

A novel approach for Word Spotting using Merge-Split Edit Distance

Khurram Khurshid¹, Claudie Faure² and Nicole Vincent¹

¹ Laboratoire CRIP5 – SIP, Université Paris Descartes, 45 rue des Saints-Pères,
75006, Paris, France

{khurram.khurshid, nicole.vincent}@mi.parisdescartes.fr

² UMR CNRS 5141 - GET ENST, 46 rue Barrault
75634 Paris Cedex 13, France

cfaure@enst.fr

Abstract. Edit distance matching has been used in literature for word spotting with characters taken as primitives. The recognition rate however, is limited by the segmentation inconsistencies of characters (broken or merged) caused by noisy images or distorted characters. In this paper, we have proposed a Merge-split edit distance which overcomes these segmentation problems by incorporating a multi-purpose merge cost function. The system is based on the extraction of words and characters in the text and then attributing each character with a set of features. Characters are matched by comparing their extracted feature sets using Dynamic Time Warping (DTW) while the words are matched by comparing the strings of characters using the proposed Merge-Split Edit distance algorithm. Evaluation of the method on 19th century historical document images exhibits extremely promising results.

Keywords: Word Spotting, Edit Distance, Dynamic Time warping

1 Introduction

Word spotting on Latin alphabets has received considerable attention over the last few years. A wide variety of techniques have been proposed in literature but the field still remains inviting and challenging specially if the document base comprises low quality images of historical documents. There are plenty of issues and problems related to ancient printed documents which are discussed in detail in [4] and [5]. These include physical issues such as quality of the documents, the marks and strains of liquids, inks and dust etc; and semantic issues such as foreground entity labeling. In this paper though, we will concentrate only on the segmentation related problems. The main focus of this paper will be the improvements in word spotting to make it clear of a good character segmentation requirement, so that if we do not get a good segmentation, we can still obtain better word retrieval rates. A brief account of some state of the art methods in word spotting that inspired our own work is given here:

Adamek et al. [2] introduced an approach based on the matching of word contours for holistic word recognition in historical handwritten manuscripts. The closed word contours are extracted and their multiscale contour-based descriptors are matched using a Dynamic Time Warping (DTW) based elastic contour matching technique. In [12], Rath et al. argued that word spotting using profile feature matching gives better results. Words are extracted from the text and are represented by a set of four profile features. Feature matching is performed using the scale invariant DTW algorithm. In [9], we showed that, for historical printed documents, instead of using word features, character features give a better and distinct representation of a word. Two words are compared by matching their character features using DTW [8]. The method can be improved by adding Edit distance matching for the words. It means that the features of two characters are matched using DTW while track of these character matching costs between the two strings is kept by Edit distance. Addition of the Edit distance stage significantly reduced the number of false positives while improving the recognition rate as later shown in results. However, if the character segmentation is not good there will be a significant drop in retrieval rate as the classic edit distance does not cater for merged and broken characters.

Different variations of Edit distance have been proposed in literature for different matching applications [3,7,11,13]. Kaygin et al. introduced a variation of Edit distance for shape recognition in which polygon vertices are taken as primitives and are matched using the modified Edit distance. The operations of inserting and deleting a vertex represent the cost of splitting and combining the line segments respectively [7]. A minimal edit distance method for word recognition has been proposed by [13]. Complex substitution costs have been defined for the Edit distance function and string-to-string matching has been done by explicitly segmenting the characters of the words. Another recent variant of Edit distance has been proposed in [11] where apart from the classic substitution costs, two new operations namely combination and split are supported.

This paper presents an effective way for word spotting in historical printed documents using a combination of our merge-split Edit distance variant and an elastic DTW algorithm. Main aim of this work is to be able to match words without the need of having perfect character segmentation *a priori*. Paper is divided into multiple sections beginning with the overview of the proposed system in the next section. This is followed by description of method and its different stages and the results achieved.

2 Proposed Method

The model is based on the extraction of a multi-dimensional feature set for the character images. As opposed to [12] where features are extracted at word level, we define features at character level, thus giving more precision in word spotting [8].

Figure 1 illustrates the different stages of our system. Document image is binarized using NICK algorithm [10]. Text in the document image is separated from graphics and words in the text area are extracted by applying a horizontal RLSA [14] and finding the connected components in that RLSA image. Characters in a word image are found by performing its connected component analysis followed by a 3-

A novel approach for Word Spotting using Merge-Split Edit Distance

step post-processing stage. For each character, we define 6 feature sequences which represents a character's overall form. Query word is searched within candidate words by a combination of our Merge-split Edit distance at word level and DTW at character level which allows us to spot even the words with improper character segmentation.

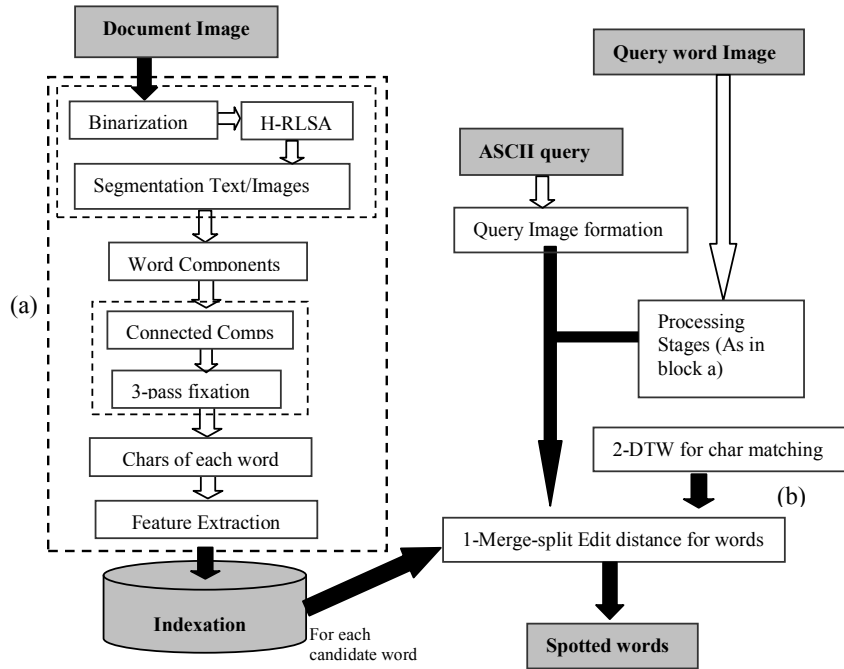


Fig. 1. a) Document image Indexing involving character segmentation and feature extraction. b) Word retrieval using a combination of DTW and Merge-split edit distance

3 Document Indexing

In this section, we only give an overview of the indexing process involving character segmentation and feature extraction.

The connected components (CCs) of the binary image do not always correspond to characters. A character may be broken into multiple components or multiple characters may form a single component. For that, we have a 3-pass processing stage for the CCs in a word component to get the characters. In the first pass, components on the top of each other are assigned to the same character (fig 2a). In the 2nd pass, overlapping components are merged into a single character (fig 2a). In the 3rd pass, the punctuation marks (like ‘,’ ‘.’) are detected using size and location criteria and are removed from the word (fig 2b). After the three passes, there still remain some improperly extracted characters which are of main interest in this paper (fig 2c,3).

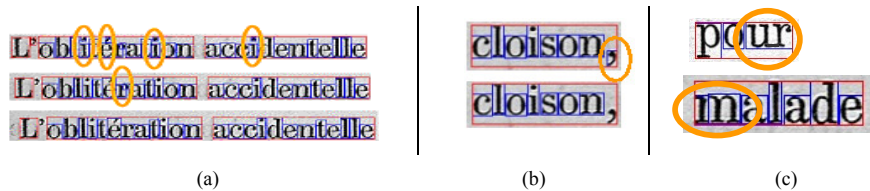


Fig. 2. a) Pass 1 & 2 b) Pass 3 c) Some examples of the remaining unfixed characters

Each character is represented by a sequence of six feature vectors. The length of each of the six vectors associated with a character is equal to the width of the character bounding box. The six features we use are *Vertical projection profile* (on gray level image), *Upper character profile position*, *Lower character profile position*, *Vertical histogram*, *Number of ink/non-ink transitions* and *Middle row transition state* (on binary image). Document processing/indexing is done offline, creating an index file for each document image. The coordinates of each word, number of characters in the word, the position and the features of each character are stored in the index files. One advantage of having an index file is that the query word can now be crisply selected by just clicking on the word in the GUI of our document processing system. This query word is processed in the same way to extract character features. We can also type in the input query word instead of a click-selection. In that case, features of the prototype characters (selected manually offline) are used for matching. The segmentation of the query characters is perfect in this case.

4 Word Retrieval

For word spotting, we have proposed a two step retrieval system. In the first step, a *length-ratio filter* finds all eligible word candidates for the query word. For two words to be considered eligible for matching, we have set bounds on the ratio of their lengths. If this ratio does not lie within a specific interval, we do not consider this word as a candidate. Through this step, we are on average able to eliminate more than 65% of the words. In the second step, the query word and candidate word are matched using a multi-stage method in which the characters of the two words are matched using the Merge-Split Edit distance algorithm while features of the two characters are matched using elastic DTW method coupled with Euclidean distance. The need for an algorithm catering for the merge and split of characters arises because we may not have 100% accurate segmentation of characters all the time (fig 3).

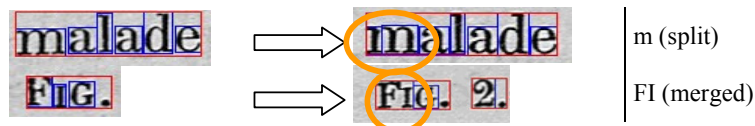


Fig. 3. Query words and some instances of them which were not spotted without Merge-split

4.1 Word Matching – Merge Split Edit Distance

To address character segmentation issues, we have introduced two new merge-based character matching operations **Merge1** and **Merge2** that enable to model a merge and split capability respectively in Edit distance, thus overcoming the limitation of having a perfect character segmentation by catering for the broken and merged characters during word matching.

Consider 2 words A and B; A, the query word, having s characters while B, the test word, having t characters. We treat both words as two series of characters, $A = (a_1 \dots a_s)$ and $B = (b_1 \dots b_t)$. To determine the distance/cost between these two character series, we find the Edit matrix W which shows the cost of aligning the two subsequences. Apart from the three classic Edit operations, we have introduced two new operations $a_i \rightarrow (b_j + b_{j+1})$ and $(a_i + a_{i+1}) \rightarrow b_j$ which represent merge1 and merge2 respectively. Merge1 function allows one character of the query word to be matched against two characters of the current test word, while merge2 function allows one character of the test word to be matched against a combination of two query word characters thus modeling a split b_j . Combination of two characters is done by concatenating the 6 feature sequences of both of them. The entries of matrix W initialized by +infinity are calculated as:

$$\begin{aligned}
 W(0,0) &= 0 \\
 W(0,j) &= W(i,j-1) + \gamma(\Lambda \rightarrow b_j) \\
 W(i,0) &= W(i-1,j) + \gamma(a_i \rightarrow \Lambda) \\
 W(i,j) &= \min \left\{ \begin{array}{l} W(i-1,j-1) + \gamma(a_i \rightarrow b_j) \\ W(i-1,j-1) + \gamma(a_i \rightarrow (b_j + b_{j+1})) \quad (\text{for } j < t) \\ W(i-1,j-1) + \gamma((a_i + a_{i+1}) \rightarrow b_j) \quad (\text{for } i < s) \\ W(i-1,j) + \gamma(a_i \rightarrow \Lambda) \\ W(i,j-1) + \gamma(\Lambda \rightarrow b_j) \end{array} \right\} \quad (1)
 \end{aligned}$$

Here, Λ is an empty character with all feature vector values set to 0. $\gamma(a_i \rightarrow b_j)$ is the cost of changing a_i to b_j , $\gamma(a_i \rightarrow \Lambda)$ is the cost of deleting a_i and $\gamma(\Lambda \rightarrow b_j)$ is the cost of inserting b_j .

$\gamma(a_i \rightarrow (b_j + b_{j+1}))$ shows the cost of changing character a_i of query word to two character components $b_j + b_{j+1}$ of the test word. It means that if the character b_j was broken into two components b_j and b_{j+1} , we would be able to match this with a_i using Merge1 function. The feature sequences of b_j and b_{j+1} are concatenated and are matched with the feature vectors of a_i to get $W(i,j)$. Once $W(i,j)$ is calculated, we copy the same value of $W(i,j)$ to the cell $W(i,j+1)$ signifying that we had used the merge function.

Similarly, $\gamma((a_i + a_{i+1}) \rightarrow b_j)$ shows the cost of changing two characters of query word to one character of test word. It means that if b_j was in fact a component having two characters merged into one, we would be able to detect and match that with $a_i + a_{i+1}$ using our Merge2 function. Here, instead of splitting the feature vectors of b_j (which is more difficult as we do not know exactly where to split), we merge the query word characters, thus emulating the split function. $W(i,j)$ is calculated the same way and it is copied to the cell $W(i+1,j)$ signifying the use of the split function.

4.2 Character matching cost - DTW

All the character matching costs are calculated by matching the feature sequences of two character components using DTW. The advantage of using DTW here is that it is able to account for the nonlinear stretch and compression of characters. Hence two same characters differing in dimension will be matched correctly.

Two characters X and Y of widths m and n respectively are represented by vector sequences $X = (x_1 \dots x_m)$ and $Y = (y_1 \dots y_n)$ where x_i and y_j are vectors of length 6 (= No. of features). To determine the DTW distance between these two sequences, we find a matrix D of m x n order which shows the cost of aligning the two subsequences. The entries of the matrix D are found as:

$$D(i, j) = \min \left\{ \begin{array}{l} D(i, j-1) \\ D(i-1, j) \\ D(i-1, j-1) \end{array} \right\} + d(x_i, y_j) \quad (2)$$

Here for $d(x_i, y_j)$, we have used the Euclidean distance in the feature space:

$$d(x_i, y_j) = \sqrt{\sum_{k=1}^p (x_{i,k} - y_{j,k})^2} \quad (3)$$

where p represents the number of features which in our case is six.

The entry $D(m, n)$ of the matrix D contains the cost of matching the two characters. To normalize this value, we divide the final cost $D(m, n)$ by the average width of the two characters.

$$Final\ char-cost = D(m,n) / [(m+n)/2] \quad (4)$$

4.3 Final word distance

Once all the values of W are calculated, the warping path is determined by backtracking along the minimum cost path starting from (s, t) while taking into the account the number of merge/split functions used in the way. The normalization factor K is found by subtracting the number of merge/split functions from the total number of steps in the warping path. The final matching cost of the two words is:

$$Final\ word-cost = W(s,t) / K \quad (5)$$

Two words are ranked similar if this final matching cost is less than an empirically determined threshold.

Figure 4 shows an example of matching two words of lengths 4 and 3. The query word is well segmented while the last two characters in the test word are merged into one. While finding W, we see that one merge operation is used for matching 'ur' with 'u', thus decrementing the value of K by one.

A novel approach for Word Spotting using Merge-Split Edit Distance



W		Test word 				Final Word Cost
Query Word		Λ	p	o	ur	= $W(4,3) / K$
	Λ	0.00	1.79	3.39	5.56	$W(4,3) = 0.09$
	p	1.78	0.02	1.62	3.79	$K = \text{steps} - \text{merge/splits used}$
	o	3.47	1.72	0.04	2.05	$= 4 - 1 = 3$
	u	5.51	3.75	2.08	0.09	Final cost = $0.09/3 = 0.03$
	r	6.85	5.10	3.42		

Fig. 4. Calculating Edit Matrix W for two similar words of lengths 4 and 3, and their final cost

5 Experimental Results

The comparison of character-feature based matching technique with other methods (method in [12] and correlation based matching method) has already been done in [9]. Here we compare our merge-split Edit distance based matching method with a similar method but using classic Edit distance in place (we will refer to it as [Ed]) and also with [9] to show the improvement in results. We also give the results of a professional OCR software Abbyy FineReader [1] on the same data set. Experiments were carried out on the document images provided by BIUM [6]. For experiments, we chose 48 pages from 12 different books (4 images from each book), having a total of more than 17,000 words in all. For testing, 60 different query lexiques having 435 instances in total were selected based on their varied lengths, styles and also context of the book. Table 1 summarizes the results of word spotting.

Table 1. Result Summary of Word matching algorithms

	Method in [9]	Classic Edit [Ed]	Our method	ABBY [1]
#query word instances	435	435	435	435
#words detected perfectly	401	406	427	422
#words missed	34	29	8	13
#False positives	51	16	4	0
Precision %	88.71%	96.20%	99.01%	100%
Recall %	92.18%	93.34%	98.16%	97.01%
F-Measure	90.41%	94.75%	98.58%	98.48%

We can see from the above table that the presented merge-split Edit distance based method achieves a much higher recognition rate while maintaining a better precision than [9] and [Ed]. This is due to the fact that we are able to spot even those words where the segmentation of characters is not good. The layout of the pages used was not difficult to manage for a professional OCR. So on the same data-set, [1] produced a precision of 100% but we still obtained a better recall rate than it.

We also analyzed the effect of word length on the Edit distance threshold, learning that for smaller words (with lesser number of characters), a lower threshold gives better precision and recall rates, while for longer words, a higher threshold

value proves to be more effective. This is because for longer words, it is more unlikely to find similar words; so we can allow a bit more relaxation in threshold value. Another thing we learned is that the time taken to search a query increases linearly with the number of documents searched.

Considering optimum threshold values (found empirically), our system achieves a precision of 99% while obtaining an overall recognition rate of 98.16%.

6 Conclusion

We have proposed a new approach for word spotting based on the matching of character features by employing a combination of DTW and Merge-Split Edit distance. The main objective of this approach is to cater for the improper segmented characters during the matching process. Results obtained using this method are very encouraging. The number of query words missed is very less as compared to [9] and [Ed] which shows the prospects of taking this method even further by improving different stages and adding more features to achieve even higher percentages.

References

1. ABBYY FineReader professional v6.0
2. Adamek, T., O'Connor N.E., Smeaton, A.F.: Word matching using single closed contours for indexing handwritten historical documents, IJDAR (2007).
3. Ambauen R., Fischer S., Bunke H.: Graph Edit Distance with Node Splitting and Merging and its Application to Diatom Identification, Lecture Notes in CS, (2003).
4. Antonacopoulos, A., Karatzas, D., Krawczyk, H., Wiszniewski, B.: The Lifecycle of a Digital Historical Document: Structure and Content. ACM Symposium on DE, 2004.
5. Baird, H.S.: Difficult and urgent open problems in document image analysis for libraries, 1st International workshop on Document Image Analysis for Libraries, (2004).
6. Digital Library of Bibliothèque Interuniversitaire de Médecine, Paris, <http://www.bium.univ-paris5.fr/histmed/medica.htm>
7. Kaygin, S. & Bulut, M. M.: Shape recognition using attributed string matching with polygon vertices as the primitives, Pattern Recognition Letters, (2002).
8. Keogh E., Pazzani M.: Derivative Dynamic Time Warping, First SIAM International Conference on Data Mining, Chicago, IL (2001).
9. Khurshid K., Faure C., Vincent N.: Feature based word spotting in ancient printed documents. In proceedings of PRIS (2008).
10. Khurshid K., Siddiqi I., Faure C., Vincent N.: Comparison of Niblack inspired binarization techniques for ancient document images. 16th International conference DDR, (2009).
11. Manolis, C. & Brey, G.: Edit Distance with Single-Symbol Combinations and Splits Proceedings of the Prague Stringology Conference, (2008).
12. Rath, T. M., Manmatha, R.: Word Spotting for historical documents. IJDAR (2007)
13. Waard, W.P.: An optimised minimal edit distance for hand-written word recognition, Pattern Recognition Letters, (1995).
14. Wong K. Y., Casey R. G., Wahl F. M.: Document analysis system. IBM J. Res. Development, (1982).