Character n-Gram Spotting in Document Images

Sudha Praveen M., Pramod Sankar K., C. V. Jawahar
Center for Visual Information Technology
IIIT-Hyderabad, INDIA
Email: {sudhapraveen.maremanda@research., pramod_sankar@research., jawahar@} iiit.ac.in

***Abstract*—In this paper, we present a novel approach to search and retrieve from document image collections, without explicit recognition. Existing recognition-free approaches such as word-spotting cannot scale to arbitrarily large vocabulary and document image collections. In this paper we put forth a framework that overcomes three issues of word-spotting: i) retrieving word images not labeled during indexing, ii) allow for query and retrieval of morphological variations of words and iii) scale the retrieval to large collections. We propose a *character n-gram spotting* framework, where word-images are considered as a bag of visual n-grams. The character n-grams are represented in a visual-feature space and indexed for quick retrieval. In the retrieval phase, the query word is expanded to its constituent n-grams, which are used to query the previously built index. A ranking mechanism is proposed that combines the retrieval results from the multiple lists corresponding to each n-gram. The approach is demonstrated on a size-able collection of English and Malayalam books. With a mean AP of 0.64, the performance of the retrieval system was found to be very promising.**

***Keywords*-Word-Spotting, Character n-Grams, Recognition-free, Scalability**

I. Introduction

There are two popular approaches to building search systems over document image collections: i) Optical Character Recognition (OCR) and ii) Word-Spotting. OCRs typically work well for *clean* documents, but face severe difficulties in character segmentation and recognition [1] in the presence of degradations. Further, robust OCRs are not yet available for many Indian languages [2], which have an extended character set, with a complex script layout. Word-spotting approaches [3], [4], [5] overcome some of the drawbacks of OCRs. Firstly, segmentation at the word level is much more accurate than character/component segmentation [6], since degradations typically do not affect the inter-word spaces. Further, matching at the word-level benefits from the additional context present in it, allowing it to disambiguate between similar looking characters by using the appearance of the entire word.

In a typical word-spotting framework, word-segments of document images are represented with holistic features. The word-images are then clustered, by matching their features using DTW-based distance measures. The clusters thus formed are used to index the document image collection, directly in the feature space. Such an indexing scheme is limited by the vocabulary that is indexed/clustered in an offline phase. If a new word is given as a query, the index either rejects it, or returns a visually-similar cluster that need not be semantically relevant. Further, word-spotting is not directly applicable to partial word matching of a query with prefixed/suffixed words in the collection. For example, let us assume that the document collection has an instance of *bicycle*, while the query is *cycling*. Ideally, the document containing *bicycle* is relevant to the given query, and should thus be retrieved. However, that would not be possible with a regular word-spotting setup. This could possibly be addressed using a DTW based partial matching [7], but such techniques are not scalable to large collections. Typical partial matching on even a few thousand document images could require many hours of computing time per query, thus making it infeasible for practical applications.

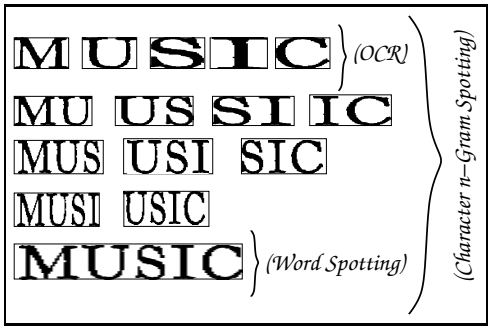

Figure 1. Traditional approaches index/recognize either characters or words. In contrast, our approach indexes the character n-grams, i.e. groups of consecutive characters in the word. With our framework, we fill a major gap in the problem space, while scaling *spotting* techniques to large vocabulary and document collections.

In this paper, we address the above mentioned shortcomings of character recognition and word-spotting, by proposing a novel *character n-gram spotting* framework. A character n-gram is a sequence of n consecutive characters in a given word. The various character n-grams for the word *MUSIC* are shown in Figure 1. While sufficient work has been seen in character recognition as well as word-spotting, little work explored the character *n-gram* spectrum between the two extremes. With our approach, we attempt to fill this gap in the problem space.

There are however certain challenges with this approach. Firstly, segmentation of a word into its constituent character n-grams is a non-trivial task, especially in the presence of

cuts and merges that are common in scanned document images. After segmentation, we obtain all possible character n-grams from each word. Thus, the number of features that need to be indexed is multiplied by a large factor. This would seriously effect the scalability of the indexing scheme to large collections of document images. Finally, during querying, a single query is replaced by an expanded query set consisting of its character n-grams. Combining multiple retrieved lists and their appropriate ranking needs to be carefully addressed.

The major contributions of our work are:

- A novel reposing of the word-spotting problem to one of character n-gram spotting. We believe that character n-gram is a more useful primitive for indexing and retrieval of document image collections.
- Making it possible to perform *spotting* for unconstrained vocabulary sets, without the need for exhaustive labeling of the vocabulary.
- Enabling sub-word retrieval in the image space, without the need for a morphological analyzer.

The approach is designed to scale to large datasets, by the effective use of efficient indexing schemes. The retrieval time of our search system, that indexes more than a million features, is typically less than one second. Our framework is easily applicable to other languages and scripts.

II. Character n-Gram Spotting

In word-spotting, an index is built by clustering word-images extracted from document images. Word-images are represented using holistic features such as those based on vertical profiles (see [8] for details). Clusters formed on these features are used to index the document collection, either directly or by automatically annotating the clusters [5]. The clusters typically contain multiple occurrences of the exact same word, since partial matching of words in feature is quite inaccurate as well as computationally expensive. The index is thus limited by the vocabulary seen during the indexing phase, making it difficult to retrieve for unseen (but similar) queries.

Instead of words, character n-grams are a better primitive to index document image. n-Grams have been a very powerful tool in traditional information retrieval. Previous approaches have used n-gram information to refine recognition explicitly with post-processors, or implicitly by modeling them in Hidden Markov Models (HMMs). However, little known work has applied n-grams for retrieval, especially in a recognition-free environment.

In our work, we treat each word-image as a bag of its constituent character-n-grams. A character n-gram is a segment of a word-image that contains n consecutive character segments. With the presence of multiple characters, character n-grams provide contextual information, that would aid in better character disambiguation. Since each word emits a large number of n-grams, with a relatively small training set, almost all possible n-gram occurrences could be identified and indexed. Even in the case of queries that give rise to rare un-indexed n-grams, other indexed n-grams from the query would typically suffice for retrieval.

The character n-grams could either be indexed in the feature space, or in the text domain by manual or automatic annotation. Thus, character n-gram spotting encompasses both OCR and word-spotting approaches, and augments them by evidence from matching n-grams. In this work, we shall limit ourselves to indexing in the feature space, leaving the task of obtaining textual labels for future work. Henceforth, when we mention n-grams, it means *character n-grams* represented in the visual feature space, and not the usually referred word n-gram probabilities.

A. Indexing Phase

In character n-gram spotting, all word-images are segmented into their constituent character n-grams. In the indexing phase, the word-images of *bicycle* would be segmented into various visual n-grams such as {b,i,...,e}, {bi,ic,...,le}, {bic,icy,...,cle}, and so on. Features extracted from the visual n-gram segments are clustered to build an index in the feature space (text-labels for clusters are not obtained). However, clustering in visual feature spaces is a computationally expensive task.

For example, K-Means clustering of feature vectors is of the order $O(N.K)$, N being the number of data points to cluster and K is the number of clusters. A large K is required to ensure that each cluster contains instances of only one given visual n-gram. However, this clearly increases the compute time, taking about 31 yrs to cluster a million visual n-grams with a K of 100,000. This issue is addressed using Hierarchical K-Means (HKM) [9]. HKM begins with a small number of clusters - say B, each of which is expanded to B clusters each and so on up to a certain depth D. The complexity is now of the order $O(N.B)$, where $B << K$, resulting in an indexing time of 13 hrs for HKM vs 31 years for K-Means. We typically use a B of 32 and a depth of $log(N)$, N being the number of data points being indexed.

Similarly KD-Trees were also be used to index the features. In KD-Trees [10], the feature space is partitioned by axis-parallel hyperplanes. The algorithm splits the data in half at each level of the tree on the dimension for which the data exhibits the greatest variance. The KD-Tree is looked up for approximate-NNs, by comparing the query with the bin-boundary at each level of the tree.

B. Retrieval Phase

In the retrieval phase, the user is allowed to query-by-example. The query-image is segmented to n-grams and the features from the query n-grams are looked up in the previously built index, for the most similar cluster in the index. For example, if the user queried for *cycle*, the query

is expanded to include all its character n-grams, including the query word itself.

The cluster of the query feature can be found in $B \cdot D$ comparisons using our indexing scheme; unlike traditional indexes (those from K-Means) which would take B^D comparisons. A linear search within this cluster, is used to rank the n-grams in the cluster based on similarity with the given query. The approximate NN lists for the respective n-grams are then combined and ranked.

The ranking function consists of two parts. If Q is the query word with length L, then let us denote as $Q_1, Q_2, ..., Q_L$, the sets of character n-grams for the query. Let F_i be the set of features corresponding to the n-gram set Q_i. For each F_i^j, the closest cluster of the indexed data, is given as R_i^j. Each point P_k in the cluster R_i^j is weighted by its distance from the centroid of the cluster as

$$W = \Big(1 - \frac{d(P_k, R_i^j)}{\sum_k d(P_k, R_i^j)}\Big).$$

This weight gives more importance to points that are closer to the cluster center than those at the fringe of the cluster. The denominator of the fraction normalizes across the size of the cluster in the feature space. Their values are independent of the query and can thus be computed in the offline phase and stored along with the index.

The second part of the ranking function ensures that longer n-grams are given more weight than shorter n-grams. We choose to reduce the combined weight of each n-gram by half for each step of the n-gram, the k-gram will be given twice the weight of $k-1$-gram and so on. The weight for the n-gram in a query word of length L, is given as

$$W'_n = \frac{1}{2^{L-n} \cdot (L-n+1)}.$$

The two weights are multiplied for each retrieved result, and the documents corresponding to the ranked list is presented to the user. In case of multi-word queries, a ranked retrieval list is generated separately for each word, which are combined using a modification of the term-frequency function.

C. Worked-Out Examples

In this section, we shall elaborate the expected retrieval procedure for some example queries.

Query: cycle, Occurrence: cycle: In the indexing phase, all instances of *cycle* are assumed to be clustered together in the feature space. During retrieval phase, when the query-image corresponding to "cycle" is given, it is looked-up in the index, and the closest cluster would be retrieved. This is essentially how the regular word-spotting framework works, which is encompassed by our framework.

Language	Books	Images	Words	n-Grams	Feature Size
English	4	470	82K	1.36M	12G
Malayalam	5	302	47K	1.28M	11G

Table I
DATASET STATISTICS.

Query: cycling, Occurrence:bicycle: During the indexing phase, the n-gram *cycl* occurring across all instances of *bicycle*, would be indexed. During retrieval, we segment the query word-image *cycling* into its constituent n-grams, each of which is queried for in the n-gram index. When queried by the n-gram *cycl*, all those occurrences of this n-gram from *bicycle* are retrieved.

Error Cases: In many cases, there might be many valid n-gram combinations within a given word. For example, the word *bicycle* also contains the word *icy*, which is not a valid morphological segmentation. Our framework would however still query for *icy*, and retrieve its multiple occurrences from the collection. Such situations are handled by the ranking mechanism that we propose, where larger n-grams are given more weight than smaller ones. It is more probable that a larger sub-word would be a more valid sub-word than a smaller one.

III. EXPERIMENTS

Our dataset comes from 4 English books and 5 *Malayalam* books. The statistics for the dataset are given in Table I. Groundtruth was created for the English dataset using semi-automatic annotation techniques [11]. About 70% of the data was annotated and the groundtruth was used only to evaluate the clustering and retrieval performance and was not used to index the data itself.

A. Features

The character n-grams are represented with profile features [3], which have been shown to perform well for the word recognition task [6]. Profile features have outperformed HOG [12] and SIFT [13] based representations, in an evaluation performed over 32K word-images [6]. The profile features that are extracted include:

- The Projection Profile is the number of ink pixels in each column.
- Upper and Lower Profile measures the number of background pixels between the word and the word-boundary
- Transition Profile is calculated as number of ink-background transitions per column.

Each profile is a vector whose size is the same as the width of the word. A fixed length representation is typically required to use simple distance measures, which can then be used for easy indexing. This is obtained by scaling all word-images to a canonical size and extracting profiles from them.

B. Indexing of Character n-Grams

Indexing of the features was performed using the L2-Norm, for all the character n-grams. We first attempted to build separate indexes for each n-gram. However, since the character segmentation was not accurate, the same character n-gram was present in multiple indexes. We abandoned this approach and instead build a single index across all n-gram segments.

In the n-gram segmentation phase, we inevitably encounter segmentation errors, such as two characters being merged together, or a single character being segmented to two. In the case of cuts, the character will be represented as a bi-gram and higher order gram will treat it as a regular pair of characters. When indexed, there will always be n-gram components that correspond to the given character, hence allowing it to be retrieved. For merged characters, the two characters will be considered as a unigram. Instead of tri-grams, bi-grams are used to represent the same information. By merging all the grams into one indexing scheme, we rely on the features to match across different n-grams, as long as they are visually similar. Thus, the indexing is mostly *insensitive* to character segmentation errors.

The indexing scheme we use is a combination of KDTrees and HKM, provided by the FLANN software [10]. The entire feature set from our book collection comprising of 12 GB cannot be indexed in one instance. Hence, the data is divided into six subsets, each of which is indexed separately. Each subset is then matched against every other subset and the (approximate) nearest neighbors are aggregated. Typical indexing time is 7 minutes per subset, while the looking-up of nearest neighbors for a given pair of index and data-subset is about 5 minutes.

The indexing accuracy was evaluated using the precision-recall measures. For a given n-gram sequence, the corresponding ranked list of approximate NNs is obtained from the indexing scheme. If the given query has R relevant documents, as given by the groundtruth, the precision is calculated over the top R retrieved results. The Precision@R values for different n-gram lengths are presented in Table II. It can be seen that the precision is high at the character and bi-gram level, as well as at larger n-grams, but is poor for smaller n-grams (3-7). The recall is poor for the smaller n-grams because the number of occurrences of each character are typically in many thousands, while only a few hundred are retrieved from the index. The recall improves with larger n-grams, with a recall of 60% at the 10-gram, resulting in a 70% precision. Thus, bigger n-grams are more reliable than smaller n-grams, hence they are weighted more in ranking retrieval results.

C. Retrieval

The retrieval performance was evaluated over a set of random queries, posed as a corresponding query from the collection. Each query is expanded to its n-grams, which are in-turn queried for in the index. The retrieved list for each n-gram is ranked using the weighting function discussed in Section II-B. The final ranked list is labeled as being relevant to the query, or not. The Recall-Precision plot for our retrieval system is shown in Figure 3. The area under the PR curve, or the mean average precision (mAP) for our system is a respectable 0.64.

n-gram	No. Occurrences	Precision @ R	Recall
1	1.5M	0.74	0.01
2	1M	0.60	0.06
3	0.6M	0.34	0.13
4	0.4M	0.28	0.22
5	0.2M	0.33	0.28
6	0.15M	0.40	0.35
7	92K	0.48	0.41
8	50K	0.55	0.47
9	27K	0.61	0.53
10	14K	0.70	0.60

Table II
PERFORMANCE OF THE N-GRAM INDEXING AND RETRIEVAL SCHEME THAT WE EMPLOY IN OUR RETRIEVAL SYSTEM. RESULTS SHOWN HERE ARE FOR THE GROUNDTRUTHED ENGLISH DATASET.

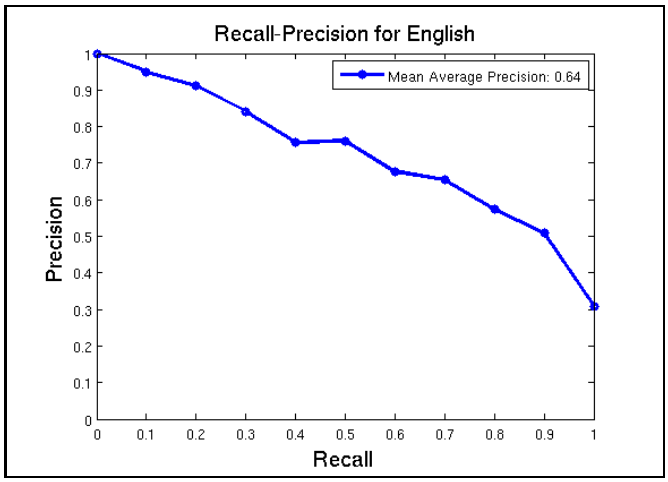

Figure 3. PR Curves for example queries from English dataset.

The value of our mAP is poorer than classical word-spotting, which would typically result in an mAP of 0.8. This is mostly due to the introduction of noise from similar looking, but distinctive n-grams. For example, there are many similar looking n-grams, such as $\{boy, toy\}$, $\{com, coin\}$, etc., that cannot be disambiguated using the n-gram itself. Such visually similar n-grams get clustered together. During query time, if the context of the rest of the word cannot disambiguate between such similar n-grams, errors are found in the retrieval results. However, the loss in accuracy is an acceptable cost to alleviating the need to label/train for entire vocabularies.

Some example retrieval results are shown in Figure 2. Only those words that matched the query at the sub-word level are shown here for brevity. Such a sub-word retrieval was hitherto not possible in a completely recognition-free

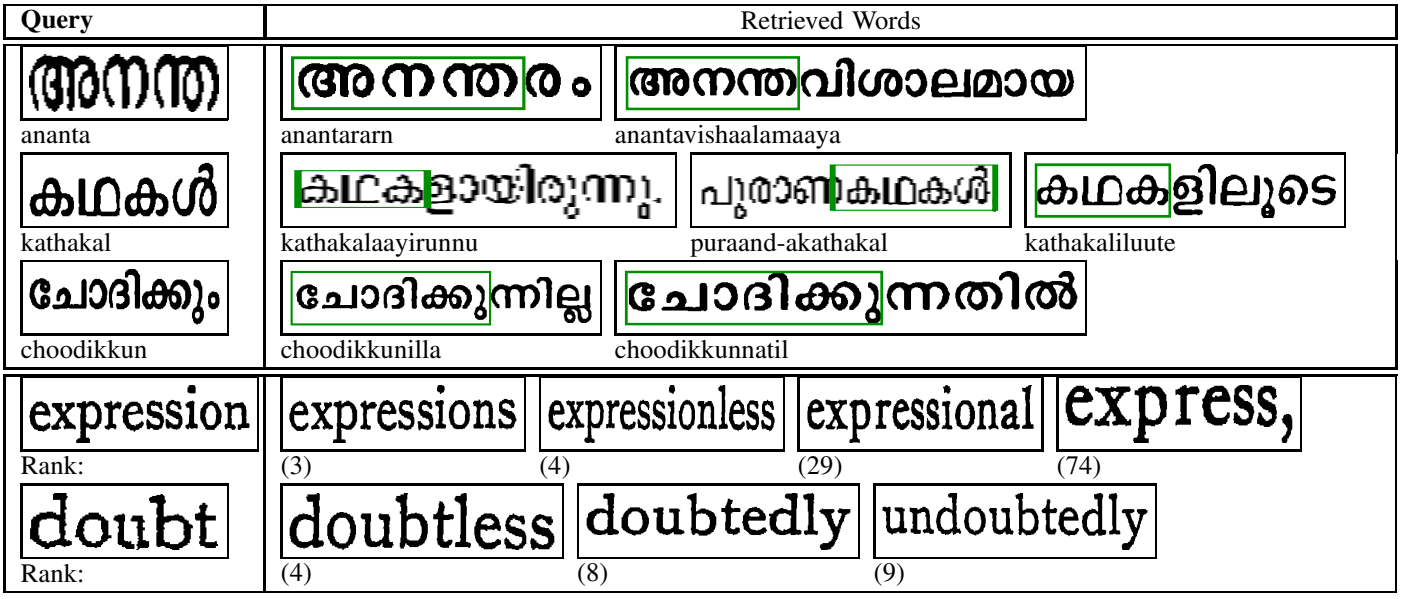

Figure 2. Example retrieval results from Malayalam and English document images. For the query images of on the left column, the retrieved results are given on the right. The relevant sub-word to the given query is highlighted in green for Malayalam results. The corresponding ITRANS notation for the Malayalam words is given below the word-images. The rank in the retrieved list for the English words, is given below for the word-images. Clearly, our approach enables retrieval for word-images that match at the sub-word level. Similar results are not possible to obtain in a regular word-spotting framework.

setup. Clearly, our approach holds much promise both in performance as well as practical utility.

IV. Conclusions

In this paper, we presented a new approach to recognition-free retrieval from document image collections. We overcome major issues with both OCR and word-spotting approaches. Our approach allows for out-of-vocabulary retrieval, where (labeled) words that were not seen during training/indexing could also be searched for in the collection. The framework also performs sub-word retrieval, without an explicit word-morphology analyzer. The technique is demonstrated on multiple books from both English and Malayalam languages. The results were found to be very promising. Our approach is also directly applicable to other scripts and languages.

V. Acknowledgments

Pramod Sankar would like to thank Doug Oard for valuable advice on work and life, at SIGIR'10. This work is supported by Ministry of Communication and Information Technology, Govt. of India.

References

[1] S. V. Rice, G. Nagy, and T. A. Nartker, *Optical Character Recognition: An Illustrated Guide to the Frontier*. Kluwer, 1999.

[2] V. Govindaraju and S. Setlur, Eds., *Guide to OCR for Indic Scripts*. Springer, Sep 2009.

[3] T. Rath, R. Manmatha, and V. Lavrenko, "A search engine for historical manuscript images," in *Proc. SIGIR*, 2004, pp. 369–376.

[4] T. Konidaris, B. Gatos, K. Ntzios, I. E. Pratikakis, S. Theodoridis, and S. J. Perantonis, "Keyword-guided word spotting in historical printed documents using synthetic data and user feedback," *IJDAR*, vol. 9, no. 2-4, pp. 167–177, 2007.

[5] Pramod Sankar, K. and C. V. Jawahar, "Probabilistic reverse annotation for large scale image retrieval," in *Proc. CVPR*, 2007.

[6] Pramod Sankar K., C. V. Jawahar and R. Manmatha, "Nearest Neighbor based Collection OCR," in *Proc. DAS*, 2010.

[7] A. Balasubramanian, M. Meshesha, and C. V. Jawahar, "Retrieval from document image collections," in *Proc. DAS*, 2006, pp. 1–12.

[8] T. M. Rath and R. Manmatha, "Word spotting for historical documents," *IJDAR*, vol. 9, no. 2-4, pp. 139–152, 2007.

[9] D. Nister and H. Stewenius, "Scalable recognition with a vocabulary tree," in *Proc. CVPR*, 2006, pp. 2161–2168.

[10] M. Muja and D. G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration," in *Proc. VISAPP*, 2009, pp. 331–340.

[11] C. V. Jawahar and A. Kumar, "Content-level annotation of large collection of printed document images," in *Proc. ICDAR*, 2007, pp. 799–803.

[12] J. A. Rodriguez and F. Perronnin, "Local gradient histogram features for word spotting in unconstrained handwritten documents," in *Proc. ICFHR*, 2008.

[13] E. Ataer and P. Duygulu, "Matching Ottoman words: An image retrieval approach to historical document indexing," in *Proc. CIVR*, 2007, pp. 341–347.