Information Extraction among Symbolically Compressed Image in Database

[1]Rashmi M.Choudhari, [2]Dr.D.M.Dakhane

[1]Student, [2]Assistant professor

[1]Department of Computer Science and Engineering,

[1]Sipna College of Engineering, Amravati, India

Abstract: The detection of duplicate images is a useful means of indexing a large database of documents. An algorithm for duplicate document detection is proposed in this papar that operates directly on images that have been symbolically compressed using techniques related to the ongoing JBIG2 standardization effort. This report describes a hidden Marko model (HMM) method that recognizes the text in an image by deciphering data from the compressed representation. Experimental results show that it can recover better than 90% of the text in compressed document images and that this is sufficient to identify duplicates in a large database. This describes a method for duplicate detection on symbolically compressed document images. This paper describes a method for duplicate detection on symbolically compressed document images. It recognizes the text in an image by deciphering the sequence of occurrence of blobs in the compressed representation. We propose a Hidden Markov Model (HMM) method for solving such deciphering problems and suggest applications in multilingual document duplicate detection.

***IndexTerms* - Duplicate detection, Document indexing, Symbolic compression, IM^3.**

I. Introduction

Document matching is an important component of a document image storage system, allowing for removal of duplicates, copyright violation detection and other applications. Since document images are usually stored and transmitted in compressed formats, considerable advantages are realized by performing the matching process directly on compressed images [5][10]. One technical challenge is the extraction of meaningful information from compressed data.

Duplicate detection is particularly important for large document databases like that produced by the Infinite Memory Multifunction Machine (IM) .The IM^{3}system (Hull and Hart, 1998; Hull et al., 1999) captures a copy of every printed, copied, or faxed document generated in an office. This guarantees that almost any document a user might need will be available when they need it. This concept was developed after it was observed that even though the obvious method for document capture, namely scanners, were commonly available, they were not commonly used.

The dramatic increase in the use of document images in recent years has led to research in compression technology for textual images. Symbolic compression schemes preserve much of the structure in a document image thereby facilitating feature extraction. They cluster individual blobs in a document and store the sequence of occurrence of clusters and representative blob templates. This kind of compression scheme was originally proposed binary images of text [1]. Lossless compression algorithms can be designed around this idea by supplementing a coding scheme for the residuals that result from pattern matching [11]. It has been shown that such separate coding of patterns and residuals achieves better compression ratios than conventional methods. Numerous algorithms based on the pattern matching approach such as JBIG2 [4] and others [8][7] have been proposed recently.

Economic considerations are always important factors when users decide whether to adopt new technologies. An important consideration in the design of the IM^3 system was the relative cost of printing a document on seminar vs. storing an image of the same document on magnetic disk. Of course, there is a wide variation in the price of seminar and toner needed to print the range of documents encountered in the typical office. For the purposes of this analysis, it was assumed that, on average, the cost of an 8.5x11 inch sheet of seminar is one cent. It was also assumed that a 400 dpi binary image of the same document on average would require 100 KB. At the time the IM^3 project was started (late 1993) it was observed that it cost about 3 cents for 100 KB of magnetic disk storage. This was just for the disk space. It did not take into account the cost of the computer, etc. However, we projected that over time these costs would significantly decrease and eventually become less than the cost of a sheet of seminar. That time arrived sometime in 1996. Today, it costs about 0.27 cents for 100 KB of disk space. This is significantly less than the cost of a sheet of seminar. The 4:1 difference in cost of the two media now favors the adoption of a document storage and retrieval system like the IM^3.

Lost documents are a significant problem for office workers. A recent study estimates that at any time 3% to 5% of documents are lost and the average cost of lost documents to a Fortune 500 company is in the range of $3 million to $5 million [10]. An obvious solution to this problem is for users to scan all their documents. However, for any number of reasons, not the least of which are users' natural reluctance to alter their work practices, such an approach has not been widely adopted [3]. In fact, recent results indicate that for new retrieval techniques to gain wide acceptance they should be easy to use, be familiar and require as little user effort as possible.

II. INFINITE MEMORY MULTIFUNCTION MACHINE IM^3

Lost documents are a significant problem for office workers. A recent study estimates that at any time 3% to 5% of documents are lost and the average cost of lost documents to a Fortune 500 company is in the range of $3 million to $5 million [10]. An obvious solution to this problem is for users to scan all their documents. However, for any number of reasons, not the least of which are users' natural reluctance to alter their work practices, such an approach has not been widely adopted [3]. In fact, recent results indicate that for new retrieval techniques to gain wide acceptance they should be easy to use, be familiar and require as little user effort as possible [6].

The Infinite Memory Multifunction Machine (IM3) isa document storage and retrieval system that solves alarge portion of the problem of lost documents [6]. It captures a copy of every printed, copied, or faxed documentgenerated in an office. This guarantees that almost any document a user needs will be available when they needit. This concept was developed after it was observed thateven though the obvious method for document capture,namely scanners, were commonly available, they werenot commonly used. The IM3 makes document capture effortless by saving an electronic copy of *every* document that users copy, print, or fax. Furthermore, users are not asked whether any particular document should be captured -- no consciousdecision is required at capture time. Thus, every person in an office that copies, prints, or faxes a document automatically contributes data to the IM3.

Economic considerations are always important factors when users decide whether to adopt new technologies. An important consideration in the design of the IM^3 system was the relative cost of printing a document on seminar vs. storing an image of the same document on magnetic disk. Of course, there is a wide variation in the price of seminar and toner needed to print the range of documents encountered in the typical office. For the purposes of this analysis, it was assumed that, on average, the cost of an 8.5x11 inch sheet of seminar is one cent. It was also assumed that a 400 dpi binary image of the same document on average would require 100 KB. At the time the IM^3 project was started (late 1993) it was observed that it cost about 3 cents for 100 KB of magnetic disk storage. This was just for the disk space. It did not take into account the cost of the computer, etc. However, we projected that over time these costs would significantly decrease and eventually become less than the cost of a sheet of seminar. That time arrived sometime in 1996. Today, it costs about 0.27 cents for 100 KB of disk space. This is significantly less than the cost of a sheet of seminar. The 4:1 difference in cost of the two media now favors the adoption of a document storage and retrieval system like the IM^3. Lost documents are a significant problem for office workers. A recent study estimates that at any time 3% to 5% of documents are lost and the average cost of lost documents to a Fortune 500 company is in the range of $3 million to $5 million [10]. An obvious solution to this problem is for users to scan all their documents. However, for any number of reasons, not the least of which are users' natural reluctance to alter their work practices, such an approach has not been widely adopted [3]. In fact, recent results indicate that for new retrieval techniques to gain wide acceptance they should be easy to use, be familiar and require as little user effort as possible.

A. IM^3 System Design

Documents stored in the IM^{3}are accessed with a web browser. Each user has a home page that provides a portal to their document collection. A number of techniques are provided for search and retrieval. These include full text search and various methods for browsing based on the dates when documents were captured. A method for duplicate detection would give users a means for retrieving exact copies and other versions of a given document. Version retrieval would be particularly useful when after retrieving a document by full text search with a certain set of keywords; the user would like to retrieve other versions of that document. Even though substantial amounts of text may be common between versions, they might not all share the keywords that were used for full text search.

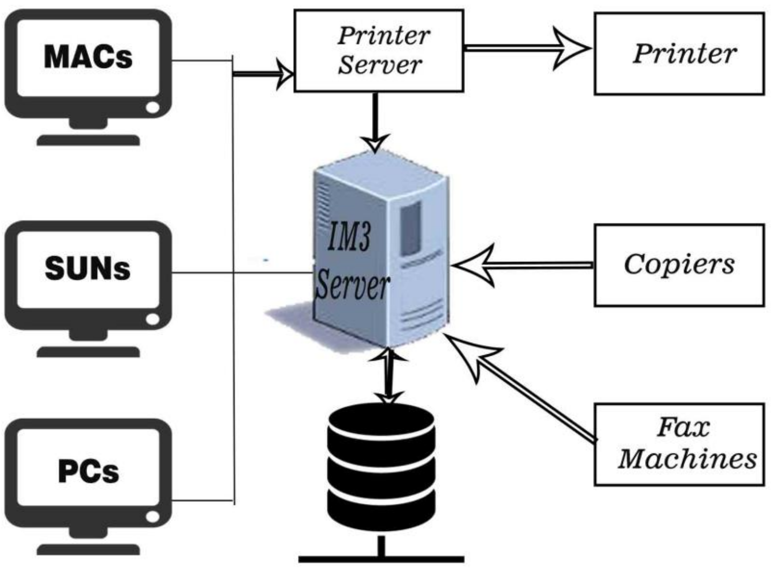

Figure 1: IM^3 System Design

An outline for the design of the IM^3 system is presented in Figure 1. Specially modified digital photocopiers were developed that automatically capture an image of every copied document. Aside from a user identifying himself by pressing a button on a touchscreen, this process is completely transparent. The captured images are transferred to the document server where they are permanently stored and indexed for later retrieval. Print jobs are automatically captured by software running on a Unix print server. A copy of every printed document is transferred to the document server as it is sent to a printer. This is done by a filter in the spooling system that is applicable to jobs printed on PC's, Apple computers, and Unix workstations. In this way the capture of printed documents is completely transparent to the user and is independent of any application software. Every document sent to printers serviced by the Unix server is saved.

An indexing process is applied to every saved document. The images from the photocopiers are OCR'd. Text is extracted from the postscript files for printed documents. This is used to choose keywords for each document and build data structures for full text retrieval. Thumbnail images are also calculated at several resolutions (4 dpi, 8dpi, and 72 dpi) for use in various brows able interfaces

III. DUPLICATES DETECTION AMONG DOCUMENT IMAGES

We assume that all document images are compressed with a ``symbolic" technique (e.g., JBIG2 (Howard et al., 1998)). Features are extracted directly from the compressed version of document images [3]. A comparison procedure determines whether the feature descriptions of any two documents are similar enough for the original documents to be duplicates. Hull et al.introduced a Symbolic compression for binary document images first clusters connected components, which often correspond to isolated characters. A unique identifier is then assigned to each cluster. The compressed document contains one image for each cluster and the sequence of identifiers for the connected components (also called blobs) in the original image[7]. This sequence of identifiers corresponds to the sequence of occurrence of characters in the original document.

The extraction of information from compressed document images is useful since the compression algorithm not only reduces the size of the image, providing less data to process, but also represents characteristics of the original image in the compressed data stream that can be used directly to compute information about original document. CCITT groups 3 and 4 compression are one example. These methods include pass codes in the compressed data stream, which are attached to connected components. The configuration of pass codes in CCITT-compressed document images has been used for skew detection [27] and duplicate detection [12, 18] Symbolic compression has recently been proposed for inclusion in the JBIG2 standard [10]. Symbolic compression methods were first discussed by Ascher and Nagy [1]. More recent works include [9, 17, 28, 30]. In symbolic compression, images are coded with respect to a library of pattern templates. Templates in the library are typically derived by grouping (clustering) together connected components that have similar shapes. One template is chosen to represent each cluster. The connected components in the image are then stored as a sequence of template identifiers and their offsets from the previous component. In this way, an approximation of the original document is obtained without duplicating storage for similarly shaped connected components.

Minor differences between individual components and their representative templates, as well as all other components which are not encoded in this manner, are optionally coded as residuals. An example of symbolic compression is shown in Figure 5. After connected component clustering, the original document image is represented as a set of bitmap templates, "A a h i s t" in this example, their sequence of occurrence in the original image ("0 1 2 1 5 3 4 1 2 1 5 3 4 1 5 1 5"), as well as information about the relative geometric offset between adjacent connected components (e.g., (+2, 0) means the beginning of the second component in the sequence is 2 pixels to the right of the end of the first component), and a compressed residual image. The residual is the difference between the original image and the pattern templates. This data can be compressed with arithmetic coding or another technique. A lossy representation for a symbolically compressed image could be obtained by not storing the residual image. Symbolic compression techniques improve
compression efficiency by 50% to 100% in comparison to the commonly used Group 4 standard [19, 29]. A lossy version can achieve 4 to 10 times better compression efficiency than Group 4 [29].

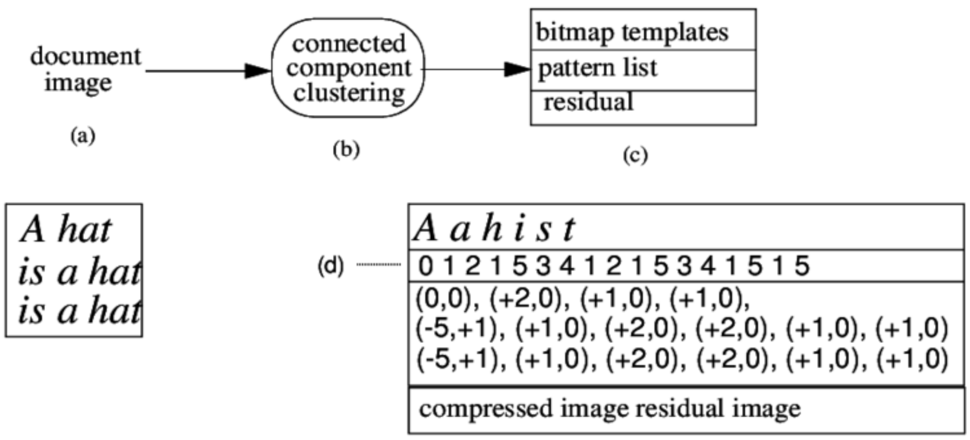

Figure 2: Example of Depiction of symbolic Compressed Images

An idealized example of a symbolically compressed (similar to JBIG2) document image is shown in Figure 3. An original document image is shown (a) as well as its compressed form (b). The unique letters in the original image are represented as individual sub-images and numeric identifiers at the top of (b). The sequence of identifiers shown in Fig. 3(c) encodes the order in which the corresponding sub-images occurred in the original image (a). For example, ``0 1 2 3" are the first four sub-images in this sequence. They correspond to the first four letters in the image, ``PALO". The x±y locations of the sub-images and image residual data are also encoded in the compressedformat.

The characteristic of symbolic compression that we use for duplicate detection is the sequence of cluster identifiers (``0 1231243032567"in Fig. 3(c)). This sequence encodes a representation for the text in the original document. Since each cluster, for the most part, corresponds to a single character, we can treat the sequence of cluster identifiers as a substitution cipher.

A substitution cipher replaces one character with another to produce an enciphered message. The original plain text can be recovered by a deciphering algorithm that computes the pairwise correspondence between symbols in the enciphered message and plain text characters. For the example shown in Figure 3, this correspondence is f 0; P ; 1; A ; 2; L ; 3; O ; 4; T ; 5; I ; 6; C , 7; Y g.We apply a deciphering algorithm to the sequence of cluster identifiers. It computes a pairwise correspondence between cluster identifiers and letters. This correspondence is used to recover the text in the original document image. This is essentially OCR'ing the document without actually applying any OCR techniques. A similar idea was first proposed in (Casey and Nagy, 1968). Our method takes advantage of the image preprocessing done by the symbolic compression technique. Also, we developed a new algorithm for substitution cipher decoding that takes into account characteristics of symbolic clustering in document images. Other techniques for substitution cipher decoding are described elsewhere.

Rabiner, L.R., Juang, et al,[11] introduced the deciphering algorithm reads the sequence of cluster identifiers from a symbolically com pressed document image and uses character transition probabilities and a hidden Markov model (HMM) to estimate the text that appeared in the original document. There might not be a decision for every character and all the decisions might not be correct. However, enough of the text is usually correctly recovered that accurate duplicate detection can be performed.

The text strings extracted from two documents are compared using th conditional n-grams they have in common[15]. A conditional n-gram is a sequence of n characters where each character satisfies a predicate. This predicate converts the original document into a new string from which n gram indexing terms are extracted. For example, we used a predicate that every character must follow a space. Therefore, the new string generated by this predicate contains the first character of every word. Conditional trigrams are formed from the first characters of three consecutive words[9].

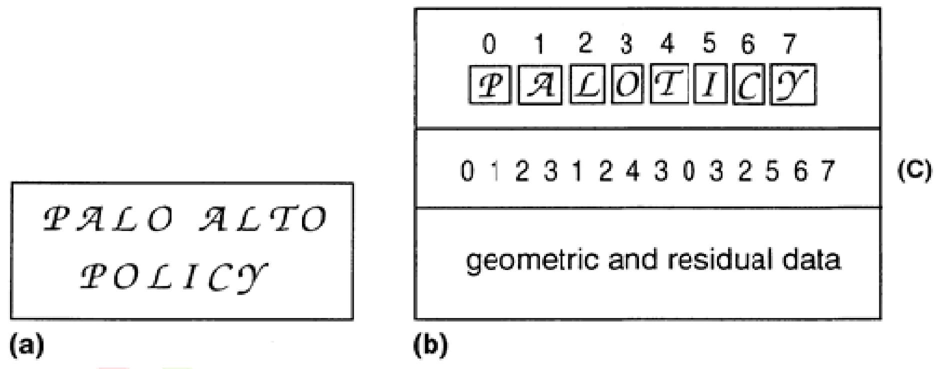

Figure 3: Depiction of symbolic compression, showing an original image (a) and the compressed image (b). The sequence of identifier's in (c) encodes the order of characters in the original document.

3.1 Image Compression

Image compression the initial breakthroughs in the compression of one-dimensional signals were easily extended to the image domain by concatenating image rows or columns into a single stream. Techniques such as Shannon-Fano coding and human coding use redundancy-reduction mechanisms which result in shorter codes for more frequently appearing samples. It is necessary to scan the data samples in order to determine their probabilities of occurrence and create an appropriate code.

This allows local changes in probability measurements and achieves higher global compression. Run-length coding is another redundancy-reduction coding method where in a scan-line each run of symbols is coded as a pair that specifies the symbol and the length of the run. While most redundancy-reduction methods are lossless, other arbitrarily lossy coding methods have achieved higher levels of compression. Transform coding, sub band coding, vector quantization, and predictive coding are among the ones that have achieved high levels of lossy compression. All transform coding based compression algorithms are pixel-based approaches which decompose a signal unto an orthonormal basis to achieve energy compaction. Lossy compression is then achieved by coding the high energy components and leaving out the low energy.

This method of coding needs to limit prediction error for efficient coding and achieves lossless compression in coding the entire error; lossy coding is achieved by limiting the range of the error value. More recently, second-generation compression algorithms based on human visual behavior have been proposed which have the potential for much higher compression ratios.

Also, developments in user-specified wavelet transforms can be used to address compression requirements for specific image types. Fractal coding, which is also called chaos coding, is an example of progress in the area of two-dimensional redundancy reduction. All of these compression techniques are pixel-based and they perform best for textured images.

3.2 Document characteristics

Document images are scans of documents which are in most cases pseudo-binary and rich in textual content. Informally, we can define document images as images that contain components that resemble the symbols of a language. Many documents look like those shown in Figure 4 and are represented adequately by binary images produced by scanning at a resolution of 300 dots per inch (dpi). They tend to be highly structured in terms of layout, and have significant redundancy in the symbols which occur in them. Since a document can be used in a large number of ways, an important consideration is how the image is affected by compression. For example, if we intend that the document ultimately be read by humans, it is necessary for compression schemes to preserve the shapes of the components so that they are recognizable by the reader after retrieval. If the document must be reproduced in near-original form, techniques which reduce resolution may not be acceptable. Thus the required resolution is dependent on the task; some resolutions may leave the document readable, but may destroy one detail. As is well known, the performance of conventional compression algorithms (such as those based on transform coding, vector quantization, fractal compression, pattern matching and substitution, and other approaches) depends on the types of images being compressed, and on their texture and content characteristics. Algorithms often do well on some classes of images and not so well on others. Since document images differ significantly from scene images, it is reasonable to assume they will benefit from a specialized compression scheme.

IV. DOCUMENT DECIPHERING

To implement proposed approach the HMM model are used. In an even more realistic scenario, a single pattern could correspond to a partial symbol or multiple symbols due to image fragmentation and segmentation errors. A deciphering algorithm is available for simple substitution ciphers. By exploiting the redundancy in a language, the plain text message can be recovered from a sequence of cipher symbols of sufficient length. Numerous algorithmic solutions have been proposed for simple substitution ciphers, including relaxation techniques dictionary-based pattern matching and optimization techniques. We propose a deciphering algorithm that uses a Hidden Markov Model (HMM) . Considering the Markov process of state traversal as a language source from which a particular plain text message can be generated with some probability, then the added symbol production at the traversed states in an HMM describes the enciphering process of a substitution cipher, where each letter in plain text is replaced with a cipher symbol one at a time.

This analogy between the source language modeling as a Markov process and the representation of the enciphering function by symbol probabilities is the basis for our solution. The state probabilities are initialized with language statistics, and the symbol probabilities are estimated with the EM algorithm. Information extraction from symbolically compressed documents can be viewed as a deciphering problem. The objective is the recovery of the association between character interpretations and pattern templates from a sequence of template identifiers. In symbolic compression schemes, image components are grouped to improve clustering and they are also roughly sorted in reading order to the reduce entropy in their relative offsets.

The objective of both measures is to improve compression performance. However, they also facilitate the application of deciphering techniques for information extraction. Figure shows an outline of the proposed HMM deciphering algorithm. It reads the pattern identifier sequence from a symbolically compressed document image and uses character transition probabilities to produce partial OCR results.

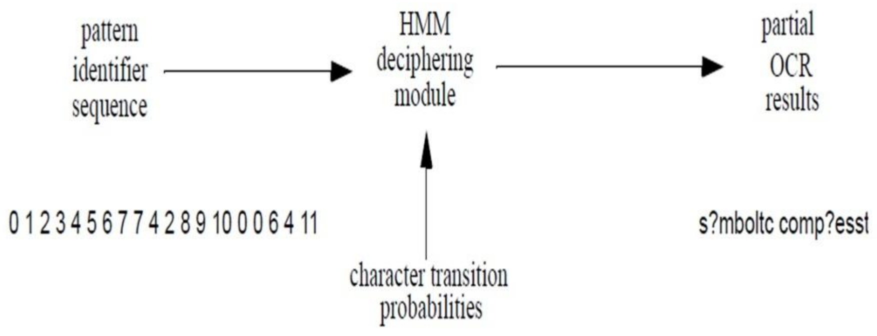

Figure 4: Deciphering a symbolic compressed image produces partial OCR results.

These results may not be completely correct. However, they are often adequate for various tasks which will be described in the next section First of all, it is obvious that the problem is never truly a simple substitution. Even with ample exemplars, certain contents such as numeric strings cannot be deciphered due to lack of context. Nevertheless, we believe sufficient information can be recovered for language identification, duplicate detection or document classification. Identification of the language of the text in the original image can also be performed by a version of the HMM deciphering algorithm.

4.1 Deciphering by Hidden Markov Models

Numerous solutions for the deciphering problem have been proposed, including relaxation algorithms [13][7], dictionary based pattern matching [12], and optimization techniques [3][16]. We propose a solution that uses the well-developed theory of Hidden Markov Models [15]. The abstraction of state transitions and observable symbols in an HMM is analogous to the separation of a Markov language source and subsequent enciphering step.

Markov models have been used for natural language modeling. If we accept the Markov process of state traversal as a language source from which a particular plain text message can be generated with some probability, then the added symbol production at the traversed states in a hidden Markov model perfectly describes the enciphering procedure of a monographic substitution cipher, where each letter in plain text is replaced with a cipher symbol one at a time. This analogy between source language modeling as a Markov process and representation of the enciphering function by symbol probabilities is the basis for our solution, as shown in Figure.

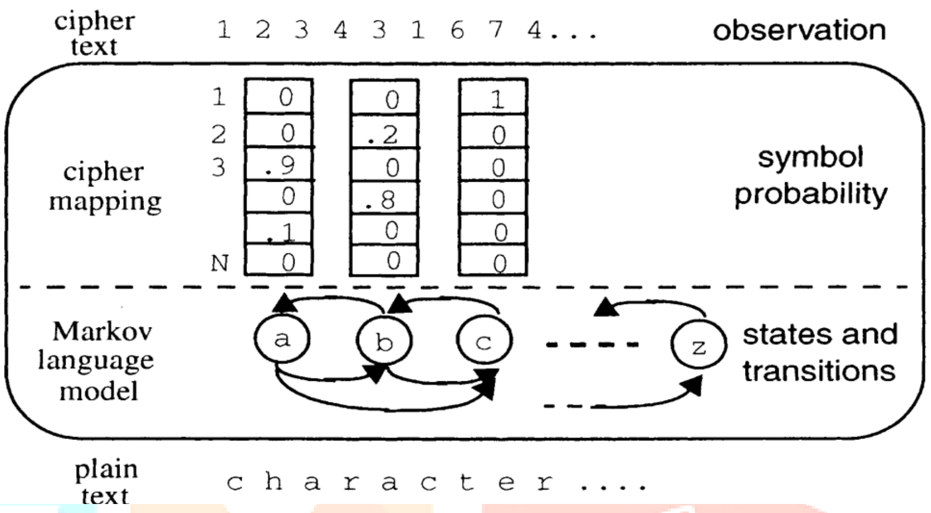

Figure 5: In the hidden Marko approach, the deciphering problem is formulated as ending the enciphering mapping that most likely produced the observed cipher text for the underlying Markov language source.

In a first order model, there are n states, each representing a letter in the plain text alphabet. Associated with each state, is a state transition probability function, and a symbol probability function. The first state in a sequence is selected according to an initial probability. Subsequent states are generated according to the transition probabilities, outputting one of the cipher symbols at each state with probability. Figure 5 - In the hidden Markov approach, the deciphering problem is formulated as finding the enciphering mapping that most likely produced the observed cipher text for the underlying Markov language source. The initial state probability is simply the character frequency of . Both the initial and transition probabilities are estimated from a corpus of the source language and remain fixed, providing a first order Markov modeling of the source language. Symbol probabilities are estimated using the forward-backward algorithm [15]. The initial estimation is defined as where is calculated from a cipher of length with occurrences of symbol using a binomial distribution. To determine the decipher mapping, we assign a plain symbol that most likely corresponds to each cipher symbol. Since is conditioned on the cipher symbol, the following decision criterion is used.

It should pointed out that we have explicitly constructed the deciphering function from the estimated symbol probabilities. However, it is unnecessary, even circuitous, to produce the underlying plain text this way. With fixed transition probabilities and estimated symbol probabilities, the Viterbi algorithm can be used to find the most likely sequence of states through which the observed symbols are produced. This state sequence, corresponding to the most likely plain text from which the cipher text is generated, given our parameter estimations, can be used directly for evaluation. Contrary to the deciphering solution where all occurrences of a cipher symbol in the cipher text must decode into the same letter in plain text, the most probable plain text generated from a state sequence may not have a consistent one-to-one mapping to the cipher text. Constructing the deciphering function incorporates the likelihood of all possible paths and has shown better results in our experiments than the direct method.

V. RESEARCH METHODOLOGY

In reference model symbolic compression is a method of coding similarly-shaped marks in an image with reference to a set of templates. In doing so, individual marks must rest be segmented out. For binary document images, segmentation is typically a matter of determining connected components of black pixels in the image. Perfect segmentation is not necessary since the residual bits which remain after replacing a component with a template are also coded. The segmented components are clustered to capture their redundancies.

Various methods of clustering, such as simple exclusive-or and compression-based template matching, can be used. Once all symbols within the image have been considered for clustering, the cluster centers can serve as representative templates or prototypes for the clusters. Some clusters may need to be removed because of inadequate membership, and some clusters may exhibit large in class variances which result in abnormally large residuals. The prototypes are stored in a library. The locations of the symbols must be recorded so that a highly compact representation, based on prototypes only, is immediately available.

This type of degradation does not adversely affect to the readability of the document and does not provide structural information. Conversely, it is observed that pixels farther from edges have dramatic effects on the readability of symbols and are structurally more important. By ordering the error pixels in distance-to-edge order, we can exploit their statistical characteristics by putting the most probable error pixels last, and we can provide structural coding, by putting pixels which have the potential to greatly impact the correct recognition of characters rest. This paves the way for compression by packing the energy in the error stream, for lossy compression by terminating the error stream at an index calculated by the signal-to-noise ratio, and for progressive transmission by interleaving the error signals for a set of image marks based on their distance order.based on prototypes only, is immediately available.

The difference between this symbolic representation and the original image is a residual image which has far fewer foreground pixels than before; this implies a high compression ratio. Context-predictive coding of the residual image further improves the compression ratio. Residual coding is the portion of the compression algorithm which encodes the residuals. To preserve symbol access for compressed-domain analysis, it is necessary to code the error on a per-symbol basis rather than coding it for the entire image. Using a structural model, it is possible to order the error pixels based on their structural importance. The structural order, derived in part from the work is based on the observation that the degradation of symbols in a document image from the effects of copying and scanning occurs mostly around the edges of symbols. Nagy et al. also observed the near-edge degradation of document images by considering the effects of point spread functions on the quality of a scanned image.

The fact that the components are repetitive in nature makes them a rich source for compression. Also the fact that document images contain a large amount of textual material laid out in a two-dimensional space makes them ideal for a two-dimensional redundancy reduction scheme. Since the useful information in the textual portion is contained in the components, access to the components provides access to that information. There is a strong case for the use of symbolic compression in document image coding, in addition to space savings.

With the use of only about one percent of the information in the original image, it is possible to code the dominant symbol shapes and their locations in the image. Since most of the information in a document is in the symbols, it is possible to perform compressed-domain analysis by using a symbolic compressed representation. Using only this representation, though it is not perfect, it is possible to perform a large number of document analysis tasks such as skew estimation/correction, Optical Character Recognition (OCR), layout analysis, and so on. Although some work has been done on the processing of compressed document images the outlook for applying such techniques to pixel-based methods does not appear promising.

To perform successful symbolic compression, we must perform a segmentation that decomposes the image into components that repeat. These components then need to be clustered so as to generate a set of representative prototypes. The quality of performance on these tasks, along with the quality of the image, can drastically affect the overall compression performance. In our approach to this problem we concentrate on bi-level document images that have large textual image content. This is not a strict requirement, but allows us to measure the performance of our document image compression scheme effectively. Work done by proves that block-based compression gives better performance on document images that contain images. It is also useful to consider algorithms such as, and others to provide image segmentation into regions of different types so that different compression schemes may operate on the regions. We assume that there exist enough components in the text regions that can be extracted by connected component analysis and that can be recognized correctly by a human reader.

VI. RESULTS AND DISCUSSION

From the above discussion it is observed that duplicate detection on compressed images is better performance then the uncompressed images so we described the different methods for performing document duplicate detection directly on images in a symbolic compression format. Since the language statistics inherent in document content are largely preserved in the sequence of cluster identifiers, the original character interpretations can be recovered with a deciphering algorithm. We proposed an HMM solution for the deciphering problem. While the overall character interpretation rates are not perfect, we demonstrated that sufficient information is recovered for document duplicate detection. This offers an efficient and versatile solution to detecting full and partial duplicates. It also provides a useful method for indexing large document databases. Future work will consider implementation of this technique in the IM3 system.

REFERENCES

[1] Ascher, R.N., Nagy, G., 1974. A means for achieving a high degree of compaction on scan-digitized printed text. IEEE Trans. Comput. C-23 (11), 1174±1179.
[2] Casey, R., Nagy, G., 1968. Autonomous reading machine.IEEE Trans. Comput. C-7.
[3] Howard, P., Kossentini, F., Martins, B., Forchhammer, S., Rucklidge, W.J., 1998.The emerging JBIG2 standard. IEEE Trans. Circuits Systems Video Technol. 8 (7), 838±848.
[4] Hull, J.J., Hart, P., 1998.The infinite memory multifunction machine. In: Pre-proceedings 3rd IAPR Workshop on Document Analysis Systems, Nagano, Japan, 4±6 Novem- ber, pp. 49±58.

[5] Hull, J.J., Lee, D.-S., Cullen, J., Hart, P., 1999.Document analysis techniques for the infinite memory multifunction machine. In: Proc. 10th Internat. Workshop on Database and Expert System Applications, Florence, Italy, 1±3 September, pp. 561±565.

[6] King, J., Bahler, D., 1992. An implementation of probabilistic relaxation in the cryptanalysis of simple substitution ciphers. Cryptologia 16 (3), 215±225.

[7] Lee, D.-S., Hull, J.J., 1999.Information extraction from symbolically compressed document images. In: Proc. 1999 Symposium on Document Image Understanding Technology, Annapolis, MD, 14±16 April, pp. 176±182.

[8] Lee, D.-S., Hull, J.J., 1999. Duplicate detection for symbolically compressed documents. In: Proc. 5th Internat. Conf.Document Analysis and Recognition, Bangalore, India, 20±22 September, pp. 305±308.

[9] Peleg, S., Rosenfeld, A., 1979. Breaking substitution ciphers using a relaxation algorithm. Commun. ACM 22 (11), 598± 605.

[10] Phillips, I.T., Chen, S., Haralick, R.M., 1993. CD-ROM document database standard. In: Proc. 2nd ICDAR, pp. 478±483.

[11] Rabiner, L.R., Juang, B.H., 1986. An introduction to hidden Markov models. IEEE ASSP Magazine, 4±16.

[12] Witten, I., Moffat, A., Bell, T., 1994. Managing Gigabytes: Compressing and Indexing Documents and Images. Van Nostrand Reinhold, New York.

[13] K. Mohiuddin, J. Rissanen and R. Arps, "Lossless binary image compression based on pattern matching," Proceedings of International Conference on Computers, Systems & Signal Processing, December, 1984.

[14] G. Nagy, S. Seth and K. Einspahr, "Decoding substitution ciphers by means of word matching with application to OCR," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. PAMI-9, no. 5, pp. 710-715, 1987.

[15] S. Peleg and A. Rosenfeld, "Breaking substitution ciphers using a relaxation algorithm," Communications of the ACM, vol.22, no.11, pp. 598-605, November 1979.

[16] I. T. Phillips, S. Chen, R. M. Haralick, "CD-ROM document database standard," Proceedings of the 2nd ICDAR, pp. 478-483, 1993.

[17] L. R. Rabiner and B. H. Juang, "An introduction to hidden markov models," IEEE ASSP Magazine, pp. 4-16, January 1986.

[18] R. Spillman, M. Janssen, B. Nelson and M. Kepner, "Use of a genetic algorithm in the cryptanalysis of simple substitution ciphers," Cryptologia, vol. 17, no. 1, pp. 31-44, 1993.

[19] I. Witten, T. Bell, H. Emberson, S. Inglis, and A. Moffat, "Textual Image Compression: two stage lossy/lossless encoding of textual images," Proceedings of the IEEE, vol. 82, no. 6, pp. 878-888, June 1994.

[20] I. Witten, A. Moffat and T. Bell, "Managing Gigabytes: Compressing and Indexing Documents and Images," Van Nostrand Reinhold, New York, 1994.