

Program Proposal for Ph.D. in Computer Science

“Voracious Classifiers”

May 8, 2006

Michael A. Moll

Computer Science & Engineering Dept

Lehigh University

19 Memorial Dr West

Bethlehem, PA 18017 USA

E-mail: mam7@lehigh.edu

1. INTRODUCTION

The “Digital Age” has not led to a paperless office^{SH03} or rendered hard copies of documents obsolete as some predicted. If anything, it seems this appreciation of the necessity and value these documents have has raised the accessibility and distribution of non-digital documents. However, this increase in accessibility also leads to an overload of information as the rate at which documents can be obtained and digitized exceeds the rate at which sense (context, content, etc) can be made of documents. Existing methods for extracting content from document images are limited in several ways: they don’t yet perform well on the full diversity of document and content types, and they are incapable of being trained on the huge collections of document images now rapidly being scanned in (by Google, Amazon, and others). This proposal suggests a research program that achieves high speed classification of document image content utilizing extremely large scale data sets.

Classification problems are observed in different contexts that may focus on different, yet complementary goals. Within the Digital Libraries community, one important goal is the preservation and sharing of historical documents. While the expectations on the speed of per image processing may not be that high, the overall volume of images being processed is most likely extremely high. Overall accuracy of classification in the Digital Libraries context is probably more important and useful than high speed in processing. Classification of document content also facilitates Web Search problems. In this context a higher premium is placed on speed of processing than accuracy as any information about a document is more useful than none. These tradeoffs between speed and accuracy can be combined in the needs of the intelligence community. The scenario of a soldier in the field finding a scrap of a document of potentially mixed content types and languages, requires both high speed and accurate classification of that document image before any translation or downstream processing can take place.

The internet makes many large collections of document images accessible, some of which already have their content symbolically encoded. How is it possible to utilize this massive amount of existing information to quickly and accurately classify new and unseen documents? Further, is it possible to do this in full generality, for all possible types of documents? What level of accuracy is necessary for classification of new documents to be minimally useful and how can we assure this level given the vast diversity of document images that exist? What are the lowest goals we can accept for accuracy to improve throughput?

One scientific goal of this research is to investigate methods for high speed, versatile classifiers that utilize extremely large, internet-scale data sets. More specifically, goals of this research are to achieve classification of document images in effectively constant ($O(1)$) expected time, ideally in practice at I/O rates, that is as fast as data can be read. These algorithms will be applied in the context of document image content extraction to locate regions containing handwriting, machine print, graphics, etc. Regardless of implementation, *versatility*—that is, good performance across a wide range of document types—will be the most important emphasis. We will try to avoid overspecialized, fragile algorithms, that is ones that perform well only on special cases. We will investigate “voracious” classifiers than can learn from extremely large number of training samples, assuming that larger training sets generally allow for better classification than smaller ones and that very large sets of data are accessible. Initial research in this area has been promising.^{BMN⁺06}

Current classification techniques in this domain suffer from a variety of problems. Classifiers built on static k -d trees or CARTs may perform well with a few hundred thousand training samples, but would choke on the billions of training samples we expect will be necessary. With so much data now easily and freely accessible (zoning processing costs aside) or synthetically generatable, why not use as much of it as possible to make decisions? We want to employ methods that can “absorb” as much of this data as possible (assuming no physical constraints on the amount we can store) and make use of it at a later time when necessary. Therefore, if we are clever enough about classifier design, accuracy should depend only on collecting a truly representative training sample, the larger the training sample the better.

2. LITERATURE REVIEW

Following is an overview of literature and ideas relevant to this proposal. We begin with a review of basic notation and methodology:

- **k Nearest Neighbors Classifiers**

Each training and test sample is a pixel in a document image, and each pixel owns d features which are scalar properties extracted by image processing in the vicinity of that pixel. Our statistical sample space (the “universe”) $U = \mathbf{R}^d$, the multidimensional reals. We expect the dimensionality $d \approx 100$. A member of the sample space is called $\mathbf{x}, \mathbf{y}, \mathbf{z}$, etc.

All samples belong to one of m classes $\mathbf{C} = \{c_j\}_{j=1,\dots,m}$. The number of classes m will, we expect, be ≈ 10 .

The components $\{x_i\}_{i=1,\dots,d}$ of an observed sample \mathbf{x} are measurable *features* of it, chosen with the hope that they are *discriminating*, that is, for any pair of samples, they have *similar* sets of feature values if and only if they belong to the same class.

Training data consists of a set of samples $\mathbf{T} \subset \mathbf{R}^d$ labeled with their classes. We expect the number of training samples $n \equiv |\mathbf{T}| \approx 1$ billion. The true class of each training sample is given by a *ground-truth* function $\mathbf{G} : \mathbf{T} \rightarrow \mathbf{C}$.

Test samples, sometimes called *previously unseen* samples, have unknown classes which must be inferred by generalizing somehow from the evidence of the training data. We will discuss the problem of classifying a single test sample \mathbf{x} but eventually we expect to batch-classify large sets \mathbf{X} of test samples.

Classification problem

Given \mathbf{T}, \mathbf{G} , and a previously unseen sample \mathbf{x} ,
find the most probable class c for \mathbf{x} .

Adopting a naive Bayesian approach, we choose $c_{max} = \operatorname{argmax}_{c_j \in \mathbf{C}} \{P(c_j|\mathbf{x}, \mathbf{T}, \mathbf{G})\}$ where $P(c_j|\mathbf{x}, \mathbf{T}, \mathbf{G})$ denotes the posterior probability of class c_j given the new observation \mathbf{x} and the prior knowledge expressed in training data \mathbf{T} and \mathbf{G} .

K Nearest Neighbors algorithm

INPUT: $\mathbf{T}, \mathbf{G}, k \in \mathbf{N}^+$, and \mathbf{x}

OUTPUT: $c \in \mathbf{C}$

Step 1. within \mathbf{T} , find k nearest neighbors of \mathbf{x}

that is, find a set $Y \subset \mathbf{T}, |Y| = k$, s.t.

$$\forall \mathbf{y} \in Y \quad \forall \mathbf{z} \in \mathbf{T} - Y \quad \|\mathbf{x} - \mathbf{y}\| \leq \|\mathbf{x} - \mathbf{z}\|$$

Step 2. within Y , use \mathbf{G} to find the most frequently occurring class, breaking ties at random.

For any particular set of features, assuming that the training data is representative, the kNN algorithm enjoys a valuable theoretical property: as the size of the training set increases, the error rate of kNN approaches no worse than twice the Bayes error,^{DH73} which is the minimum achievable by any classifier given the same information. This remarkable performance is due, fundamentally, to its not committing

to a specific parametric model for the per-class distributions. It also generalizes directly to more than two classes (unlike several competing methods such as neural nets and classification trees). Finally, it has often been competitive in accuracy (if rarely in speed) with more recently developed methodologies such as support-vector machines. For these reasons, we consider kNN to be a nearly ideal classifier for this problem, were it not for the high cost in both time and space of naive implementations and the difficulty of crafting algorithms that approximate its performance in high dimensions.

While in low dimensions kNN can be solved relatively inexpensively in both space and time using simple methods, as dimensionality increases the time and space required to solve the problem increases extremely rapidly. Many techniques for speeding up kNN exist, including editing—later we will discuss why most of these do not meet our needs. A popular, practical approach to approximate kNN in higher dimensions is Bentley’s k -d trees.^{BF76} Under some restrictive assumptions on the distribution of the input, the expected run time of the k -d tree is logarithmic in n , but in the worst case for some distributions runtime can be as bad as linear. Clarkson^{Cl88} discusses a randomized algorithm for closest-point queries in fixed dimensions that is based on finding Voronoi diagrams of randomly selected subsets of samples. While the algorithm’s expected time is $O(\log n)$, in the worst case the space requirements grow as $O(n^{d/2+\gamma})$.

Arya and Mount^{AM93AMN+98} propose that by considering approximate nearest neighbors instead of exact neighbors, they can achieve efficient performance in time and space, regardless of data distribution. They define an approximate nearest neighbor as this: Given ϵ greater than 0, a point p is a $(1+\epsilon)$ -nearest neighbor of a query point q if the ratio of distances from q to p and from q to its nearest neighbor is at most $(1+\epsilon)$. Their implementation of this algorithm uses randomization to preprocess data in $O(n^2)$ expected time and store the result in a data structure of size $O(n \log n)$ in the worst case. This results in $(1+\epsilon)$ -nearest neighbor queries being answered in $O(\log^3 n)$ time. However, their approach is not practically competitive with other methods when the dimension becomes moderately large or the acceptable error term too low. Further papers on this topic explore reducing the constant terms in both space and time with the goal of providing $O(\log n)$ query time and $O(n)$ space.

Beyer, Goldstein, Ramakrishnan and Shaft^{BGRS98} discuss the effect of high dimensionality on searching for nearest neighbors. As others have observed, it is sometimes the case as dimensionality increases the distance from a sample to the nearest data point to it approaches the distance to furthest data point. At these higher dimensions, they suggest that a simple linear search may perform as well or better than more complex indexing techniques. They report on experiments that shows this behavior is triggered at as few as 10 to 15 dimensions on data that is not independent and identically distributed. Special instances are known where kNN is applicable in high dimensions are also discussed, verifying that this problem is not inherent to all data sets. They also emphasize methodological issues in evaluating high dimensional indexing schemes, as some methods may perform well on specialized cases but may be outperformed by linear search on others. This is relevant to our research, as feature selection is not something we believe to be as important as the training data used and while we currently may use near 50 features for classification, early results suggest the intrinsic dimensionality of the data is much lower than that.^{BMN+06}

Song and Roussopoulos^{SR01} investigate approaches for finding k nearest neighbors for a *moving* query point. In the context they discuss, the query point moves in a continuous fashion, leading to the obvious assumption that there is some similarity between consecutive queries. Therefore it should not be necessary to launch a complete kNN search for each data point, instead they develop methods to try to reuse the result of the previous query. If the previous result is not directly applicable they try to use it to guide the search for the next point. They claim their algorithms reduce disk page accesses by nearly 70% than existing methods. A further refinement that improves performance occurs when the position of the next query point can be precisely predicted. This relates to our problem, since if the pixels we are testing are not randomized, they come from the same document and the next pixel to be tested is known. More papers will be discussed later regarding exploiting style consistency.

Clarkson proposes a new data structure, called an RPO tree (Randomized Post Office), for a randomized algorithm for closest-point queries^{Cl88, Cl94}. The primary step in the construction of the RPO tree is the creation of Voronoi diagrams for a random sampling data points and the triangulation of that diagram.

The expected time to build an RPO tree is $O(n^{(1+\epsilon)d/2})$ and results in any query being answered in $O(\log n)$ worst-case time. Worst case space required by an RPO tree is $O(n^{(1+\epsilon)d/2})$.

- Locality Sensitive Hashing

This collection of papers^{DIIM04aDIIM04bGIM99IM98} lead up to the proposal of a solution to the Approximate Nearest Neighbor Problem using a Locality-Sensitive Hashing scheme. The Approximate Nearest Neighbor Problem is a similarity search problem that tries to find the most similar object to a query object, where objects are defined by a set of features that map to set of points in an attribute space. They propose specifically a solution to the search problem in high-dimensional spaces, as search in few dimensions is well-solved. However, as the number of dimensions grow sufficiently large, most algorithms and techniques perform not significantly better than a simple brute force linear search. A method to get around this “curse of dimensionality” is to use approximation, assuming an acceptable trade-off exists between finding the exact nearest neighbor, or more quickly, an approximation that meets a certain level of quality.

Early work^{GIM99IM98} by these authors suggest using tree structures to partition the search space, using a form of hashing called locality-sensitive hashing (LSH) to subdivide the search space for the approximate high dimensional nearest neighbor problem. The key idea is to “hash the points using several hash functions so as to ensure that, for each function, the probability of collision is much higher for objects which are close to each other than for those which are far apart”.^{DIIM04a} However the potential applications of this early work was very narrow in scope as LSH functions were suggested for data with solely binary features. While showing that experimentally this method provided a large speedup over some tree like structures and a sub linear dependence on data size, this was only true for binary data while query time and algorithm complexity increased greatly for higher dimensions.

The most recent work^{DIIM04aDIIM04b} works on the same approximate nearest neighbor problem as before, however it is extended to use data in Euclidean space (not Hamming space) and makes the improvements of reducing the exponent in the analysis of the query time and works on any L_p norm (where $0 \leq p \leq 2$). Like earlier schemes, it retains the properties that it works well on sparse, high dimensional data and that it “reports the exact near neighbor very quickly, if the data satisfies a certain bounded growth property”.^{DIIM04a} The definition of a LSH family of functions is described in.^{IM98} Most recent work^{DIIM04a} presents a LSH family of functions based on p -stable distributions to be used to solve the near neighbor problem. “Stable distributions are defined as limits of normalized sums of independent identically distributed variables.”^{DIIM04a} As mentioned before, the general idea of these papers is to use some hashing scheme that will places instances that are “physically” close to each other, and therefore neighbors, into the same hash buckets. These p -stable distributions are used to facilitate this by reducing the dimensionality of feature vectors by “sketching” them. This is done by generating a random set of vectors α from a p -stable distribution of the same dimension d as the feature vectors. A sketch of a feature vector then is defined by the dot products of each α with that vector, effectively reducing the representation of a feature vector from d unique coordinates to a set of distances from these vectors alpha that are shared with all other data points. Each of these projections can be treated as hash values and for each α , for a given vector, can be concatenated together. This hash should be locality sensitive, that is, it is their intention that a vector will collide with another (share the same hash value) if they are close to each other.

Their analysis follows by synthetically generating data that meets the unique criteria that each data point has exactly one unique near neighbor within $(1+\epsilon)*R$ and comparing with the performance of k -d trees. While processing and query times improve on k -d trees, no claims are made regarding performance on real world data that does not exhibit the unique constraints their synthetic data does.

- Adaptive k -d Trees

One multidimensional search technique with broad applicability is Bentley’s k -d trees.^{Ben75} The variant of k -d trees most relevant to classification seems to be the *adaptive* k -d tree.^{FBF76} Briefly, these partition the set of points recursively in stages: at each stage one of the partitions is divided into two sub partitions; we will assume that it is possible to choose cuts that achieve *balance*, that is, to divide into sub partitions containing roughly the same number of points. Division is by cutting along one of the dimensions $i \in \{1, \dots, d\}$, *i.e.* by choosing a threshold value and assigning each of the partition’s points x to sub partition

(a) or (b) according to whether its x_i component value is (a) less than, or (b) greater than or equal to the threshold. (In some implementations, the threshold corresponds to a component value for one of the points, which is then stored in the interior node of the search tree; but we will assume here that all points are stored in leaf nodes). Generally, at each stage a different dimension is cut: one simple strategy is to cycle (and if necessary recycle) through the dimensions in a fixed order; another strategy is to cut the currently most populous partition. Cutting proceeds until all partitions contain few enough points to invite a final fast sequential search. The final partitions are generally hyper rectangles (not always hyper cubes) with orthogonal sides (parallel to the coordinate axes of \mathbf{R}^d).

Balancing each cut ensures that find operations execute in $\Theta(\log n)$ time in the worst case. Consistent with a guarantee of such logarithmic-time finds, Bentley’s k -d tree construction achieves a minimum number of cuts and thus a minimum number of partitions (close to n/p , where p is the number of points in the final partitions; in our context, assuming $p = 10$, this is still huge, $\approx 10^8$). The threshold value chosen for each cut depends on the distribution of points within the partition to be cut, so the thresholds are not *independently predictable*: that is, none (after the first) can be computed without knowledge of the cut thresholds in some earlier stages. Further, given a previously unseen x it is not possible to compute the d upper and lower bounds of its k -d hyper rectangle (within which it lies) without traversing the k -d tree. Thus locating x ’s k -d hyper rectangle *requires* $\Theta(\log n)$ time.

The pruning power of k -d trees speeds up range searches. Given a query point x and a radius r , defining a search hypercube, it is straightforward to generalize the find algorithm to explore all k -d tree nodes whose subtrees overlap the search hypercube. The asymptotic runtime of such variants has been studied: ^{LW77} reports that the worst-case number of tree nodes explored is $\Theta(d n^{1-1/d})$. In our context, this may (or may not) promise much improvement over brute-force search, since (neglecting the multiplicative constant) $d n^{1-1/d} \approx 100(10^9)^{0.99} = 100(10^{8.91}) = 10^{10.91} \gg 10^8 \approx n$. Of course this may be a pessimistic bound for several reasons. Whether or not k -d tree range searches are efficient for classification may ultimately be decidable only by experiment.

- Pixel Representation of Images for Classification

In work on off-line handwriting segmentation, ^{Bre} Thomas Breuel proposes a pixel-based representation of a segmentation for the purposes of classifying handwritten characters. One of the benefits he discusses of this approach is that each pixel-based representation can be implemented as 24 bit RGB color images, meaning he can use 2^{24} potential labels for classes, more than sufficient to represent any segmentation encountered. Using pixels for segmentation is easy to implement and also provides a common basis for comparing different methods or algorithms.

- Isogeny, Style Consistency

Sarkar ^{Sar00SN01SN} researches improving classification by exploiting style consistency. He assumes that in many classification problems, there are groups or fields of patterns that have a common origin. For example, in handwriting lines or pages from the same writer will have some consistency or in printed text the same font may be used. Measurements or features on groups like these will be similar and indicate a common origin, and therefore not independently distributed. Instead they are related through an underlying *style* that can be attributed to the origin and in common classification techniques, fields are classified without considering these inter-field relationships. Sarkar proposes a *style-conscious classification* that will “classify entire fields at a time so that each pattern can affect and guide the classification of other patterns in the field by furnishing information about the underlying style, in addition to any linguistic context.”

- Editing

We must acknowledge well-studied “editing” ^{DH73} methods which prune training samples with little or no loss of classification accuracy, and often significant speed-ups—but this thesis will not carry out an exhaustive set of comparison experiments in this large space. These methods are certainly relevant to our research, but we feel they will be complementary to our methods. If they are run separately on our data, as preprocessing, they may improve our results, but fundamentally our approach will be different in detail

and does not compete with editing. Most editing methods are sub-optimal and suffer from poor asymptotic run time performance.

3. THE PROPOSED APPROACH

Having now introduced the topic and briefly discussed the background literature, I will now discuss our ideas for approaching this problem and what form we expect a “solution” to take.

As stated before, the goal of this research is classification of document images in nearly constant expected time. Regardless of implementation or algorithms used, our primary point of emphasis will be *versatility*, as we want to make some useful classification for all types of documents. We have no interest in over-specialized, brittle algorithms that perform with high accuracy on only select, limited classes of document images. Instead we believe it is more useful to make a potentially less accurate decision about the classification of the most difficult to classify document images. Later, we will discuss how our ideas will facilitate this.

Under the two assumptions described before, that generally larger training sets allow for better classification than smaller ones and the availability of vast amounts of data we will also focus on investigating *voracious* classifiers that use potentially **billions** of training samples. A potentially novel aspect to our research will be what we consider a training sample and at what level we choose to classify document images. Unlike most approaches that choose to classify content in typically rectilinear zones, we will perform classification on the pixel level.^{Bre} This is done to avoid the potential arbitrariness of using shapes to zone documents that may not always correspond exactly to page layout. We also feel that this decision will be more complementary to the methods we will use for classification and in the end will be more useful for classification at lower accuracies. This decision will also allow us to easily achieve on the order of billions of training samples as each document image can contain millions of pixels.

We have gathered sample page images containing the following types of content: handwriting (HW), machine Print (MP), line Art (LA), photos (PH), math notation (MT), maps (MA), engineering drawings (ED), chemical drawings (CD), “junk” (JK, e.g. margin and gutter noise), and blank (BL). This list of document image types was derived to capture the versatility that we hope to capture in our algorithms. By choosing these classes, we hope to represent a sufficiently large space of possible document image types, while not focusing on any image type that is too specialized or of narrow interest.

The ideal choice of a classifier for our research would be k-Nearest Neighbors (for reasons described in literature review). However the cost of a purely naive, brute-force implementation may make that implausible, but ideally our classifiers should come as close as possible to the performance of kNN. Published worst-case asymptotic analyses of such algorithms are discouraging, predicting exponential time or space performance. However, we are not aware of any realistic analysis of expected-case performance, and some methods seem to work well in practice. One new and promising approach to near kNN performance in constant expected time is via hash-driven table lookup. This is made possible by our assumption that the distribution of data from document images will be highly non-uniform and only populate small, dense “pockets” in high-dimensional feature space. Therefore, the actual number of hash buckets that are filled (contain any data at all) should be much lower than the total possible number of buckets and manageable on current hardware. Early experiments^{BMN⁺06} seem to suggest this, as the number of occupied buckets appear to grow initially exponentially as expected, before settling into a region of less than exponential, cubic growth, before plateauing. It is also in this area of cubic growth where we see the biggest trade off between the speed up of using fewer bits to address the data and the number of pixels correctly classified. For example, on documents categorized by 15 features, using 10,000,000 training samples and 50 bits used in the bit-interleaved address for hashing, the absolute number of bins actually occupied was 2,100,000 which is of course easily manageable by single-stage hashing where the hash table is stored in main memory. As will be discussed later, this lends itself to the idea of a two staged hashing scheme, keeping frequently paged tables for a document in memory and the rest of the data on disk. This cubic growth feature of the data from document images also appears to be a property of the actual distribution of data, not as a result of an arbitrary choice of algorithm used.

Another attractive strategy is batch processing to exploit similarities among samples. This might help because, for example, pages within the same document have the same or similar origin: they are *isogenous*,

having been generated by similar processes. Even more strongly, from pixel to neighboring pixel within an image, we expected to see very little change since they result from virtually identical conditions. While we stated that we believe that training set size contributes more to the success of a classifier than the selection of features, this still underscores the need to select “good” features that will allow to us to see and exploit such similarities. Another belief that we have is that the intrinsic dimensionality of our data will be much lower than the actual number of features we use, making our approaches feasible. Combining batching with hashing seems to be a new idea in the Nearest Neighbor literature and it invites applying modern paging techniques so that frequently used parts of the hash tables are more likely to be maintained in main memory.

There are some issues and beliefs in this proposal that we will not explicitly focus on. While we are motivated by a belief that training and testing on extremely large representative sets of samples is critically important, and while we will experiment with large training sets, this thesis will not focus on the underlying methodological problems of collecting representative samples, ground-truthing, etc. It has been proposed that training sets can be artificially amplified (*e.g.* using pseudorandom variations and interpolations), and certainly such methodologies are one way to provide vastly larger, and arguably more systematically representative, training sets for which our technology is designed—but, again, such methods are not the focus of this thesis.

However, there are a very hard set of problems at the foundation of this thesis that are clearly difficult, but we will not try to avoid them. How can we carry out realistic expected time and space complexity analyses of these algorithms? Published analyses rely on simplistic models and assumptions about the distributions of data (mixtures of Gaussians, etc). These analyses yield pessimistic expected performance results in time and space. However, in practice some of these methods perform much better than their worst case performance assumptions. How is it possible to characterize the real distributions of data and rigorously predict the observed performance?

4. QUESTIONS TO BE INVESTIGATED

Some immediate steps to be taken in this research follow:

1. Extend previous experiments in the DICE project.^{BMN⁺06} Previous work used testing and training samples from a limited set of content types and document images. Currently, training only occurs on three content types, all in grayscale: handwriting, machine print and photographs. We would like to immediately expand testing to include full color images with more content types, such as maps, math equations, line art, etc. Current work has also trained and tested on images with almost exclusively rectangular zones and we believe a strength of our methods will be processing images with any type of layout so we would like carry out new experiments with samples with more complex or unique layouts. Also, early experiments did not train on data classified as blank. Output from these experiments express the necessity of this as margins of documents of grayscale images were not left unclassified and were instead forced to one of the three content types that had been trained. Can training of the blank content type be done through synthetically creating blank documents of varying “background” color pixels or most images also have regions like margins be zoned as blank?
2. Continue the collection of document image samples. We have attempted to collect images from as wide a range of sources as possible to ensure that a variety of each image type was collected. Sources gathered so far include:
 - University of Washington Document Image Data Bases CD-ROMs (UW)
A three volume collection of ground truthed document images for Optical Character Recognition (OCR) research in English and Japanese. It was developed by the Intelligent Systems Laboratory, at the University of Washington. This collection is the most thorough in the sense that each document image is zoned and has a text ground truth associated with each zone, as well as bounding box coordinates and other information about other page attributes including qualitative information on the condition of each page.

- (a) Volume 1^{PH93}
This is a two CD-ROM set containing 1147 document page images from English scientific and technical journals. Included in the collection are binary images scanned from various generation photocopies, binary and grayscale images scanned directly from technical journals and synthetic images generated from LaTeX files.
- (b) Volume 2^{PH95}
This collection contains 624 English journal pages images and 477 Japanese journal page images. All of the images are binary. The same information is provided for each image as in Volume 1
- (c) Volume 3^{PHC95}
This collection contains 25 pages of chemical formulae, 25 pages of mathematical formulae, 40 pages of line drawings and 44 images of engineering drawings.
- Information Science Research Institute at UNLV OCR Performance Toolkit Data Sets^{atUoNLV96} (UNLV)
This collection contains 2889 zoned and ground truthed document images. Most images are provided in both bitonal and grayscale formats. The collection is primarily English documents with a small sample (144 images) of Spanish newspaper articles. The English document images come from newspapers, magazines, business letters, legal documents and Department of Energy documents. Associated with each image is a manually created ground truth file with manually created zone information.
- American Memory^{oC} (AMem)
This is the digital archive of the Library of Congress for public domain media. Contained are millions of images of historical documents. Content is in varying formats and resolutions. Currently we have collected 472 images from this source of content types that we did not have good examples of from other sources, focusing primarily on handwriting. Naturally, none of these images are zoned or have corresponding ground truth.
- Indian Statistical Institute, Kolkata Mathematics Notation Collection^{Gar} (ISI)
This is a dataset of 102 binary isolated mathematical expressions with corresponding ground truth and bounding box information.
- Lehigh University Digital Library^{LS}
This is the online collection of Lehigh University's library containing selections digitized to support teaching and research at Lehigh. We have collected 1342 images from this collection in grayscale and color of historical manuscripts and engineering drawings. None of the images are zoned or have corresponding ground truth.
- New York Public Library Digital Gallery^{Lib} (NYPL)
Similar in content and mission as American Memory that contains over 415,000 document images. No images have been collected locally yet from this collection.
- Astrophysics Data Systems Historical Scans(ADS)
This is a NASA-funded collection of digitized images historical observatory publications. 404 binary images containing mathematics, tables and engineering drawings have been collected locally.
- Sofia-Munich Corpus (Mnch)
30+ GB Corpus of scanned Bulgarian and German documents for OCR evaluation. This was recently received and has not yet been added to local collection.
- National Institute of Standards (NIST)
- National Library of Medicine (Med)

We began our search with images that had previously been zoned and converted those zones to our list of content classes. However, zoned images are a rare commodity and we had little luck in finding them other than from the University of Washington Databases, the UNLV datasets and smaller, more specialized sets like the ISI collection, which consists solely of mathematical equations. As databases were located and images collected we maintained a taxonomy of the image types as a matrix, seen below:

	Bitonal	Grayscale	Color
Handwriting	AMem, UNLV, NIST	AMem, UNLV, DL	AMem, NYPL
Machine Print	UW, AMem, UNLV	UW, UNLV, AMem, DL, ADS	AMem
Line Art	AMem, DL, UNLV	AMem, DL, ISRI	AMem
Photos	UNLV	UNLV, AMem, DL	AMem, DL
Math Notation	ISI, UW	ADS, DL	N/A
Maps	NYPL, AMem	AMem	Memory
Engineering Drawings	UW	DL	N/A
Chemical Diagrams	UW		N/A
“Junk”	All	All	All
Blank	All	All	All

- Carry out experiments in comparing different classification methods and running them in competition. We have downloaded and installed Indyk’s code for LSH^{DIIM04a} and will now convert our data to be compatible with it. Finish implementation of Classification and Regression Tree (CART)^{BFOs84} methods. Compare these methods with our implementation of brute force kNN and Casey’s implementation of static k -d trees using bit interleaving addressing.^{Cas06}
- Plan experiments to characterize the strengths and weaknesses of these methods with regards to isogenous data. Experiment with training classifiers on samples from all content types and only test on one class at a time. Train on specialized, limited inputs and test on all content types. Which performs better? Can we automatically adapt inputs based on what we want to test on?
- Explore implications of skewed data distributions. The worst-possible case scenario for our methods would find the data to be uniformly distributed in the feature space. However, we assume, and early experiments suggest, that training data is skewed (meaning there is some non-uniform clustering of data). We want to explore how we can exploit this skewed distribution of data. One of our proposed ideas is to combine paging schemes with hashing classifiers. If data from all document images are uniformly distributed, then of course paging will be of no use. We expect isogenous subsets of data to be even more skewed than (random? normal?) data and therefore makes using paging schemes plausible by running some simple simulations.

First, consider classifying two document images. The first document image is an arbitrary image, for example a journal article. The second document image is of the same dimensions and resolution, however is synthetically created from snippets of a dozen, unrelated images from different sources, however all of the same content type. Now, as we classify we keep a count of the number of hash tables paged in for each image. As a result of isogeny, the first image, a single document from a single source, should have paged in fewer tables than the number seen by the image consisting of multiple documents from multiple sources. This experiment can be extended to multiple content types, though we would not expect to see as large a difference in the number of tables paged as there is great variability within each content class.

Assuming we find that isogeny leads to some tables being looked at more frequently than others, we can experiment with implementing a two stage hashing scheme. First, assume the hash tables we build will be so large that they must be stored on disk, so we begin by training and storing the resulting hash tables on disk. We then create a second, smaller hash table in main memory that uses a different hash function than the table on disk. When we test a sample we first hash into the table in memory using its function. If we find that cell is empty, we then rehash into the table on disk and retrieve all of the data in that cell and store than in memory using the smaller table’s hash function. The local hash table will continue to grow until it is no longer necessary to look on disk (frequently) for a document. Eventually, when a new document image is encountered, we will have to start bringing in large number of pages from disk again which may fit into the local table or may collide with existing data there. At this point, we can throw out the local table and start to repopulate it. How do we know when throw away the local hash table? For example, when is it too full because it is just not large enough for the data associated with the current image? In this case, we might not want to throw away the entire table since we will likely look at that same data while we are classifying that current image. However, when is it too full because we are now

hashing data completely unrelated to a previous image? This is when we should throw out the existing table and start over. Are there some tables that will be frequently paged by most types of documents and should always be kept locally? What is the speedup of keeping this local hash table and having to throw it away and repopulate it when a new class of documents is encountered and what is its tradeoff against the overhead of maintaining it?

6. Scale up the number of features used. While we currently use roughly 30 features to classify data, in the future this must scale up to maybe near 100 to allow for classification of more distinct content types. However, we believe the intrinsic dimensionality of the data is much lower and we would like to investigate how to characterize it and the effect that it has on the feasibility of using different classification schemes.
7. Experiment with batch pre-processing of test data. Traditional approaches to NN problems, like the methods we have proposed, begin with the preprocessing of training data into a structure that is then stored on disk for later use. This structure allows the neighbors of testing data to be more quickly computed than if training data was simply stored sequentially on disk. Testing data is processed as it is encountered and in general is not sequenced or organized in any unique way. However, what if we were to reverse this process and instead begin by looking at the testing data and storing it in our hash tables on disk. Now, to classify each point, we would begin by reading in training samples sequentially for each test sample and stopping once we have collected enough neighbors to classify it. Obviously, problems arise from this approach, such as repeatedly searching through the entire training volume when an image with samples that have never been seen before is encountered. Instead, what if we perform preprocess the training data as usual, store it to disk and then perform the same processing on testing data and store that in a separate hash table on disk. Now, we should be able to speed up testing as we can process test samples in batches, as those stored in the same cell will be classified by the same training samples. Assuming that the feature space is populated by small dense pockets of samples, this should result in a significant reduction in the number of classifications that must be made.

5. CONCLUSION

If such a research program as proposed is successful, it would result in the following...

- fast approximate kNN algorithms, interesting to the pattern recognition, machine learning, artificial intelligence, and computer vision research communities
- a case study of a voracious classifier technology able to learn from vast training sets on an Internet scale
- remarkably versatile computer vision technology, contrasting with competing overspecialized, brittle methods
- fast, broadly applicable document image classifier would be of interest to digital libraries, web search, and intelligence communities

REFERENCES

- [AM93] Sunil Arya and David M. Mount. Approximate nearest neighbor searching. In *Proceedings 4th Annual ACM SIAM Symposium on Discrete Algorithms*, pages 271–280, 1993.
- [AMN⁺98] Sunil Arya, David M. Mount, Nathan S. Netanyahu, Ruth Silverman, and Angela Y. Wu. An optimal algorithm for approximate nearest neighbor searching in fixed dimension. *Journal of the ACM*, 45(6):891–923, November 1998.
- [atUoNLV96] Information Science Research Institute at the University of Nevada Las Vegas. Ocr performance toolkit data sets. <http://www.isri.unlv.edu/ISRI/OCRtk>, 1996.
- [Ben75] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517, 1975.
- [BFOS84] Leo Breiman, Jerome H. Friedman, Richard A. Olshen, and Charles J. Stone. *Classification and Regression Trees*. Wadsworth& Brooks/Cole, Pacific Grove, CA, 1984.

- [BGRS98] Kevin Beyer, Jonathan Goldstein, Raghu Ramakrishnan, and Uri Shaft. When is nearest neighbor meaningful? In *ICDT*, pages 217–235, 1998.
- [BMN⁺06] Henry S. Baird, Michael A. Moll, Jean Nonnemaker, Matthew R. Casey, and Don L. Delorenzo. Versatile document image content extraction. In *SPIE IS&T Document Recognition & Retrieval XIII Conf.*, San Jose, CA, 2006.
- [Bre] Thomas M. Breuel. Representations and metrics for off-line handwriting segmentation. In *8th International Workshop on Frontiers in Handwriting Recognition*, pages 428–433.
- [Cas06] Matthew Casey. Fast approximate nearest neighbors. Master’s thesis, Lehigh University, (to appear) April 2006.
- [Cla88] K. L. Clarkson. A randomized algorithm for closest-point queries. *SIAM J. Comput.* 17, 1988.
- [Cla94] K. L. Clarkson. An algorithm for approximate closest-point queries. In *Proceedings of the 10th Annual ACM Symposium on Computational Geometry*, pages 160–164, 1994.
- [DH73] Richard O. Duda and Peter E. Hart. *Pattern Classification and Scene Analysis*. Wiley, New York, 1973.
- [DIIM04a] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proc. 42nd Annual ACM Symposium Computational Geometry*, pages 253–262, 2004.
- [DIIM04b] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S. Mirrokni. *Locality-sensitive hashing using stable distributions*, chapter 4. 2004.
- [FBF76] Jerome Friedman, Jon L Bentley, and Raphael A Finkel. An algorithm for finding best matches in logarithmic expected time. Technical report, Stanford, CA, USA, 1976.
- [Gar] Utpal Garain. A sample corpus for ocr of printed mathematical expressions. <http://www.isical.ac.in/utpal/pdf/MathOCR.zip>.
- [GIM99] Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Similarity search in high dimensions via hashing. In *Proceedings of the 25th VLDB Conference*, Edinburgh, Scotland, 1999.
- [IM98] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*, pages 604–613, New York, 1998. ACM.
- [Lib] New York Public Library. Digital gallery. <http://digitalgallery.nypl.org/nypldigital/index.cfm>.
- [LS] Lehigh University Library and Technology Services. Lehigh university digital library project. <http://digital.lib.lehigh.edu/>.
- [LW77] D. T. Lee and C. K. Wong. Worst-case analysis for region and partial region searches in multidimensional binary search trees and balanced quad trees. *Acta Inf.*, 9:23–29, 1977.
- [oC] Library of Congress. American memory. <http://memory.loc.gov/ammem/>.
- [PH93] Ihsin Phillips and Robert Haralick. University of washington english document image database i. CD-ROM, 1993.
- [PH95] Ihsin Phillips and Robert Haralick. University of washington-ii english/japanese document image database. CD-ROM, 1995.
- [PHC95] Ihsin Phillips, Robert Haralick, and Bhabatosh Chanda. University of washington-iii english/technical document image database. CD-ROM, 1995.
- [Sar00] Prateek Sarkar. *Style Consistency in Pattern Fields*. PhD thesis, Rensselaer Polytechnic Institute, 2000.
- [SH03] Abigail J. Sellen and Richard H. Harper. *The Myth of the Paperless Office*. MIT Press, 2003.
- [SN] Prateek Sarkar and George Nagy. Classification of style-constrained pattern-fields. In *15th International Conference on Pattern Recognition*, year = 2000.
- [SN01] Prateek Sarkar and George Nagy. Style-consistency in isogenous patterns. In *Sixth International Conference on Document Analysis and Recognition*, 2001.
- [SR01] Zhexuan Song and Nick Roussopoulos. K-nearest neighbor search for moving query point. In *SSTD 2001*, pages 79–96, 2001.