspects of the process fall neatly into a parallelized solution, which would provide a significant improvement in preparation time. Of course as hardware keeps getting smaller, faster, and cheaper, eventually the problems encountered today will be a trivial hurdle of tomorrow.

Assuming the problem encountered in this thesis is addressed, or at least circumvented through additional resources, some other variations on the system presented here may prove to improve performance. Obviously, the careful addition of new attributes will provide a new dimensionality to the problem instances that may well help to increase the performance significantly. Also, a more thorough treatment of some of the attributes used here may have a similar effect. For example, the zone-based attributes were of some use in classification, but some work could be done to make them stronger standalone discriminators.

A slightly more theoretically-based approach may be to treat the input values

as "belief values", which is an interpretation of probability that allows for degrees of uncertainty in the sources of the probabilities produced [25]. Several experiments have been run using belief values as inputs to statistically-driven classifiers and have shown potential [14]. Such approaches to combining unique signals from the data may prove to be extremely useful in future experiments, without adding nearly as much processing time as continuously adding new attributes in order to improve performance.

As research and technology move forward, clearly the need for more comprehensive approaches to many problems will increase as well. This experiment exemplifies the issues that many of these multifaceted approaches will incur; the largest issue being that of richness of data vs. time allocated to complete the task. While many techniques may provide strong experimental results, combining them together may easily prove to be prohibitive without further thought into the methods by which the data is prepared and represented.

BIBLIOGRAPHY

[1] Mausumi Acharyya and Malay K. Kundu. Document image segmentation using wavelet scale-space features.

[2] Charles Anderson and et al. Semi-automated boundary tracing of medical images for three-dimensional model development.

[3] Andrew D. Bagdanov. Style Characterization of Machine Printed Texts. PhD thesis, Universiteit van Amsterdam, 2004.

[4] Henry S. Baird, Daniel Lopresti, Brian D. Davison, and William M. Pottenger. Robust document image understanding technologies. In HDP '04: Proceedings of the 1st ACM workshop on Hardcopy document processing, pages 9–14, New York, NY, USA, 2004. ACM Press.

[5] Doug Beeferman, Adam Berger, and John D. Lafferty. Statistical models for text segmentation. Machine Learning, 34(1-3):177–210, 1999.

[6] D. Bikel, S. Miller, R. Schwartz, and R. Weischedel. Nymble: a high-performance learning name-finder, 1997.

[7] R. Cattoni, T. Coianiz, S. Messelodi, and C. Modena. Geometric layout analysis techniques for document image understanding: a review, 1998.

[8] De Chen. Form processing with the hough transform. Master's thesis, UNLV, 2003.

[9] Kevyn Collins-Thompson and Radoslav Nickolov. A clustering-based algorithm for automatic document separation, 2002.

[10] Dayne Freitag and Andrew Kachites McCallum. Information extraction with hmms and shrinkage. In Proceedings of the AAAI-99 Workshop on Machine Learning for Informatino Extraction, 1999.

[11] P. Gupta, N. Vohra, S. Chaudhury, and S. Joshi. Wavelet based page segmentation, 2000.

[12] D. Hiemstra and A. de Vries. Relating the new language models of information retrieval to the traditional retrieval models, 2000.

[13] Paul Hough. P.v.c. method and means for recognizing complex patterns., 1962. U.S. Patent 3069654.

[14] Gertrud Jeschke and Mounia Lalmas. Hierarchical text categorisation based on neural networks and dempster-shafer theory of evidence.

[15] Todd Jochem, Dean Pomerleau, and Charles Thorpe. MANIAC: A next generation neurally based autonomous road follower. In Proceedings of the International Conference on Intelligent Autonomous Systems: IAS–3, 1993.

[16] R. Kohavi, B. Becker, and D. Sommerfield. Improving simple bayes, 1997.

[17] T. R. Leek. Information extraction using hidden Markov models. Master's thesis, UC San Diego, 1997.

[18] X. Li. Versatile page numbering analysis. In Proceedings of the 9th International Conference on Document Analysis and Recognition, 2007.

[19] J. Liang, I. Phillips, J. Ha, and R. Haralick. Document zone classification using the sizes of connected components, 1996.

[20] Jon Pastor and Suzanne Liebowitz Taylor. Recognizing structured forms using neural networks.

[21] Lawrence R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. pages 267–296, 1990.

[22] Stuart Russell and Peter Norvig. Artificial Intelligence: A Modern Approach. Pearson Education, Inc., Upper Saddle River, New Jersey 07458, 2003.

[23] G. Salton, A. Wong, and A. C. S. Yang. A vector space model for automatic indexing. Communications of the ACM, 18:229–237, 1975.

[24] Kristie Seymore, Andrew McCallum, and Roni Rosenfeld. Learning hidden Markov model structure for information extraction. In AAAI 99 Workshop on Machine Learning for Information Extraction, 1999.

[25] Glenn Shafer. A Mathematical Theory of Evidence. Princeton University Press, Princeton, N.J., 1976.

[26] Basilio Sierra, Nicolas Serrano, Pedro Larranaga, Eliseo J. Plasencia, Inaki Inza, Juan Jose Jimenez, Pedro Revuelta, and Maria Luisa Mora. Using bayesian networks in the construction of a bi-level multi-classifier. a case study using intensive care unit patients data. Artificial Intelligence in Medicine, 22(3):233–248, 2001.

[27] Moninder Singh and Gregory M. Provan. Efficient learning of selective bayesian network classifiers. In International Conference on Machine Learning, pages 453–461, 1996.

[28] S. N. Srihari. Document image understanding. In Proceedings of 1986 ACM Fall Joint Computer Conference, pages 87–96, Los Alamitos, CA, 1986. IEEE Computer Society Press.

APPENDIX A

VARIOUS EQUATIONS AND METRICS

Document Representation

The general idea is to represent a document D as a vector of terms. Suppose we have a document labeled as D_i containing n unique terms. We represent this entity mathematically as such:

$$D_i = [t_{1,i}, t_{2,i}, t_{3,i}, \ldots, t_{n,i}] \tag{A.1}$$

where $t_{1,i}$ (known as the term frequency) represents the number of occurences of term t_1 in document D_i.

We will use this defintion in the following sections.

TF*IDF Model

$$w_{t,D} = t_{t,D} * \ln(\frac{|D|}{\{t \epsilon D\}}) \tag{A.2}$$

$|D|$ is the number of documents in our collection, and $\{t \epsilon D\}$ is the number of documents that term t appears in. Collectively, the multiplier added to the equation serves to include information about the term with respect to the entire collection.

Term Count Model

For the purposes of the comparison we define the "term weight" vector for document D_i:

$$W_i = [w_{1,i}, w_{2,i}, \ldots, w_{n,i}] \tag{A.3}$$

where

$$w_{t,i} = t_{t,i} \tag{A.4}$$

Vector Cosine

Our scalar similarity value, Sim_{D_i,D_j}, is calculated as such:

$$W_i \cdot W_j = \sum_{\forall l,m:t_{i,l}=t_{j,m}} w_{i,l} * w_{j,m} \tag{A.5}$$

$$\|W_i\| = \sqrt{(w_{1,i}^2 + w_{2,i}^2 + \cdots + w_{n,i}^2)} \tag{A.6}$$

$$Sim_{D_i,D_j} = \frac{W_i \cdot W_j}{\|W_i\| * \|W_j\|} \tag{A.7}$$

Precision and Recall

If the documents are completely dissimilar, then the numerator term, $W_i \cdot W_j$, is 0. However if $W_i = W_j$, the resulting value is 1, which acts as our upper bound for this value. Assume that for any given test instance t, it has a real-world value of

true, *false*, and a possible classification value of *positive*, *negative* which correspond to their respective real-world equivalents. So we want every "true" example to be classified as "positive".

	Positive	Negative
True	30	20
False	10	40

Table 8: Short example of statistical errors

Remember that true positives are the number of test instances that are actually true and classified correctly. False positives are the number of test instances that were not true but classified as so. True negatives are test instances that are false and classified correctly. False negatives are the instances that are in fact true but classified as false.

Precision is a quantity that measures the proportion of actually positive instances that were classified as true. This quantity is computed as:

$$Precision = \frac{TP}{TP + FP} \tag{A.8}$$

In our example, this is $30/(30 + 10) = 0.75$.

Recall measuures what proportion of the true instances were actually classified as positive:

$$Recall = \frac{TP}{TP + FN} \tag{A.9}$$

In our example, this is $30/(30+20) = 0.6$. Associated values are also the false negative rate, which is $1 - Recall$, and the false positive rate, which is $\frac{FP}{FP+TN}$.

APPENDIX B

EXAMPLE OF THE THRESHOLDING METHOD

In this appendix we illustrate the operation of the thresholding method.

Example	Is Boundary?	VC
1	N	0.3799
2	N	0.4921
3	Y	0.0562
4	N	0.3621
5	N	0.9227
6	N	0.7001
7	N	0.5056
8	Y	0.1192
9	N	0.2739
10	Y	0.2012

Table 9: An example of boundaries and their corresponding vector cosine values.

Suppose Table 9 is our set of instances. There are 3 actual boundaries in the set (instances 3, 8, and 10), indicating that the set consists of 4 documents. Table 10 shows the iterations of the process of determining the average error, then adjusting the threshold value according to that average, and then attempting the classification again. The process stops when the new average error calculated is greater than the error determined in the previous round, or when the error calculated is 0. In the example shown, our starting threshold value is 0.5. After round 1, the value is adjusted by -0.1582 to 0.3418, which is then used for round 2. After round 2, the value is adjusted by -0.0679 to 0.2739, which then produces no error in round 3. Since there is no error, the loop exits, producing the value 0.2739 as the threshold.

Example	Round 1	Error	Round 2	Error	Round 3	Error
1	Y	-0.1201	N	-	N	-
2	Y	-0.0079	N	-	N	-
3	Y	-	Y	-	Y	-
4	Y	-0.1379	N	-	N	-
5	N	-	N	-	N	-
6	N	-	N	-	N	-
7	N	-	N	-	N	-
8	Y	-0.3808	Y	-	Y	-
9	Y	-0.2261	Y	-0.0679	N	-
10	Y	-0.2988	Y	-	Y	-
	# wrong	avg. error	# wrong	avg. error	# wrong	avg. error
	6	-0.1582	1	-0.0679	0	0.0

Table 10: Threshold approximation.

APPENDIX C

SUPPLEMENTAL DATA

This appendix contains various data sets compiled over the course of the thesis. It is provided placate a reader's curiosity, should the urge to look over more data take them.

Alternative Network Performances

Three network configurations were trained using 10-fold cross validation. Each table represents a particular configuration. Each row in a table describes the classification performance of the classifier after 200 training iterations. The "K" for the row describes the subsample fold used as the validation set, so if $K = 2$, the 2^{nd} subset was left out of the training subsample, and was used as the validation set to determine performance.

K	Precision	Recall	FP Rate	FN Rate
1	1.0	0.530973451327434	0.0	0.469026548672566
2	1.0	0.592920353982301	0.0	0.407079646017699
3	1.0	0.407079646017699	0.0	0.592920353982301
4	1.0	0.256637168141593	0.0	0.743362831858407
5	1.0	0.424778761061947	0.0	0.575221238938053
6	1.0	0.610619469026549	0.0	0.389380530973451
7	1.0	0.300884955752212	0.0	0.699115044247788
8	1.0	0.451327433628319	0.0	0.548672566371681
9	1.0	0.389380530973451	0.0	0.610619469026549
10	1.0	0.554621848739496	0.0	0.445378151260504

Table 11: Performance of the 8-3-1 configuration.

Baseline Data

This the baseline data gathered for the thresholds of the individual attributes across the varying training sets. "K" refers to the fold left out, so $K = 2$ means the training set is comprised of all subsamples except the second one.

K	Precision	Recall	FP Rate	FN Rate
1	1.0	0.530973451327434	0.0	0.469026548672566
2	1.0	0.592920353982301	0.0	0.407079646017699
3	1.0	0.407079646017699	0.0	0.592920353982301
4	1.0	0.256637168141593	0.0	0.743362831858407
5	1.0	0.424778761061947	0.0	0.575221238938053
6	1.0	0.610619469026549	0.0	0.389380530973451
7	1.0	0.300884955752212	0.0	0.699115044247788
8	1.0	0.451327433628319	0.0	0.548672566371681
9	1.0	0.389380530973451	0.0	0.610619469026549
10	1.0	0.571428571428571	0.0	0.428571428571429

Table 12: Performance of the 8-5-1 configuration.

K	Precision	Recall	FP Rate	FN Rate
1	1.0	0.530973451327434	0.0	0.469026548672566
2	1.0	0.592920353982301	0.0	0.407079646017699
3	1.0	0.407079646017699	0.0	0.592920353982301
4	1.0	0.256637168141593	0.0	0.743362831858407
5	1.0	0.424778761061947	0.0	0.575221238938053
6	1.0	0.610619469026549	0.0	0.389380530973451
7	1.0	0.300884955752212	0.0	0.699115044247788
8	1.0	0.451327433628319	0.0	0.548672566371681
9	1.0	0.389380530973451	0.0	0.610619469026549
10	1.0	0.563025210084034	0.0	0.436974789915966

Table 13: Performance of the 8-10-1 configuration.

Attrbiute	Optimal Value	Precision	Recall	FP Rate	FN Rate
$attr_{TF}$	0.0423	0.8542	0.9391	0.1224	0.0609
$attr_{HDR}$	0.0040	0.6248	0.9323	0.4276	0.0677
$attr_{FTR}$	0.0013	0.5194	0.9391	0.1224	0.0609
$attr_{PN(TOP)}$	N/A	0.4330	1.0	1.0	0.0
$attr_{PN(BOT)}$	N/A	0.4330	1.0	1.0	0.0
$attr_{ZONE(TOP)}$	0.3358	0.5545	0.6772	0.4155	0.3228
$attr_{ZONE(BOT)}$	0.1964	0.4727	0.8488	0.7241	0.1512
$attr_{HOUGH}$	N/A	0.5779	0.9549	0.5328	0.0451

Table 14: Baseline Performance for 1-fold removed training set.

Attrbiute	Optimal Value	Precision	Recall	FP Rate	FN Rate
$attr_{TF}$	0.0390	0.8429	0.9276	0.1244	0.0724
$attr_{HDR}$	0.0045	0.6126	0.9346	0.4252	0.0654
$attr_{FTR}$	0.0010	0.5079	0.9720	0.6773	0.0280
$attr_{PN(TOP)}$	N/A	0.4184	1.0000	1.0000	0.0000
$attr_{PN(BOT)}$	N/A	0.4184	1.0000	1.0000	0.0000
$attr_{ZONE(TOP)}$	0.2538	0.4931	0.7477	0.5529	0.2523
$attr_{ZONE(BOT)}$	0.0483	0.4475	0.8762	0.7782	0.1238
$attr_{HOUGH}$	N/A	0.5659	0.9533	0.5261	0.0467

Table 15: Baseline Performance for 2-fold removed training set.

Attrbiute	Optimal Value	Precision	Recall	FP Rate	FN Rate
$attr_{TF}$	0.0414	0.8598	0.9379	0.1206	0.0621
$attr_{HDR}$	0.0046	0.6390	0.9379	0.4178	0.0621
$attr_{FTR}$	0.0009	0.5221	0.9712	0.7010	0.0288
$attr_{PN(TOP)}$	N/A	0.4409	1.0000	1.0000	0.0000
$attr_{PN(BOT)}$	N/A	0.4409	1.0000	1.0000	0.0000
$attr_{ZONE(TOP)}$	0.4390	0.5794	0.5987	0.3427	0.4013
$attr_{ZONE(BOT)}$	0.2750	0.4906	0.8093	0.6626	0.1907
$attr_{HOUGH}$	N/A	0.5722	0.9579	0.5647	0.0421

Table 16: Baseline Performance for 3-fold removed training set.

Attrbiute	Optimal Value	Precision	Recall	FP Rate	FN Rate
$attr_{TF}$	0.0406	0.8594	0.9362	0.1302	0.0638
$attr_{HDR}$	0.0043	0.6241	0.9362	0.4792	0.0638
$attr_{FTR}$	0.0013	0.5363	0.9745	0.7161	0.0255
$attr_{PN(TOP)}$	N/A	0.4594	1.0000	1.0000	0.0000
$attr_{PN(BOT)}$	N/A	0.4594	1.0000	1.0000	0.0000
$attr_{ZONE(TOP)}$	0.4563	0.5848	0.5723	0.3454	0.4277
$attr_{ZONE(BOT)}$	0.2748	0.4955	0.8170	0.7071	0.1830
$attr_{HOUGH}$	N/A	0.6151	0.9553	0.5081	0.0447

Table 17: Baseline Performance for 4-fold removed training set.

Attrbiute	Optimal Value	Precision	Recall	FP Rate	FN Rate
$attr_{TF}$	0.0430	0.8700	0.9355	0.1165	0.0645
$attr_{HDR}$	0.0029	0.6344	0.9290	0.4462	0.0710
$attr_{FTR}$	0.0014	0.5500	0.9699	0.6613	0.0301
$attr_{PN(TOP)}$	N/A	0.4545	1.0000	1.0000	0.0000
$attr_{PN(BOT)}$	N/A	0.4545	1.0000	1.0000	0.0000
$attr_{ZONE(TOP)}$	0.4187	0.5858	0.6022	0.3548	0.3978
$attr_{ZONE(BOT)}$	0.3082	0.5152	0.8043	0.6308	0.1957
$attr_{HOUGH}$	N/A	0.6024	0.9548	0.5251	0.0452

Table 18: Baseline Performance for 5-fold removed training set.

Attrbiute	Optimal Value	Precision	Recall	FP Rate	FN Rate
$attr_{TF}$	0.0366	0.8565	0.9184	0.1111	0.0816
$attr_{HDR}$	0.0041	0.6156	0.9371	0.4226	0.0629
$attr_{FTR}$	0.0012	0.5085	0.9744	0.6801	0.0256
$attr_{PN(TOP)}$	N/A	0.4194	1.0000	1.0000	0.0000
$attr_{PN(BOT)}$	N/A	0.4194	1.0000	1.0000	0.0000
$attr_{ZONE(TOP)}$	0.3396	0.5589	0.6970	0.3973	0.3030
$attr_{ZONE(BOT)}$	0.1659	0.4574	0.8625	0.7391	0.1375
$attr_{HOUGH}$	N/A	0.5667	0.9604	0.5303	0.0396

Table 19: Baseline Performance for 6-fold removed training set.

Attrbiute	Optimal Value	Precision	Recall	FP Rate	FN Rate
$attr_{TF}$	0.0422	0.8619	0.9418	0.1252	0.0582
$attr_{HDR}$	0.0046	0.6299	0.9353	0.4562	0.0647
$attr_{FTR}$	0.0013	0.5250	0.9741	0.7317	0.0259
$attr_{PN(TOP)}$	N/A	0.4536	1.0000	1.0000	0.0000
$attr_{PN(BOT)}$	N/A	0.4536	1.0000	1.0000	0.0000
$attr_{ZONE(TOP)}$	0.4865	0.5850	0.5711	0.3363	0.4289
$attr_{ZONE(BOT)}$	0.3008	0.5061	0.8060	0.6530	0.1940
$attr_{HOUGH}$	N/A	0.5798	0.9547	0.5742	0.0453

Table 20: Baseline Performance for 7-fold removed training set.

Attrbiute	Optimal Value	Precision	Recall	FP Rate	FN Rate
$attr_{TF}$	0.0401	0.8504	0.9326	0.1263	0.0674
$attr_{HDR}$	0.0041	0.6064	0.9348	0.4671	0.0652
$attr_{FTR}$	0.0014	0.5263	0.9685	0.6713	0.0315
$attr_{PN(TOP)}$	N/A	0.4350	1.0000	1.0000	0.0000
$attr_{PN(BOT)}$	N/A	0.4350	1.0000	1.0000	0.0000
$attr_{ZONE(TOP)}$	0.4324	0.5735	0.6135	0.3512	0.3865
$attr_{ZONE(BOT)}$	0.1920	0.4736	0.8472	0.7249	0.1528
$attr_{HOUGH}$	N/A	0.5637	0.9640	0.5744	0.0360

Table 21: Baseline Performance for 8-fold removed training set.

Attrbiute	Optimal Value	Precision	Recall	FP Rate	FN Rate
$attr_{TF}$	0.0425	0.8677	0.9434	0.1170	0.0566
$attr_{HDR}$	0.0044	0.6349	0.9434	0.4415	0.0566
$attr_{FTR}$	0.0012	0.5352	0.9760	0.6897	0.0240
$attr_{PN(TOP)}$	N/A	0.4487	1.0000	1.0000	0.0000
$attr_{PN(BOT)}$	N/A	0.4487	1.0000	1.0000	0.0000
$attr_{ZONE(TOP)}$	0.4256	0.5631	0.6122	0.3865	0.3878
$attr_{ZONE(BOT)}$	0.2771	0.4954	0.8148	0.6755	0.1852
$attr_{HOUGH}$	N/A	0.5819	0.9673	0.5656	0.0327

Table 22: Baseline Performance for 9-fold removed training set.

Attrbiute	Optimal Value	Precision	Recall	FP Rate	FN Rate
$attr_{TF}$	0.0336	0.8805	0.9108	0.0931	0.0892
$attr_{HDR}$	0.0049	0.6354	0.9291	0.4017	0.0709
$attr_{FTR}$	0.0010	0.5177	0.9703	0.6810	0.0297
$attr_{PN(TOP)}$	N/A	0.4297	1.0000	1.0000	0.0000
$attr_{PN(BOT)}$	N/A	0.4297	1.0000	1.0000	0.0000
$attr_{ZONE(TOP)}$	0.3415	0.5493	0.6888	0.4259	0.3112
$attr_{ZONE(BOT)}$	0.1588	0.4647	0.8581	0.7448	0.1419
$attr_{HOUGH}$	N/A	0.5830	0.9565	0.5155	0.0435

Table 23: Baseline Performance for 10-fold removed training set.

VITA

Graduate College
University of Nevada, Las Vegas

Marc-Allen Cartright

Local Address:
9490 S. Bermuda Rd, Apt. 1029
Las Vegas, Nevada 89123

Home Address:
12 Garfield Dr.
Westborough, Massachusetts 01581

Degrees:
Bachelor of Science, Computer Science, 2002
Stanford University, Palo Alto California

Thesis Title:
Document Boundary Determination using Structural and Lexical Analysis

Thesis Examination Committee:
Chairman, Dr. Kazem Taghva, Ph. D.
Committee Member, Dr. Thomas Nartker, Ph. D.
Committee Member, Jan B. Pedersen, Ph. D.
Graduate Faculty Representative, Dr. Z. Y. Wang, Ph. D.