Digit/Symbol Pruning and Verification for Arabic Handwritten Digit/Symbol Spotting

Nicola Nobile, Chun Lei He, Malik Waqas Sagheer, Louisa Lam, Ching Y. Suen
CENPARMI (Centre for Pattern Recognition and Machine Intelligence)
Computer Science and Software Engineering Department, Concordia University
Montreal, Quebec, Canada
{nicola, cl_he, m_saghee, llam, suen}
@cenparmi.concordia.ca

Abstract—In order to spot the digits in a handwritten document, each component is sent to a classifier. This is a time consuming process because a document usually contains several hundred components. A method is presented to reduce the number of candidate components from a handwritten document sent to the classifier. Furthermore, since the classifier does not contain a rejection class, this led to several misclassifications. To lessen this, a verification post processing module was developed in order to reject some false positives. We reached an overall precision of 80% and 83.3% recall on our test set of handwritten documents.

I. Introduction

In the paper [**1**], the authors presented a word spotting technique to spot words in handwritten Urdu documents. In this paper, we propose a similar methodology but are more concerned with finding the digits 0…9 and some special symbols ("@", "/", and "#") in a handwritten Arabic document. Digit spotting has its own set of challenges to overcome as we address these issues in our methodology and explain how we overcome them.

Generally, a digit spotting method involves a segmentation routine working in concert with a classifier. As the segmentation module finds a connected component, it sends it to the classifier for identification. If the classifier returns a successful classification, that result will effectively signal that that component is one of the target classes.

Several problems arise from this methodology. As a handwritten document can contain several hundred connected components, this can lead to a very slow system since the classifier is usually the slowest module in the system. Secondly, some classes are very similar in shape and form to non-target classes (i.e. the digit "1" with the letter Alif). Lastly, if the classifier does not contain a rejection class, this will cause several misclassifications, which are undesirable.

The remainder of this paper describes the goal and challenges of our research in digit and symbol spotting. This is followed by the procedures implemented to reduce the number of candidate components sent to the classifier and how we compensate for the lack of a rejection class. This is followed by our results from some handwritten documents written by unknown writers. We end by discussing our conclusions and future work that can be done to our system in order to improve results.

II. Objectives and Challenges

The goal of our digit and symbol spotting system is to locate and recognize the digits and some special symbols in a handwritten document. We developed our system for five languages (Arabic, Dari, Farsi, Pashto, and Urdu). However, in this paper we will focus our discussion on Arabic since the procedure and results are similar with the other languages.

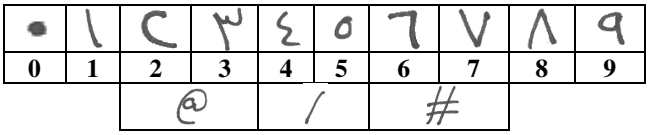

Figure I: Digit and Symbol Spotting Target Classes

Figure I shows our target classes for digit and symbol spotting in a handwritten document. Documents can contain digits, symbols, words, and other components which do not belong to our target classes.

The challenges encountered with such a task include excluding those components in a document that look similar to a target class, but are not. For example, the digit one (١) is similar in shape and form with the characters "/", "\", "(", ")", "[", "]", "{", "}", "l", "!", and Arabic character Alif (ا). The Alif character poses the biggest difficulty because it occurs very frequently in Arabic text. Also, the digit five (٥) is very similar to the character hey (ه). Our pruning module (described in the next section) filters out most of the unwanted components.

The classifier we used was described in [2]. This classifier was trained and tested on the CENPARMI Arabic Digits Database [3] which is a database of isolated digits and symbols. The classifier is used as part of the spotting system. However, because of the slower speed of the classifier, sending every connected component to it resulted in a very slow system. Therefore, our pruning module filtered out those unlikely candidates thereby speeding up the overall system.

Another characteristic of the classifier was that when it was trained and tested, it was not given unseen data and therefore, a rejection class was unnecessary. This caused several misclassifications from spotting since generally, the majority of the candidate components in a document do not belong to any of our target classes. Therefore, we required a type of rejection to compensation for this. Our solution was to build a verification module which accepts or rejects the classifier output based on features that were extracted from the CENPARMI Arabic Digit Database [3].

III. Methodology

The flowchart of our overall system is shown in Figure

1520-5363/11 $26.00 © 2011 IEEE
DOI 10.1109/ICDAR.2011.136

II. The preprocessing performs noise removal, document skew correction, and line detection. This is followed by a module which finds all the connected components in the document. This includes larger base components as well as the diacritics. All the connected components are sent to the pruning module (described in Section III-A).

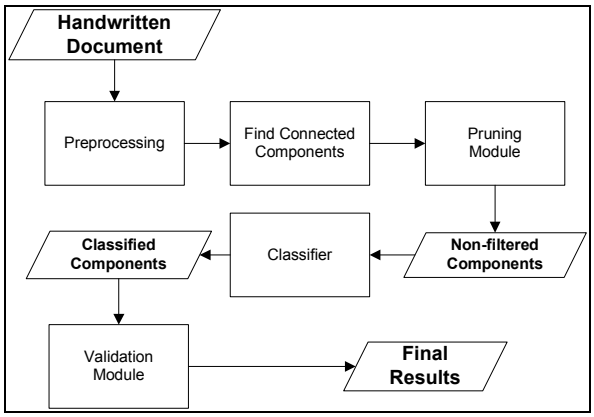

Figure II: Digit/Symbol Spotting Application Flowchart

Components that the pruning module does not filter out are sent to the classifier for recognition. Of these, those components which have a high classifier confidence are then sent to the verification module (described in Section III-B) to provide a final decision on the spotted components.

A. *Pruning*

The primary function of the pruning module is to reduce the number of candidate words that are sent to the classifier. Additionally, it contains rules to handle the special cases to distinguish between the digit one and the components similar to it and with the digit and five and the letter hey.

Pruning was done by comparing statistical features obtained from our training set with the same features extracted from the candidate component. The training set we used is the CENPARMI Arabic Digits Database [3]. Similar pruning modules were created by using CENPARMI training data for Dari [4], Farsi [5], Pashto [6], and Urdu [7] for documents written in those languages.

We created 40 heuristics that were created from statistical features we extracted from the training set and from line positional information we extracted from handwritten training documents. Examples of features include number of end points, number of intersection points, width, height, and density. For dimensional features such as width, we compare the width of a component to the average width of all the components that reside on the same line. This was done to provide a more dynamic method to remove unlikely candidates. It has the advantage of conforming to a test subject's handwriting style.

Information we extract from the training set are lower and upper bound values of structural features such as width/height, density of black pixels, texture information, etc. For example, one heuristic compares the width/height ratio of the classes. The width/height ratios of all our training samples are calculated as well as the following values:

$$r_{min} = min\left(\frac{width_i}{height_i}\right), r_{max} = max\left(\frac{width_i}{height_i}\right), \quad for\ i = 1 \ldots n$$

For Arabic, we obtained the values r_{min} = 0.01613 and r_{max} = 11.2174. We use these values to prune out components unlikely to be a target class. For each candidate component, we calculate its width/height ratio, r_{cand}, and then compare it with the training values. For this heuristic, if $r_{cand} < r_{min}$ or $r_{cand} > r_{max}$, then this heuristic would reject the candidate. A final decision has not been made on the fate of this component since we have 39 other heuristics to evaluate.

Note that we do not perform normalization because we want to preserve the original handwritten aspect ratios of the writer's handwriting.

Another heuristic compares the width of a component with the average width of all the components on the same line it resides on. In this case, we do not use the isolated database as the previous heuristic, but we use full page handwritten documents for training.

From this training set, we locate the target class components in the documents. For each of the components, we find their minimum and maximum ratios of the width over the average width. The average width is computed from those components on the same line only. This was decided because people tend to write smaller or larger based on how many words are on a line. For digits we obtained a lower bound of 0.1682 and an upper bound of 2.7543. This means that for a potential candidate with width w, and if the average width of all the components on the line it resides on is w_{avg}, then we compute the ratio $r = w / w_{avg}$. This heuristic will reject the candidate if r<0.1682 or r>2.7543.

For the structural features such as intersection points and end points, we determine these values based on the skeleton image of the connected component. The CENPARMI isolated digits were used to obtain these lower and upper bounds.

For positional features, the full page handwritten training documents were used to find vertical positions of the target components based on landmarks of the line it resides in. Some positional ratio features include *top/midline*, *bottom/midline*, *top/baseline*, and *bottom/baseline*. Both the midline and the baseline are reference values and the ratio measures the vertical position of the component with respect to the landmark lines as seen in Figure III.

Once all the heuristics have been evaluated, we send the results to a rule which decides if the candidate will be pruned out or not. This is based on a subset of heuristics which we found to be optimal when they were evaluated using our validation documents. Out of the 40 heuristics, 18 were chosen as useful for pruning. The others were useful but were not included for reasons such as redundancy or similarity with an existing heuristic. The rule will fail if any one of the heuristics is false (i.e. if a heuristic rejects the

candidate). If the rule fails, we prune out the candidate from further processing.

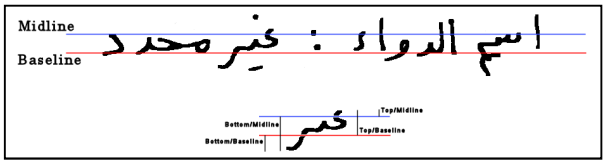

Figure III: Positional Pruning Heuristics

The candidates that pass the rule will be sent to the special pruning sub function which will identify pruning cases such as the colon symbol, symbols which are similar to one (i.e. Alif, "[", …), or components which pass the pruning rule but cannot possibly be a digit or symbol because of its location or position with respect to its neighbouring components. These special cases are not represented in our set of 40 heuristics, so we must identify them separately. Once a candidate passes this step, it is sent to the classifier for recognition.

We did not use a classifier such as a neural network or SVM because our goal was to avoid using our classifier (SVM) as much as possible because of the slower speed.

B. Verification Module

Since our classifier does not have a rejection class, we created a verification module which takes the classifier output and determines if the recognized result is likely correct. We want to reduce the false positives originating from the classifier. Similar to the pruning module, we extracted structural features from our CENPARMI isolated digits and symbols. However, in this case, we built models for each individual class because we now need to verify a specific class.

The verification module is based on several features such as 8-directional freeman chain codes [8], image difference, and other structural features.

Chain codes are features that give a description or shape of each connected component. We used the CENPARMI isolated training digits and symbols in order to match the target sample with its respective class. To compare the candidate chain code with the trained target class chain code, the Mahalanobis distance [9] measurement method was used.

A modified Zheng-Suen [10] skeletonization method was used to generate a skeleton of each image. The chain code algorithm starts from the top right pixel and traces along a sequence of skeleton pixels based on counter clockwise eight directions. The direction of each pixel 'P' was encoded by numbers (1, 2, 3, ... 8), as shown in Figure IV.

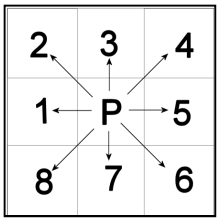

Figure IV: Eight Directional Freeman Chain Code

A skeleton image of an Arabic digit 2 is shown in Figure V.

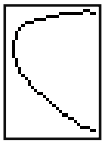

Figure V: Skeleton Image of an Arabic Digit '2'

The chain code tracing for small portion of skeleton image of above image is shown in Figure VI.

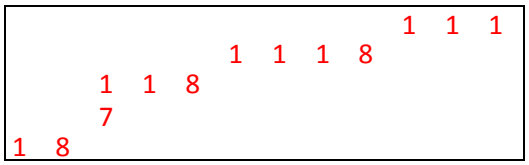

Figure VI: Chain Codes for Small Portion of Figure V

After creating the chain code from each image, a histogram of each direction was acquired and is shown in Table I. The first row represents the directions and the second row represents the power of each direction.

TABLE I: CHAIN CODE DIRECTION HISTOGRAM

Direction:	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Strength:	18	0	0	0	7	21	11	7

Once the histogram of directions for each image in the training set was created, the Mahalanobis distance measurement was used for accepting or rejecting any candidates. Mahalanobis distance measurement [11] [12] is based on the correlation between variables and it is very efficient for determining the similarity of an unknown data set to a known one. The Mahalanobis distance '*r*' between test sample $x=(x_1, x_2, x_3, \ldots x_n)$ and the mean of known samples $\mu = (\mu_1, \mu_2, \mu_3, \ldots \mu_n)$ is given by:

$$r = \sqrt{(x-\mu)^t \sum^{-1} (x-\mu)} \qquad (1)$$

where '$\sum^{-1}$' is the inverse of the covariance matrix '$\sum$' of known samples and the size of covariance matrix is $n \times n$. The symbol 't' represents the transpose.

In our experiments, the size of a covariance matrix was 8x8. For each image in the CENPARMI digit and symbol

training set, a histogram of chain codes was created. This resulted in matrices of histogram features. A mean vector (μ) was created for each class by taking the mean of each column from the feature matrix of that class. A covariance matrix was also created for each class from that histogram feature matrix. After obtaining the covariance matrix '$\sum$' and the mean vector (μ), each target sample was tested with its respective known class. A threshold value was obtained from the training samples in order to reject or accept the target sample. If the distance is greater than a predetermined threshold, we reject the classifier output class.

In addition to the chain code feature, we also include a few structural features used to verify the output. The features used are number of endpoints, number of intersection points, and number of loops in the skeleton image. Even though we used endpoints and intersection points in the pruning stage, here we calculated these values for each specific class whereas we calculated the values for the entire set of classes as a group for pruning.

TABLE II: ACCEPTABLE STRUCTURAL VERIFICATION VALUES/CONDITIONS

Class	Endpoints	Intersection Points	Loops
2	2,3	0,1	0
3	3,4		0
6			0
7		0,1,2,3	0
8			0
#		4	1
0	a. Position must not be above any other component. b. If it is the last component in a line, it is preceded by a digit.		
1	a. Must have at least one neighbouring digit. b. If it is first component in a line, the next symbol is a colon.		
/	a. Must not be first or last component in a line. b. Must neighbour at least one digit. c. Must not contain two smaller components within its bounding box.		
Any	a. Classifier confidence greater than a threshold. b. Chain code distance below threshold		

End points and intersection points were extracted from the skeletons and loops were obtained from the chain codes and a minimum and maximum bound of acceptable values were determined for each class.

These features had great improvement in rejecting misclassifications between the ambiguous digits two and three. In addition, results for the pound sign, which was misclassified with other symbols/letters, was improved upon. Heuristic acceptable values and acceptable conditions for these classes are shown in Table II.

If the candidate does not comply with any one of these heuristics, it is rejected. The class "1" had strong rejection conditions because of the high confusion with the letter Alif, the slashes "\" and "/" as well as the bracketing symbols ("{", "}", "(", ")", "[", "]"). For zero, it should not be above any other component in order to avoid classifying small diacritics as that digit. The classifier does not use positional features so the verification module compensates for this.

IV. Results

Figure VII shows a magnified section of one of our sample documents where the spotted classes are highlighted.

We can see in Figure VII that most of the classes were spotted and several false components were rejected by the verification stage. We can see that several Alif characters that were accepted by the classifier, have now all been rejected by verification as well as the "(" and ")" components on the first line. Three of the colons ":" were removed from consideration by the pruning module. The other one was not because the two components that make up the colon were not overlapping enough to be pruned out. The components of that colon were classified are zeroes.

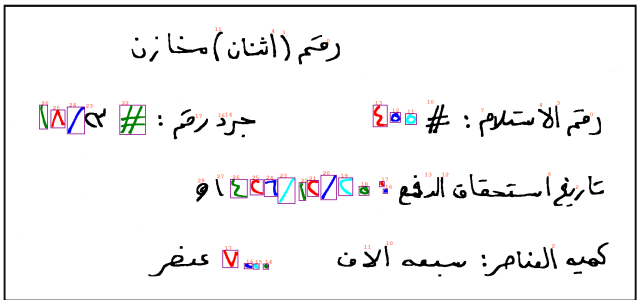

Figure VII: Highlighted Spotted Digits and Symbols

Unfortunately, we did encounter a few false negatives such as the first "#" in the second line. The touching digits component "23" in the second line was properly pruned out because that component did not comply with any of the pruning heuristics. The third line contains the digit "1" that was rejected because of a low classifier confidence value (0.40). Next to it, the digit "9" was rejected because the chain code distance (18.1) was greater than the threshold (2.0).

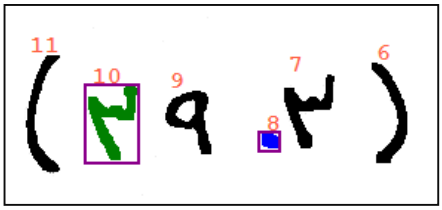

Figure VIII: Verification False Positives (Components 7 and 9)

In Figure VIII, two digits were filtered out by the verification module. Component 7 was rejected by the verification module because the classifier (incorrectly) recognized it as the digit "2". The features for this component did not match the verification model for the digit "2", so therefore, the verification module rejected it. The "9" was rejected because the distance between the component chain code feature and the chain code model for "9" (3.28) exceeded the threshold (3.00). Therefore, the verification module rejected it. The zero (component 8) and the three (component 10) were correctly passed by pruning,

were correctly classified, and accepted by the verification module. The other non-target class components were rejected by verification.

One limitation of our system is that it expects all the components it receives to be in Arabic. When a document contains non-Arabic text as in Figure IX, it will process it without language identification. We can see the English letter "o" was classified as class "5". The full-stop was recognized as zero and the "c" as the digit "2".

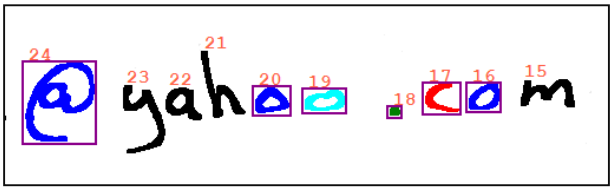

Figure IX: Results from Non-Arabic Text

We tested on two Arabic handwritten documents. Results for our entire combined system (pruning, classification, and verification) are shown in Table III. The writers for these test documents are not represented in our isolated digits/symbols training database or in our handwritten training documents.

The high true negative rate is an indication of (mostly) the pruning module and little bit of the verification module (since fewer candidates reach the verification step). This also accounts for the high accuracy rate. We have high precision and recall for the second document and overall, we have a very good result of 80% precision and 83.33% recall. Note that results were a bit higher before we added the verification module which has the job to reject. That includes, sometimes, rejecting some true positives. We prefer to reject rather than misclassify and overall, the verification module rejected more true negatives than true positives. Our results for the other languages are similar.

V. Conclusions

We have shown that we have developed a very robust digit and symbol spotting system by incorporating a pruning module, a classifier, and a verification module. Together, the three components generated a high overall result when tested on documents written by writers that were not trained on.

The limitations that we found were that the system does not recognize touching digits. They are correctly pruned out however since the component does not conform to any trained target class model - which is an isolated digit or symbol.

Future work can include identifying touching connected components and determining if one or both (or all) of them belong to one of our target classes. Furthermore, even though the symbols that were chosen are those that occur frequently in financial documents, we can add more symbols to our target set. Finally, we believe that better results can be achieved if we train on more documents. The current system was trained on only three documents, yet performed respectively. This is especially true for positional features that were extracted only from the documents and not the training set of isolated digits/symbols. By adding more training documents, we can possibly identify more circumstances for pruning and rejection.

TABLE III: ARABIC HANDWRITTEN DIGIT/SYMBOL SPOTTING RESULTS

	D4_W2	D5_W7	Overall
Target Classes:	20	54	74
Candidates:	523	529	1052
True Positives (TP):	10	50	60
True Negatives (TN):	495	470	965
False Positives (FP):	8	7	15
False Negatives (FN):	10	2	12
Precision:	55.56%	87.72%	80.00%
Recall:	50.00%	96.15%	83.33%
True Negative Rate:	98.41%	98.53%	98.47%
Accuracy:	96.56%	98.30%	97.43%

REFERENCES

1. Sagheer, M., Nobile, N., He, C., Suen, C.: A Novel Handwritten Word Spotting Technique Based on Connected Component Analysis. In: 20th International Conference on Pattern Recognition (ICPR), Istanbul Turkey, pp.2013-2016 (August 2010)
2. Sagheer, M., He, C., Nobile, N., Suen, C.: Holistic Urdu Handwritten Word Recognition Using Support Vector Machine. In: 20th International Conference on Pattern Recognition (ICPR), Istanbul Turkey, pp.1900 - 1903 (August 2010)
3. Alamri, A., Sadri, J., Suen, C., Nobile, N.: A Novel Comprehensive Database for Arabic Off-Line Handwriting Recognition., Montreal Canada (August 2008)
4. Shah, M., Sadri, J., Suen, C., Nobile, N.: A New Multipurpose Comprehensive Database for Handwritten Dari Recognition. In: Eleventh International Conference on Frontiers in Handwriting Recognition (ICFHR), Montreal Canada, pp.635-640 (August 2008)
5. Jifroodian Haghighi, P., Nobile, N., He, C., Suen, C.: A New Large-Scale Multi-Purpose Handwritten Farsi Database. In: Proceedings International Conference on Image Analysis and Recognition (ICIAR), Halifax Canada, pp.278-286 (July 2009)
6. Shah, M., He, C., Nobile, N., Suen, C.: A Handwritten Pashto Database With Multi-Aspects for Handwriting Recognition., Dijon France, pp.157-161 (September 2009)
7. Sagheer, M., He, C., Nobile, N., Suen, C.: A New Large Urdu Database for Off-Line Handwriting Recognition. In: Procedings International Conference on Image Analysis and Processing (ICIAP), Salerno Italy, pp.538-546 (September 2009)
8. Freeman, H.: Computer processing of line-drawing images. In: Computing Surveys, pp.57-97 (March 1974)
9. Mahalanobis, P.: On the Generalised Distance in Statistics. In: Proceedings of the National Institute of Sciences of India, p.49–55 (1936)
10. Zhang, T., Suen, C.: A Fast Parallel Algorithm for Thinning Digital Patterns., Communications of the ACM, vol. 27, No. 3, pp.236-239 (March 1984)
11. McLachlan, G.: Discriminant Analysis and Statistical Pattern Recognition. Wiley Interscience (1992)
12. McLachlan, G.: Mahalanobis Distance. (1999)