Towards Recognition of Degraded Words by Probabilistic Parsing

Karthika Mohan*
CVIT
IIIT, Hyderabad
AP, India 500 032
karthika.nair@gmail.com

K. J. Jinesh
CVIT
IIIT, Hyderabad
AP, India 500 032
jinesh@research.iiit.ac.in

C. V. Jawahar
CVIT
IIIT, Hyderabad
AP, India 500 032
jawahar@iiit.ac.in

ABSTRACT

Though, Indian language OCRs have shown significant improvement in classification rates in recent years, recognition of degraded words still pose a big challenge for the development of robust OCR systems. Ours is an attempt to formulate the problem of degraded word recognition in a generic and formal structure. We formulate the problem of degraded word recognition as a probabilistic parsing problem. A probabilistic parsing based framework is used to rank and validate various possible hypotheses. We effectively combine it with an alternate word generator, symbol recognizer and verification unit to improve recognition rates of degraded words without compromising good characters. We demonstrate our method on Malayalam. We experiment our method on a complete annotated book, where around 65% of the degraded words are correctly recognized using this approach.

1. INTRODUCTION AND RELATED WORK

Recent years have witnessed a rapid progress in the research and development of Indic Language OCRs [8]. High recognition accuracies have been reported for good quality images [15]. However, with degradation in image quality we observe a steep decline in overall accuracy. This is primarily due to the difficulty in correctly parsing connected components present in the degraded words. Our focus, in this paper, is on recognizing words with degradations such as *cuts*: separation/splitting of genuine components and *merges*: touching of genuine image components to form a single connected component. Throughout this paper, we assume that words are correctly segmented out. In other words, our focus is restricted to recognition of words in isolation. We demonstrate the performance of our methods on degraded word images in Malayalam, a prominent south Indian language.

There are myriad causes for image degradation. Cuts arise due to improper folding of paper, low quality of paper or presence of foreign materials. Large ink-blobs create merges. Heavy print can distort the symbols and render them unidentifiable [19]. Dust, fading away of ink and floating ink from facing pages accentuate the degradations. In addition to all these, the modern word processors, which are primarily designed for English introduce merges (and sometimes cuts) while formatting Indic scripts.

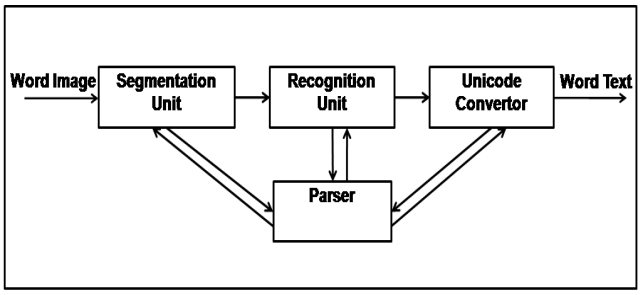

Figure 1: Architecture with parser embedded for enhanced performance

Basic recognition architecture for Indic languages comprises of three sequential steps: (i) Segmentation, where the word image is decomposed into a set of symbols, often connected components. (ii) Classification/Recognition, where the images of the symbols are converted to a set of class identifiers, and finally (iii) Unicode generation, where the sequence of symbols gets translated to Unicode or any other appropriate standard representation. Degraded words are often segmented incorrectly. Due to the linearity of the above mentioned architecture, the errors introduced by segmentation affect the accuracies of the ensuing modules and decrease the overall OCR accuracy. We propose a framework as described in Figure 1 wherein the parser is coupled to the segmentation unit, the recognition unit and Unicode generation unit. Our parsing scheme is motivated by the recent attempts in computer vision to analyse and understand complex and cluttered scenes [9, 18, 22].

In this paper, we model degraded word recognition as a probabilistic parsing problem and resolve it with the help of a parse graph. We proceed in two stages. In the first stage, the input image is pre-processed to yield a set of image fragments. In the second stage, the image fragments are parsed based on a grammar to create a set of feasible fusions. Ranked list of alternate words for the most probable word is then generated using prior knowledge available in the form of

*Corresponding author

ICVGIP '10, December 12-15, 2010, Chennai, India

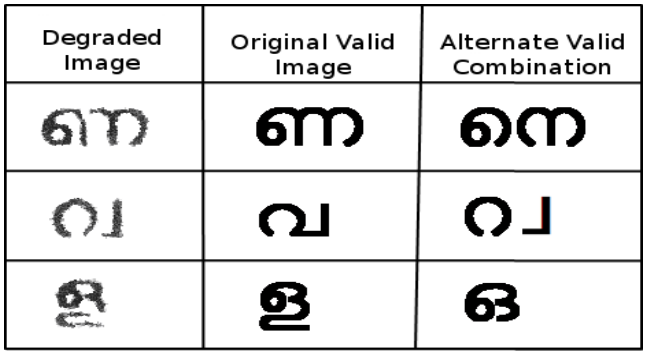

Figure 2: Valid symbols created by degradations, makes the problem further challenging

Statistical Sub-character Language Models(SSLM) [14] and confusion matrix. The correctness of words in the candidate set is confirmed by a verification unit.

Image parsing may be expressed in simple terms as soft labelling of pixels [17]. We partition the pixels in the word-image into components such that any foreground pixel in the image is contained in exactly one component. Image parsing may also be construed as an integrated framework for segmentation, detection and recognition [22]. The aim is to decompose the image into constituent components such that it retains a hierarchical structure that can be represented by a parse graph [9].

Parsing a two dimensional image of a constrained text block is detailed in [16]. Categories or high level descriptions are assigned to segmented regions. In the first step parser assigns a list of possible classes to each word. This is followed by horizontal and vertical refinement. The output is a ranked list of patterns based on the probability that patterns are correct. The use of structural information eliminated the need for perfect recognition and exhaustive dictionaries.

Image parsing requires a framework comprising of grammar, parse graphs, language etc. We now examine some of the grammar and allied parsing schemes that have been proposed to effectively parse images. Rekers and Schurr [18] have described the use of layered graph grammars for defining and parsing visual languages. Two types of graphs namely spatial relationship graph and abstract graph are put in use. Spatial relationship graph is a directed graph that depicts spatial relationships like above, contains etc. Abstract graph elucidates the syntax and semantics of visual sentence. Syntax of visual languages are defined by layered context sensitive graph grammars that are more general and less restricted compared to context free graph grammars. Shaw [20] had proposed an entire picture description language to aid picture parsing wherein the parsing algorithm is based on a top-down goal directed syntax analyzer.

In this work, our focus is on parsing degraded word images. Our objective is to correctly recognize degraded words. Earlier attempts in this direction formulated the problem as that of detecting a valid hypothesis, rather than formulating it as a probabilistic reasoning in a larger hypothesis space. One of the first solutions for segmentation of merged characters was classification based segmentation as put forth by Casey and Nagy [6]. Though the basic approach remained the same, further enhancements were proposed in [10, 13]. Degraded words were identified from reject set of OCR or by a module that checked for obvious cuts and merges. Different techniques were employed to cut the merges and thus decompose the symbol. The components so obtained were then re-classified. Segmentation of merged characters using a hybrid method integrating conventional as well as neural network based technique was proposed in [24]. A penalty metric based least cost path was used to determine the cut in degraded words that were detected using feed forward neural network. Penalty metric is chosen in such a way that only reasonable cuts are produced. Constraints are imposed to reduce time complexity and ensure efficiency.

Recognition of degraded words is a severe problem in Indian language documents. Garain and Chaudhuri [7] proposed a technique based on fuzzy multi factorial analysis for degraded document recognition. Break locations were identified using a predictive algorithm. Five fuzzy factors were designed based on their effectiveness and speed. Break locations are confirmed by using predictive parser concept since it significantly reduces the size of the candidate break location set.

Bansal and Sinha [3] presented a two pass algorithm for segmentation and recognition of merged Devanagari characters. In the first pass, statistical information is used to check if the segments, formed by decomposing word into easily separable parts, contain composite characters. In the second pass the composite character identified in the first pass is segmented into constituent symbols by extensively using structural properties.

An overview of character segmentation has been discussed in [12]. Cuts or broken characters may be handled either by using a merging process based on width and interval or by adopting recognition based segmentation approaches. In the merging process regions with minimum gap are merged and the statistics is recomputed. The process is repeated until the statistics fall within an acceptable range. Recognition based segmentation approach includes sliding window method, closed loop segmentation etc. Composite characters are decomposed by recognition based or feature based techniques. Feature based techniques make use of characteristics like width, height, aspect ratio, candidate segment, contour analysis etc. Segmenting based on objective functions were formulated and its maxima served as cut within the composite character.

As can be seen, most of the work on recognition of degraded words are based on a recognizer driven segmentation. In the case of Indian scripts, the problem is more compounded. With the complex layouts of symbols within the words (consisting of 1.5D placing of consonants and vowel modifiers), simple rules for detecting possible cuts and merges become impractical. Many of the modern word processors make the problem further complex by providing inconsistent spacing between symbols. It is very common to have the space between two cut-components higher than that of the distance between two consecutive symbols. Also the number of components and similar shapes are high in most Indian scripts. Therefore, often a cut or merge results in a connected component which is similar to another valid symbol as shown in Figure 2. All these point to the need for a formal framework where the optimal recognized word is obtained with the help of a probabilistic reasoning in the image space.

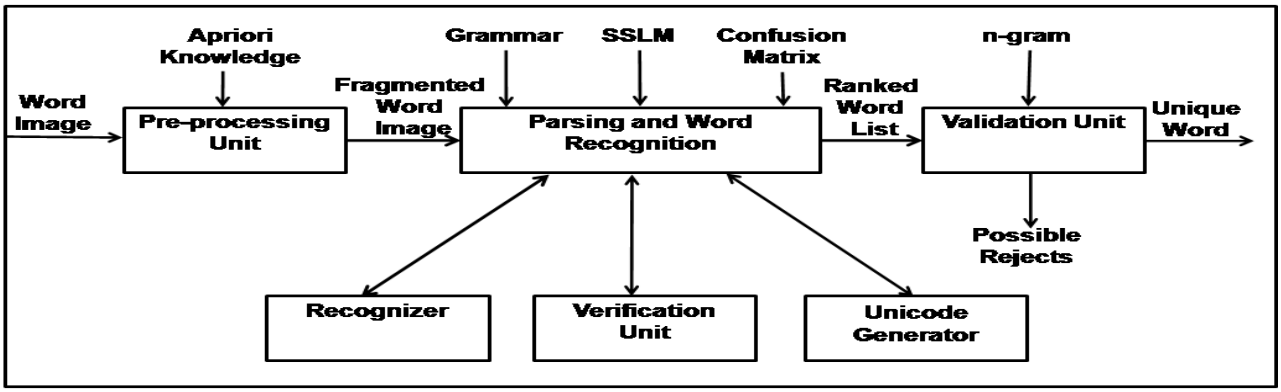

Figure 3: Overall Recognition Pipeline

2. OVERVIEW OF PARSING AND RECOGNITION

As explained in Section 1 (and Figure 1), our recognition process is driven with the help of a parsing module which does dialogue with segmentation, recognition and Unicode generation modules. A more detailed recognition pipeline is shown in Figure 3. Major difference is that we avoid talking to the segmentation module multiple times. The degraded (as well as the normal) words are first preprocessed and segmented. We assume that the word image is subjected to thresholding, skew-correction etc. prior to this stage. At this stage, components, including those which are fused are segmented out. Often this module results in an over segmentation of the word image. The fragments so obtained are sent to the parser module, that is tightly coupled with the recognition unit, visual verification unit and a Unicode generation unit. Parsing image fragments by using grammar, results in the generation of sequence of class indices. This information is used to generate alternate words and correct substitution errors, if any. The correction is aided by confusion matrix and statistical sub-character language models [14]. The ranked set of candidate words is processed by validation unit to generate the unique text output. In this paper, our interest is primarily limited to the parser module. Rather than outputting a list of words, we output the most probable word. This is done by applying backtracking algorithm which ensures that all available options are explored systematically.

Parsing.

In this section, we dwell on fundamentals of parsing, parse tree and grammar [1]. Parsing, also known as syntactic analysis or hierarchical analysis, is the process of determining if a set of components (tokens) can be generated by a grammar. Grammar, in general terms, is a set of rules that guides syntactic analysis. Parse trees are used to conveniently represent grammatical phrases and pictorially depict how tokens can be derived from the start symbol of the grammar. A language is the set of all tokens that can be derived from the start symbol of the grammar. Grammar consists of:

- Non-terminals: Syntactic variables or syntactic categories that impose a hierarchical structure on the language defined by the grammar.
- Terminals: They are the set of tokens or basic strings contained in the language.
- Start symbol: It is a non-terminal or start variable such that set of strings/tokens denoted by it is the language defined by the grammar.
- Productions: Set of rules that specify how non-terminals and terminals can be combined to form strings.

We are interested in generating words that are part of human language vocabulary. In other words the language generated by graph grammar is finite. Hence in our case, the parse graph grammar should be non-recursive.

We start by segmenting the word image into multiple components during the segmentation and pre-processing stage. Note that the word preprocessing is used in the context of pre-processing for a parser. It should not be confused with the traditional preprocessing in document image processing (which includes filtering, enhancement etc.) The aim of preprocessing module is to isolate different components as well as to detect break locations such that two symbols that are fused together can be broken or isolated. We look at feature based segmentation techniques. Features like width, average height, aspect ratio may be employed for break point detection. A careful examination of horizontal and vertical profile of the word image, by looking at the number of peaks, number of valleys and smoothness of peaks and valleys, throws light on possible break locations and aids in the generation of candidate break location set. Some of popular methods for this purpose in literature [4, 5, 11] are:

- T_1: Identify local minima and local maxima in symbol and make a cut passing through them
- T_2: Use horizontal profile to identify top strip, core strip and bottom strip. The word image is cut along two lines that separate the three strips.
- T_3: Using *a priori* knowledge about the script to identify components like vowel modifiers that create merged components. Once the presence of these components are confirmed, identify break locations using horizontal profiling technique.
- T_4: Use vertical profile to identify columns with minimal black pixels. Based on *a priori* knowledge, check if the location of black pixels can serve as candidate break location.

Our parsing module also looks at the statistical sub-character language models (SSLM) as described in [14]. The SSLM model describes the joint probability of pairs of adjacent symbols (sub-characters) appearing in a language. These are different from the typical uni/bi-grams computed at UNICODE or Akshara level for various language processing tasks. SSLM captures the distribution at symbol level. The utility of the SSLM in recognition is demonstrated in [14] during post processing of the classifier outputs. Generation of alternate words, their ranking and selection of optimal word are posed as an optimization problem in [14]. Here, we extend this framework by using the SSLM during the parsing phase.

The parsing and word generation modules uses *apriori* information in the form of (i) SSLM, (ii) Grammar and (iii) Confusion matrix. They are script/language specific. While generation of words, it talks with recognizer (or classifier), visual verification unit and Unicode generation module. This module generates a candidate set of words for the input word image. Accuracy is ensured by the use of statistical sub character language models. The set of classes that could be confused for a given class in the input word is obtained from the confusion matrix. Thus confusion matrix helps in generation of alternatives in a probabilistic setting. The task is modelled as a shortest path finding problem in a multi stage graph [14]. A path from the source node to the destination node in the multistage graph denotes the recognized text corresponding to the input word and the cost of the path influences the rank of the recognized text in the candidate set.

Verification module facilitates the selection of correct word from the ranked list of probable words returned by alternate word generation module. We discuss three validation/ verification techniques: visual verification, verification based on position and size and use of n-gram (or dictionary of partial words). During the parsing step, we use visual verification. We have a template based visual verification unit. Initially templates are generated for all classes in the language by superimposing example images. We store all templates in a fixed resolution (usually 40x40). The image of the component to be verified is converted to the same resolution as that of the template and then pixel wise AND operation is performed to determine the match score by aggregating the results from all the pixels. The selection of a word from the candidate set is based on a final score computed from the match score and the rank of the word in the candidate set.

Another verification possibility is based on position and size. We use this for punctuations. Suzuki *et al.* [21] have proposed the use of this technique for rectifying the mistakes in incorrectly recognized variables and numbers in mathematical expressions. Indic scripts like Devanagari always have a header line or *shirorekha*. There is a limited set of symbols that can occur in the top strip, above the *shirorekha*. Positional information can serve a strong criteria to reject classes that do not belong to the permissible list. Punctuation marks are a common source of error in document recognition. Positional information can correct some of the errors. For example, using positional information, it is possible to distinguish a comma from a closing single quote.

Dictionary of partial words (N-grams): Agglutination results in a large vocabulary size for most Indian languages. For example, 16 vowels and 37 consonants in Malayalam result in more than 200 symbols or graphemes. Hence the use of dictionary is not always recommended. N-gram (or dictionary of sub-words) based verification is another alternative. They are ideal for validation of the word or sub-word. They may be used for identifying valid words. However, it is not part of the parsing stage we discuss in this paper. Our results are reported without this stage.

3. PREPROCESSING AND PARSING

The parse graph comprises of nodes and directed edges. Nodes hold information about non-terminals and terminals. In this section, we discuss various techniques for generating nodes in the parse graph such that the word image is properly segmented and every symbol is correctly recognized. The type of degradation as well as the number of degradations in the input word image determine the number of nodes in the graph and the connectivity between them. Isolation of fused components, results in splitting of a symbol into two or more distinct symbols. In other words, whenever there are merges the number of nodes in a graph tends to increase. Cuts on the other hand, will result in two or more components combining to form a single component. In the case of cuts, there would be a reduction in the number of nodes.

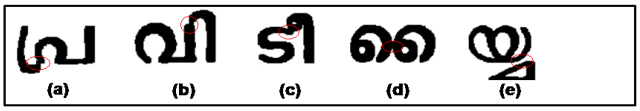

Figure 4: Frequently occurring degradations

Merges are handled as part of pre-processing. We analyze the data corpus available and identify the most frequently occurring forms of degradation patterns. Some of the frequently occurring degradation patterns are shown in Figure 4. In the following paragraphs we discuss various ways of handling degradations and show how observations from careful study of the corpus and script can be put to good use.

In most Indic scripts, vowels are represented either by using a diacritic or by applying changes to the form of the consonant or conjunct. The shape of the individual characters in many Indic scripts are such that we can visually identify several valleys and peaks. Many a time merges occur when a vowel represented by a diacritic gets fused with the consonant or conjunct. The peaks and valleys in the symbols can be identified based on the principle of local maxima and local minima. These points can serve as candidate break locations as shown in Figure 5(a). However, we observe that this technique creates several candidate break locations resulting in more number of nodes in the parse graph than expected optimally.

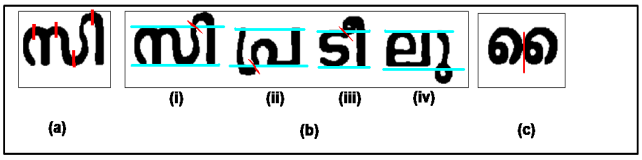

Figure 5: Identification of break locations on images

We now make an attempt to reduce the number of candi-

date break locations and correctly identify the desired break location. Bansal and Sinha [2] had proposed a segmentation technique for Devanagari script wherein word image is divided into top, core and bottom strips. Top strip is separated from core strip by a header line. Various techniques may be applied to segregate the three strips. The height corresponding to the top most and bottom most black pixel of each connected component is noted and their average value is computed. This value will help us in identifying the lines that separate the three strips. Another technique is to compute the horizontal profile and identify these lines by using a threshold value that may be computed as a function of image width. The top and bottom profile of input word image also comes handy while segmenting the image into three strips. For our experiments, we have used a combination of the first and last technique. Figure 5(b)(i),(ii),(iii) shows the unique break points identified using this technique. The vowel modifiers in Figure 4(b) and (c) form a confusing pair. A diagonal cut as shown in Figure 5(b)(i),(ii),(iii) ensures that features that are critical for distinguishing the confusing pair are preserved. The flaw in this technique is that it will segment symbols without merges also, as shown in Figure 5(b)(iv).

Now we present an *apriori* information based parsing technique. From Figure 4(a), (b) and (c) we can observe that merges are created by three different symbols that can be identified not only by their vertical or erect structure in the core strip but also by the fact that they extent considerably beyond the core strip into the top strip or bottom strip. These modifiers appear either on the right most side of the connected component or on the left most side of the connected component. We check for the presence of these components on each connected component and locate break locations only if their presence is confirmed. Hence images without merges are not unnecessarily segmented.

Figure 4(d) represents yet another merge pattern that has to be addressed. Based on an inspection of commonly occurring merge patterns, we know that typical characteristic of these merges are that they occur in the core strip. To identify and rectify such merges, we compute the vertical profile of the connected component and identify the column with minimum number of black pixels. If the pixels are concentrated in the core strip, then all black pixels along the column are converted to white, thus isolating the fused components (Figure 5(c)). Unlike, the previous technique, this has to be performed for all connected components, that do not contain merge patterns in Figure 4(a), (b) and (c).

Unlike merges, it is not possible to identify a set of frequently occurring cut patterns. However, Figure 4(e) depicts a cut pattern that can be resolved by keenly observing Malayalam script. In this cut pattern, there are connected components that are located directly under the connected component that was parsed just before. To obtain connected components we examine each column (from top to bottom) as we move from left to right.Hence a component on top has to be parsed before a component that lies right under it can be parsed. Using our knowledge of Malayalam script we can say that such cut patterns can be merged to form a single component.

After a series of pre-processing operations, we have image fragments that are unlikely to be a fused component.In the next stage of degraded word recognition, we strive to create feasible combinations of image patterns. There are various

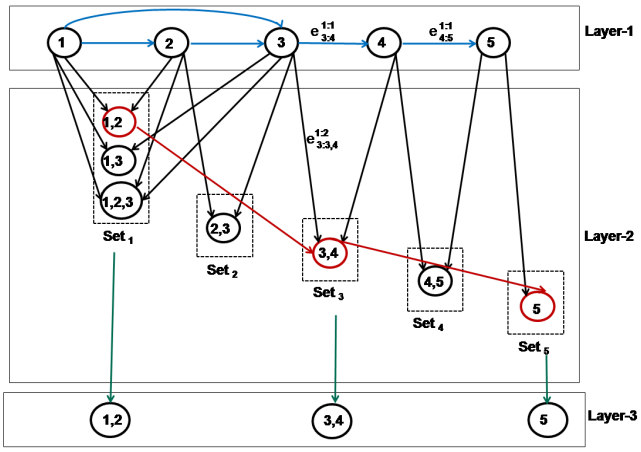

Figure 6: Sample parse graph

parameters like classifier confidence, grammar etc. which allow us to do this. Parsing proceeds in such a way that minimal combinations or merged fragments are created. At every stage, we prune the graph so that it has an optimal structure with minimal nodes and edges. The next section discusses parsing in detail.

3.1 Probabilistic Image Parsing

Image parsing is effected by building a directed parse graph as shown in Figure 6. We follow top-down parsing strategy. The nodes in the graph are encapsulated in layers. The last layer of the graph is called the leaf layer. N_i^j denotes a node in layer j labeled i. Nodes in non-leaf layers represent image fragments and contain the classes associated with the image fragments that they symbolize. The weight of the node is equal to the confidence value returned by the classifier for the image fragment that it represents. An edge in the graph from N_i^j to N_k^l is denoted by $e_{i:k}^{j:l}$. The weights of the edges in layer 1 and the weights of edges from layer 1 to layer 2 of the graph denote the probability that the image fragments contained in the corresponding nodes can be successfully merged to form a single component. Weights of edges from layer 2 to layer 3 indicate the probability that the class label contained in the layer 2 node can be successfully converted to Unicode. The edges between nodes in layer 2 indicate the SSLM values corresponding to classes associated with the nodes.

The graph grammar G, is a 4-tuple:

$$G = \{V, \Sigma, R, S\} \tag{1}$$

The start symbol S denotes the nodes encapsulated in layer 1 that represent fragmented word images obtained after preprocessing. The set of non-terminals V comprises of all nodes representing image fragments that are encapsulated in all non-leaf layers. Set of terminals Σ consists of nodes representing Unicode that are encapsulated in the leaf layer. V and Σ are disjoint sets *i.e.* $V \cap \Sigma = \phi$. We have two production rules in R:

- $r_1 : N_i^1 \rightarrow N_{i,k}^2$ where k denotes elements in the subset of the set that contains labels of all terminal nodes whose initial node is i. For example, $N_1^1 \rightarrow N_{1,2}^2$; $N_1^1 \rightarrow N_{1,3}^2$; $N_1^1 \rightarrow N_{1,2,3}^2$.

- $r_2 : N_i^2 \rightarrow N_i^3$ where $N_i^3 = unicode(N_i^2)$. It may be noted that $unicode(N_i^2)$ represents the Unicode corresponding to class label contained in node N_i^2

We now explain the operations performed to generate the parse graph. Initially we have image fragments that have been pre-processed. The nodes in layer 1 represent these image fragments and contain the class label assigned by the classifier. The confidence value returned by classifier is the node weight. The edge weights may be computed as the ratio of difference between the width of initial word image and horizontal distance between the furthermost pixels in the image fragments contained by the nodes, to the width of initial word image. Lesser the edge weight, higher the probability of merging the corresponding image fragments. If the edge weights exceed a threshold, the edges are removed. The criteria for deciding threshold could be as simple as the maximum width of a symbol in the language. Threshold value is crucial because number of edges in layer 1 influences the number of nodes in layer 2 and hence the complexity of parsing.

Layer 2 contains all candidate merges. The nodes in layer 2 are organized into sets. Set_i comprises of nodes that symbolize merged image fragments represented not only by N_i^1 but also by all nodes in layer 1 that have an edge directed towards them from N_i^1. For example, nodes in Set_1 always contains the image fragment in N_i^1 and may also contain image fragments in N_i^2 and N_i^3 because $e_{1:2}^{1:1}$ and $e_{1:3}^{1:1}$ exist. This is the essence of production r_1. The number of sets in layer 2 never exceeds the number of nodes in layer 1. The weight of an edge from layer 1 to layer 2 is the ratio of difference between weight of node in layer 2 and weight of node in layer 1, to weight of node in layer 1. Higher this value, higher the probability of the concerned image fragments being merged.

After creating all sets, if we observe that there are no edges with positive weights emanating from N_i^1 to any node in layer 2, then we add that node to set_i and rename it as N_i^2. This explains the presence of a node labelled 5 in layer 2. It indicates that the likelihood of the image fragment within the respective node to be a complete component without any degradation, is very high. Pruning of graph for higher efficiency may be effected by removing all nodes in layer 2 that have a negative edge entering it from layer 1.

We have to now identify the most probable word. There are techniques such as dynamic programming that would allow us to identify the optimal path corresponding to most probable word. However, we require all image fragments to be present in the optimal path, just once. We adopt backtracking technique to satisfy this constraint. The candidates are incrementally built. If the partial candidate does not satisfy the constraint, then we reject it and backtrack. This method is more efficient than brute force enumeration of all candidate words. Rank of word is computed as the product of node weights(confidence value) and edge weights of all nodes and edges along the path. We apply r_2 and create nodes in layer 3. Nodes in layer 3 share the same labels as that of the selected nodes in layer 2. The Unicode representation of the word image may be obtained by reading the nodes from left to right. This is under the assumption that initial parsing of word image also proceeded from left to right. The use of SSLMs considerably reduces the chances of invalid Unicode sequences. However, in case the Unicode is found to be invalid, the parser can either denote it using special symbol or backtrack and compute the next most probable word from layer 2.

4. EXPERIMENTS AND RESULTS

To evaluate the effectiveness of our parsing framework described in Section 3 we designed two different sets of experiments. First one is to evaluate the effectiveness of the algorithm on severe degradations. While another set of experiments were identified to evaluate the performance on a large real world corpus.

For the first experiment, we selected a word which is perfectly recognized by the classifier without the help of the parsing framework.Then added many levels of degradations which breaks the components severely. Figure 7 shows some samples of the dataset. It was made sure that initial word error rate for the set is 100%. That is no single word was completely recognized by the existing recognizing system.

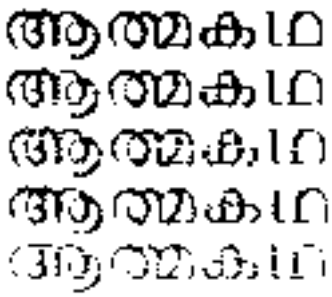

Figure 7: Sample of Degradations

For investigating the effectiveness of the method in real world conditions, we picked the words which failed recognition from an already annotated book. The corpus is carefully scrutinized to remove all those errors coming from misclassification. Also, the word annotations available in Unicode were verified. This gave us a corpus of degraded words with 100% word error rate (i.e., all words were incorrectly recognized). Some samples of degradations in dataset is shown in Figure 9 and Figure 10. Figure 8 shows some sample words in the dataset.

4.1 Improving Severe Degradations

Experiments were conducted on the severely degraded word image dataset. Since we were into testing the effectiveness of the parsing with this experiment, all degradations added were cuts. A dataset of 741 degradations were used for the experiment. Table 1 shows the over all results on the dataset.

Symbols	Accuracies			
	Symbol Level		Word Level	
	Initial	Final	Initial	Final
3705	8.8	78.14	0	60.30

Table 1: Results on Severely Degraded Words

Initial accuracy is the Unicode character level accuracy for the recognition unit without the aid of parsing framework. Final Accuracy is the accuracy of the system when the parsing module is used to recognize degradations. Almost 70% improvement in performance is obtained at Unicode level. Accuracy at Unicode string level is calculated as edit dis-

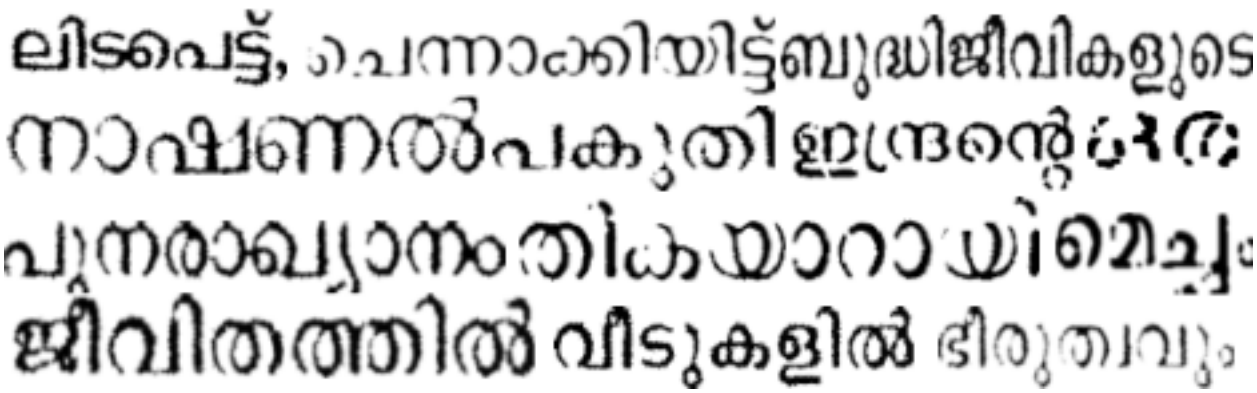

Figure 8: Sample of degraded images present in data set

tance(Levenshtein distance) between annotation and OCR output.

Table 1 also shows that 60.3% of words were corrected. It means, of the dataset 60.3% of words were completely recognized without any error.

4.2 Performance on Natural Book Dataset

Experiments and results in Section 4.1 shows that the parsing framework is working very well with degraded images. But, degradation levels in normal pages are not comparable to the dataset used for the experiment in Section 4.1. As shown in Table 2 and Table 3, initial accuracy in real world datasets are usually very high. But word error rate will be still high. Since the number of degradations per word will be less, even a slight improvement of around 20% will give an improvement of 70% at word level.

Experiments on natural dataset were conducted as two independent experiments. For that, the degraded words were divided into ones with cuts (word with both cut and merge is also added to this category) and another with only merges. The results on the first set, shown in Table 2 represents the performance on real world merge images.

Symbols	Accuracies			
	Symbol Level		Word Level	
	Initial	Final	Initial	Final
4040	71.31	93.63	0	70.45

Table 2: Results on images with cuts

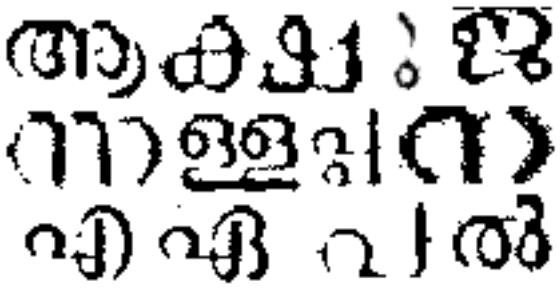

Figure 9: Sample of cut symbols in dataset

The results on the second set actually represents the improvement after some artifacts are introduced by merge processing or break point detection module in Section 3. As evident from results shown in Table 3,65% improvement in word level is still obtained.

Symbols	Accuracies			
	Symbol Level		Word Level	
	Initial	Final	Initial	Final
3615	75.66	92.5	0	65.55

Table 3: Results on images with merges

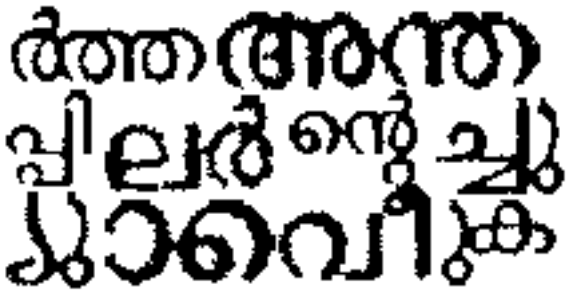

Figure 10: Sample of merged symbols in dataset

5. DISCUSSION

We have put forth a probabilistic image parsing technique that would aid recognition of degraded images. We explicitly mention the set of operations that are to be performed on the input word image to obtain the Unicode output. The parse graph depicts the sequence in which these operations are to be performed on the pre-processed image fragments so that we can obtain the text corresponding to the input image. Grammar governs the generation of non-terminals and terminals such that for every degraded word, a parse graph adhering to the grammar rules is created. Once the parse graph is in place, the most probable word at Unicode level is outputted by incrementally building solution sets using backtracking technique.

The effectiveness of syntactic pattern recognition for hard core vision problems like scene analysis has once again brought image parsing techniques to the fore. Hence the formulation of degraded word recognition as a parsing problem is

deemed appropriate. There is also a widespread use of Probabilistic Finite State Machines like Probabilistic Finite State Automata, HMM's etc. in fields that are closely associated with pattern recognition. However the use of PFSA for our problem is hindered by the fact that though PFSA facilitates the finding of optimal path for a given string, it cannot find the most probable string for a given problem [23].

We have come up with a formal structure to correctly recognize degraded images. The architecture proposed is generic and may be used for other languages to resolve the confounding problem of degraded word images in a straightforward and uncomplicated manner. However, further explorations are required to convincingly conclude the applicability for other languages like Hindi and Telugu. The multistage graph representation proposed in [14] and the parse graph employed in this paper are structurally different. But both of them help in two different phases of the recognition pipeline – in segmentation/parsing and in post-processing. One may look for an integrated framework where both these are done together.

6. CONCLUSIONS

In this paper, we proposed a formal framework for parsing and recognition of degraded words. The results in Section 4 clearly shows that the formulating degraded word recognition problem as a probabilistic parsing problem is a step in the right direction. Parsing and ranking of degradations is effective in case of severe degradations. The proposed parser framework effectively handles the false break point detection problems. This point to a possibility of using the probabilistic parser as a formal and generic framework to recognize degraded words.

7. REFERENCES

[1] A. V. Aho, R. Sethi, and J. D. Ullman. *Compilers Principles, techniques and tools*. Pearson Education, 2005.

[2] V. Bansal and R. Sinha. A complete ocr for printed hindi text in devanagiri script. In *Proc of 6th ICDAR*, 2001.

[3] V. Bansal and R. Sinha. Segmentation of touching and fused devanagari characters. *Pattern Recognition*, 2001.

[4] T. Bayer, U. Kressel, and M. Hammelsbeck. Segmenting merged characters. *ICPR*, 1992.

[5] T. A. Bayer and U. H. G. Krebel. Cut classification for segmentation. *International Conference on Document Recognition and Pattern Recognition*, 1993.

[6] R. G. Casey and G. Nagy. Recursive segmentation and classification of composite character pattern. *ICPR*, 1982.

[7] U. Garain and B. Chaudhuri. On ocr of degraded documents using fuzzy multifactorial analysis. *Advances in Soft Computing*, 2002.

[8] V. Govindaraju and S. Setlur, editors. *Guide to OCR for Indic Scripts*. Springer, 2009.

[9] F. Han and S.-C. Zhu. Bottom-up/top-down image parsing by attribute graph grammar. *ICCV*, 2005.

[10] S. Harmalkar and R. Sinha. Integrating word level knowledge in text recognition. *ICPR*, 1990.

[11] J. D. Hobby and T. K. Ho. Enhancing degraded document images via bitmap clustering and averaging. *ICDAR*, 1997.

[12] Y. Liu. Machine printed character segmentation- an overview. *Pattern Recognition*, 28, 1995.

[13] M. Maier. Separating characters in scripted documents. *ICPR*, 1986.

[14] K. Mohan and C. V. Jawahar. A post-processing scheme for malayalam using statistical sub-character language models. *DAS*, 2010.

[15] N. V. Neeba, A. M. Namboodiri, C. V. jawahar, and P. J. Narayanan. Recognition of Malayalam Documents. *Guide to OCR for Indic Scripts*, 2009.

[16] M. Prussak and J. J. Hull. A multi-level pattern matching method for text image parsing. *IEEE-CS Conference in Applications of Arti cial Intelligence*, 1991.

[17] D. Ramanan. Learning to parse images of articulated bodies. *Advances in Neural Information Processing Systems*, 2007.

[18] J. Rekers and A. Schurr. Defining and parsing visual languages with layered graph grammars. *Journal of Visual Languages and Computing*, 1996.

[19] S. V. Rice, G. Nagy, and T. A. Nartker. Optical character recognition: An illustrated guide to the frontier. *Kluwer*, 1999.

[20] A. C. Shaw. Parsing of graph-representable pictures. *Journal of the ACM*, 1970.

[21] M. Suzuki, F. Tamari, R. Fukuda, S. Uchida, and T. Kanahori. Infty-an integrated ocr system for mathematical documents. *Proceedings of ACM Symposium on Document Engineering*, 2003.

[22] Z. Tu, X. Chen, A. L. Yuille, and S.-C. Zhu. Image parsing: Unifying segmentation, detection, and recognition. *ICCV*, 2003.

[23] E. Vidal, F. Thollard, C. de la Higuera, F. Casacuberta, and R. C. Carrasco. Probabilistic finite state machines- part 1. *IEEE Trans PAMI*, 2005.

[24] J. Wang and J. Jean. Segmentation of merged characters by neural networks and shortest-path. *Pattern Recognition*, 1994.