ABSTRACT

Automatic recognition of Arabic words is a challenging task and its complexity increases as the lexicon grows. In pre-modern documents, the vocabulary is unconstrained; therefore a lexicon-reduction strategy is needed to reduce the recognition computational complexity. This paper proposes a novel lexicon-reduction method for Arabic subwords based on their shapes' topology and geometry. First the subword shape's topological and geometrical information is extracted from its skeleton and encoded into a graph. Then the graph is converted into a topological signature vector (TSV) which preserves the graph structure. The lexicon is reduced based on the TSV distance between the lexicon subwords' shapes and a query shape, by keeping the $i$ nearest subwords. The value of $i$ is selected according to a predetermined lexicon-reduction accuracy. The proposed framework has been tested on a database of pre-modern Arabic subword shapes with promising results.

Categories and Subject Descriptors
H.3.1 [Information Storage and Retrieval]: Content Analysis and Indexing; I.7.5 [Document and text processing]: Document Capture—Document analysis

1. INTRODUCTION

The study of ancient and medieval documents is important in order to know our cultural heritage and to understand the evolution of civilizations. A vast number of pre-modern texts has been scanned as digital images in order to preserve them against aging. Nevertheless, the ability of scholars to work with these images is severely limited because they cannot automatically perform certain tasks, such as a query search, which are essential to their work. It is thus important to provide them with algorithms for automatic transliteration and transcription of scanned images, which would extract the textual content of the image and reproduce it in a computer-editable text file. In this paper we focus on pre-modern Arabic documents. Pre-modern Arabic documents can be written with a variety of calligraphic styles, depending on where and when they were copied. The appearance of the written text changes greatly from one style to another. For example the Kufic style consists of straight lines and angles while the Naskh style is curved and supple (figure 1).

(a) Kufic style
(b) Naskh style

Figure 1: Pre-modern Arabic documents

The Arabic script is cursive and written from right to left. It is composed of an alphabet of 28 letters. One important feature of Arabic letters is that their shapes are context depen-
dent. More precisely, the shape of a letter is usually different if it is in an initial, medial or final position in a word. The letters have no cases and many letters share the same base-shape. They are distinguished by the addition of diacritical marks. The diacritics used in Arabic for this purpose are one, two or three dots appearing below or above the base-shape. If we ignore the dots, we obtain the archigraphemes (figure 2), where a single grapheme (letter shape) can represent many letters. Four archigrapheme letter shapes (‘A’, ‘D’, ‘R’, ‘W’) can be connected only if they are in final position. If they appear in the middle of a word, the word is divided into subwords, also known as pieces of Arabic word (PAW). Despite the fact that the diacritical marks, and especially the dots, are important cues to discriminate between different letters, in pre-modern Arabic script the dots, when they are included at all, tend to float around the subword shape and their location is often determined by aesthetic considerations rather than by their immediate proximity to the corresponding letter. This feature is therefore unreliable in pre-modern Arabic documents and must be ignored at a first stage in order to recognize the correct archigraphemes.

Figure 2: Arabic transliteration table. If a transliteration is defined in brackets, it is used when the letter appears in a non-final position of a subword composed of more than one letter

In pre-modern documents the vocabulary is unconstrained, and the number of different classes can be as large as 30000 for certain corpora, with very unbalanced distribution between the classes. Also the character segmentation of Arabic subword is a very challenging task. Therefore a holistic word-recognition strategy, in which a query word shape is matched against the word shapes of the lexicon, is suited for unconstrained Arabic documents. Nevertheless a large lexicon induces a high computational complexity as well as a decrease of the recognition performance for such approaches. Lexicon-reduction methods are used to tackle these problems. When a query word shape is submitted for recognition, the lexicon is pruned by keeping only the word shapes that are most likely to correspond to the query word class. In a second stage the chosen recognition algorithm considers only the word shapes of the pruned lexicon. The performance of a lexicon-reduction method is classically evaluated from its reduction degree (the decrease of the size of the lexicon after pruning), the reduction accuracy (the probability that the query word class was included in the pruned lexicon) and the reduction efficiency, which is a combination of the two previous criteria. Computational complexity is also a major factor as one of the goals of the process is to speed up the recognition.

The goal of this paper is to provide an efficient lexicon-reduction strategy for Arabic documents. In this work, the lexicon is composed of a vocabulary of naked subwords (Arabic subwords written with archigraphemes). This step already forms a lexicon reduction compared to classical approaches, since the number of different subwords is smaller than the number of Arabic words and that many subwords differentiated only by diacritical marks correspond to the same naked subword. The recovery of the correct subword from a naked subword can be done in a post-processing stage by considering the neighboring diacritical marks. The main contribution of this paper is to provide a lexicon-reduction method based on the structure of the Arabic (naked) subword shapes, which is described by their topology and geometry. In a first step, topological and geometrical properties of the subword shapes will be extracted from the shape skeleton, then they will be encoded into a directed acyclic graph (DAG) in order to keep information about their relations in the skeleton. Then the subword DAG will be transformed into a vector using the topological signature vector (TSV) [17]. The TSV is a powerful encoding for structured data such as a DAG as it maps the DAG into a low-dimensional vector space for fast matching but at the same time, it preserves to some extent the topological property of the DAG; this provides the TSV with a good discriminative power. The lexicon reduction is performed by selecting the $i$ nearest subwords of the lexicon to a query subword in the TSV space. This is done in two steps: first, the lexicon is indexed by ordering its subwords in ascending order based on their distance to the query subword; next, the lexicon is reduced by selecting the first $i$ elements of the indexed lexicon as candidates. The value of $i$ is evaluated during a training phase in order to reach the reduction accuracy level selected for the application. The same value $i$ is then applied for all the query shapes during the lexicon reduction (figure 3).

Figure 3: Lexicon reduction based on topological signature vector (TSV)

The organization of this paper is as follows. First, related work on lexicon reduction and Arabic word recognition will be reviewed. Then the details of the formation of the Arabic subwords DAG will be provided, followed by details on the TSV scheme. Then the classifier used to validate our approach will be described. Finally, experimental details and results of this lexicon-reduction strategy will be given, followed by the conclusion.

2. RELATED WORK

Lexicon reduction can be performed by comparing the lexicon words' optical shapes or by using application-dependent knowledge. When the word’s optical shape is used, the simplest but still efficient criterion for lexicon reduction is the word length [12], as it allows an easy discrimination between small and long words. More refined knowledge about the
The recognition of handwritten Arabic words can be either analytical, in which each character of the word is recognized individually, or holistic, in which the word is recognized as a whole. Recent analytical Arabic word-recognition systems are based on statistical classifiers. Al-Hajj et al. [2] proposed an optical character recognition (OCR) method based on HMM. The HMM model implicitly performs character segmentation. A sliding window scans the subword from right to left. On each position of the window, a set of manually designed statistical and structural features is extracted, based on the upper and lower baselines of the subword. Graves et al. [10] proposed an artificial neural-network classifier. The network takes as input directly the word image and extracts features in a manner similar to a convolutional neural-network, where the input of each layer is convolved with a local linear filter and subsampled. At the last layer the image is transformed into a 1D signal along the horizontal direction and a specialized network is used to transcribe the signal into a sequence of characters. These methods have been evaluated on constrained vocabulary databases such as the IFN/ENIT database. The analytical methods rely heavily on the knowledge of the lexicon to correct OCR errors and they remain to be tested on unconstrained pre-modern Arabic documents. Holistic word-recognition systems avoid the difficult step of character segmentation of Arabic word. A set of topological and geometrical features is extracted from the whole word shape. If the number of word classes is limited and a sufficient number of samples for each class is available, a statistical classifier can be used for the recognition [4, 16]. For a large and unconstrained lexicon, matching techniques are used for the recognition, which scales the matching complexity with the size of the lexicon.

### 3. ARABIC SUBWORD GRAPH REPRESENTATION

In this section, a method to encode the structure of Arabic subword shapes into a directed acyclic graph (DAG) will be presented. A DAG can be defined as a graph formed by a collection of vertices and directed edges, where the edges connect the vertices, in such a way that no alternating sequence of vertices and edges can reach the same vertex twice. Here the root of a DAG is defined as the vertex with the oldest formation time, so it has no incoming edge. The DAG representation is chosen for its richer expressiveness than the vector representation, thanks to the relational information it contains. The saliency of Arabic subwords comes from their topology and geometry. The shape skeleton highlights these features of the subword shape. Therefore relevant pieces of information will be extracted from the shape skeleton and represented as a non-labeled DAG. First, we can distinguish three types of points on a skeleton: the end points which only have 1 neighbor, curve points which have 2 neighbors and branch points which have 3 neighbors or more. Neighboring curve points can be grouped together and considered as skeletal curves. The end points and branch points are informative about the topology of the shape, while the skeletal curves are informative about its geometry; this is true since the skeleton approximates the loci of the center of the pen during the writing of the subword. The end points and branch points can be set as vertices of the subword graph; they will encode the topology of the skeleton. The geometry of the skeletal curve is given by its curvature:

\[ \kappa = \frac{x'y'' - y'x''}{(x'^2 + y'^2)^{3/2}} \]  

(1)

The most salient parts of a curve are given by the curvature extrema and inflection points. Nevertheless it is very hard to use the definition of Eq. (1) on digital curves to find the curvature extrema as the curves are usually not smooth. Smoothing the curve could suppress extrema; therefore it is not an optimal solution. The curvature at point \( t \) of a curve \( C \) is instead estimated using the angle \( \theta \) between two vectors \( \hat{a} \) and \( \hat{b} \) (figure 4):

\[ \hat{a}(t) = C(t + k) - C(t) \]

\[ \hat{b}(t) = C(t - k) - C(t) \]

\[ \theta = \arccos \left( \hat{a} \cdot \hat{b} / (|\hat{a}| \cdot |\hat{b}|) \right) \]  

(2)

where \( k \) is a parameter that defines the ‘support’ of a given point. The angle can be easily thresholded to select high curvature points and the non-maxima can be removed. Once
the curvature extrema are obtained, an inflection point is inserted between two consecutive extrema if their curvature signs are different. The curvature extrema and the inflection points of the skeletal curve can then be added as additional vertices of the DAG. The other geometric information contained in the skeletal curve is its length. The curve length is a quantitative measure but it needs to be transformed into a set of vertices to fit in a non-labeled DAG. For this purpose a reference length \( l \) is defined. If the quotient of the curve length by \( l \) is \( q \) (with \( q > 0 \)), the curve is split into \( q \) curve segments of equal length, and a ‘length point’ is set at the separation between the curve segments. A vertex is added to the DAG \( G \) for each length point.

The relational properties between the graph vertices are given by the graph edges. The edges of the graph will represent the skeletal curves, linking two vertices if they were connected by a skeletal curve in the skeleton image. Up to now we have obtained an undirected graph. In order to transform it into a DAG, we need a partial order over the graph vertices. It will be done by assigning to each vertex a formation time equal to its distance to the nearest end point of the skeleton. To obtain a distance, the edges will be weighted by the length of their skeletal curves. The distance between two vertices is then defined as the weight of the shortest paths between the vertices, i.e., the sum of the weights of the edges traversed by the shortest path. The distance of each vertex to the end points can be obtained using the Dijkstra algorithm, as this task corresponds to a single-source shortest-path problem on a graph. The following partial ordering is used on the graph vertices:

\[
    u \leq v : d_u \geq d_v, \quad (3)
\]

where \( u \) and \( v \) are vertices of the graph and \( d_u \) and \( d_v \) are their respective nearest distances to an end point. A path of directed edges between \( u \) and \( v \) exist only if the partial ordering \( u \leq v \) is respected. This ordering puts the vertices corresponding to the skeleton end points as leaves of the graph, as their nearest end point is themselves and thus the distance is zero. The goal of this ordering is to put the central part of the subword as the root. The root can be any type of vertex, even an end point if the graph only contains end points. The process of formation of the subword graph from a subword shape is illustrated in figure 5. First the shape skeleton is computed. Then the graph vertices are identified on the skeleton. In this example we identify two end points, two curvature extrema and one inflection point; there is no branch point or length point on this shape. Then the vertices are organized into a DAG with the previously defined ordering. It can be noted that the inflection point which is in the ‘middle’ of the skeleton becomes the DAG’s root.

**Algorithm 1** Computation of the Arabic subword DAG

**Input:** subword image; \( l \), the length of unit curve segment  
**Output:** \( G \), the subword DAG  
Compute the subword skeleton  
Detect the branch points, end points and curve points  
for all skeletal curves do  
    Detect curvature extrema and inflection points  
    Split the curve around the detected points  
end for  
for all skeletal curves do  
    Compute \( q \), the quotient of \( l \) by the length of the curve  
    Set \( q - 1 \) length points to split the curve into \( q \) curve of equal length  
end for  
Set a vertex in the subword graph \( G \) for each branch point, end point, curvature extremum, inflection point and length point  
Set an edge between two vertices if they are connected in the skeleton image  
Compute distance of each vertex to its nearest end point using the Dijkstra algorithm  
Transform \( G \) into a DAG using the partial ordering of Eq. (3)

4. **TOPOLOGICAL SIGNATURE VECTOR**

The topological signature vector is an efficient encoding of the topology of structured data such as a directed acyclic graph. The topology of a given DAG \( G \) can be represented by its adjacency matrix \( A \), where \( A(i,j) = 1 \) if an edge goes from the vertex \( v_i \) to the vertex \( v_j \), \( A(i,j) = -1 \) if an edge goes from the vertex \( v_j \) to the vertex \( v_i \), and \( A(i,j) = 0 \) in all other cases. The adjacency matrix is therefore antisymmetric. From the adjacency matrix, a signature \( S_G \) for the graph \( G \) can be extracted as the sum of the magnitude of
its \( m \) eigenvalues:
\[
S_G = |\lambda_1| + \ldots + |\lambda_m|
\] (4)

In order to enrich the signature representation of the graph, such a signature is extracted for all the subgraphs of the root \( V \) of the DAG. If \( V \) has a degree \( n \), the \( n \) signatures of its subgraphs and the graph signature are sorted by descending order and concatenated to form the TSV:
\[
\chi(G) = [ S_G \ S_{G_1} \ldots \ S_{G_n} ]^T
\] (5)

The largest signature corresponds to the DAG with the richer topology. Therefore the signature of the graph root \( S_G \) will always be greater than the signature of the subgraphs of the root, and will always be the first dimension of the TSV. As the degree of the root of the DAG changes from one graph to another, the size of the TSV is set in advance to a given value \( p \). If the size of the TSV of \( G \) is smaller than \( p \), then the TSV vector is padded with 0 to reach \( p \); if the size of the TSV is larger than \( p \), then the TSV is truncated.

The truncation removes the less informative signatures so it is safe to remove them if needed. The value of \( p \) can be set according to the maximum degree of the root of the DAGs of the database or according to a chosen complexity for the indexing process. An illustration of the formation of the TSV of a DAG \( G \) rooted at \( V \) is presented in figure 6. The root \( V \) has two subgraphs \( G_a \) and \( G_d \), thus the topological signature is computed for \( G \), \( G_a \) and \( G_d \). Their signatures are sorted in decreasing order to form the TSV \( \chi(G) \) of size \( p = 5 \) with the appropriate padding by 0. The adjacency matrix of \( G_a \) is also shown.

\[
G \quad a \quad v \quad c
\]

\[
\chi(G) = [ S_G \ S_{G_a} \ S_{G_d} \ 0 \ 0 ]^T
\]

\[
S_G = 6.82 \geq S_{G_a} = 3.46 \geq S_{G_d} = 2.82
\]

The TSV has many properties thanks to which it is well suited for the indexing of DAG databases. First, it is invariant to consistent reordering of the graph branches. Such reordering does not affect the graph topology but it leads to a different adjacency matrix. Indeed, the branch reordering is equivalent to a permutation of the adjacency matrix. As the eigenvalues of an antisymmetric matrix are invariant to any orthonormal transformation, such as a permutation, the TSV is also invariant. It has also been shown that the TSV is robust to minor perturbation of the graph structure [17]. More precisely, the error between the eigenvalues of an adjacency matrix and its perturbed version is bounded by the largest eigenvalue of the perturbation matrix. This property is very useful as natural data are often noisy and it is difficult to avoid minor perturbations such as vertex splits or merges in practice. The last but not the least property of the TSV is to map structured data into a low-dimensional vector space. The matching of structured data such as DAG has polynomial complexity while the matching of vectors has linear complexity on the dimension of the vector space. Therefore the TSV achieves a great decrease of complexity and makes the indexing of a DAG database possible, such as our lexicon of Arabic subwords represented by DAGs.

5. SUBWORD SHAPE CLASSIFIER

The subword shape classifier is based on a nearest-neighbor strategy (1-NN). A contour-based representation is chosen since it is complementary to the skeleton representation already used for the lexicon reduction. The subword contour is represented using the square-root velocity (SRV) representation [11, 18]. In the SRV representation, the contour is considered as a simple (non self-intersecting) closed curve. The curve is defined on the \( \mathbb{R}^2 \) Hilbert space and has value in the \( \mathbb{R}^2 \) Euclidean space. The SRV representation forms a unit hypersphere in \( \mathbb{L}^2 \), which allows shape-matching while being invariant to translation, and scaling of the contour curves. First, the curve \( f \) is normalized to be of unit length in order to remove the effect of the scale. The curve is then represented using the SRV representation:

\[
q(t) = f(t) / \sqrt{\|f(t)\|}
\] (6)

The geodesic distance between two curves \( q_1 \) and \( q_2 \) is defined as \( d(q_1, q_2) = \arccos (\langle q_1, q_2 \rangle) \). The best curve alignment is found by dynamic programming in order to decrease the influence of the handwriting variability on the recognition.

6. EXPERIMENTS AND RESULTS

Lexicon-reduction methods are evaluated based on the reduction degree and the reduction accuracy. A good method will obtain a high score for these two criteria, despite the trade-off between the reduction of the lexicon and the recognition performance. In this work the number of candidates \( i \) selected from the indexed lexicon is related to the reduction degree, and its value is set so that an application-dependent reduction accuracy is achieved. The average rank of the first correct entry of the indexed lexicon (i.e., the first index which has the same label as the query shape) will also be used as an indicator of performance.

This approach has been evaluated on the Ibn Sina database [8] of Arabic subword shapes. It is based on a commentary on an important philosophical work by the famous Persian scholar Ibn Sina. This database is made of 60 pages and approximately 25k Arabic subword shapes written in Nashi style (figure 1(b)). Each page contains approximately 500
subword shapes. There are 1200 different classes but the distribution of the database is highly unbalanced; some classes have up to 5000 entries while some others have less than 5 entries. The first 50 pages of the database are used as the validation set and the remaining 10 pages are used as the test set.

The skeletal graph of the subword shapes have been obtained using the divergence-ordered thinning algorithm [5], with the threshold parameter to discard irrelevant skeletal branches set to -20. The holes of the shape have been previously filled. The fork points of the graph are merged into a single point once the graph is extracted. For the detection of high curvature points, the support parameter $k$ is set to 20 and points whose vectors’ angle is below 125° are considered as curvature extrema. A more robust approach based on the search for the best support parameter can be used [19]. The size of the TSV has been set to $p = 5$ and the magnitudes of the eigenvalues of the adjacency matrix of a graph are obtained by singular value decomposition (SVD). We consider as the root of the DAG the vertex which has the oldest formation time, i.e., the largest distance. For simplicity, the curve segment length is computed using the $L_\infty$ metric; in other words the length of the segment is equal to its number of pixels.

Some examples of subword topological graphs and their TSV are shown in figure 7. For each shape, the skeleton image is labeled by the graph-vertices index. The corresponding subword graph is shown as well. The vertices of the graph are labeled by two numbers. The first one represents the index of the vertex in the graph and the skeleton image. The second number after the colon represents the point type. The meaning of its value and its correspondence with the color of the vertices on the skeleton is as follows. If the value is lower than 10 it represents the number of neighbors of the vertex point in the skeleton image, i.e., 1 represents an end point (red), 3 and above a branch point (yellow). The value 10 represents a curvature extrema (blue), the value 11 an inflection point (green) and the value 12 a length point (purple). The value of the reference length has been set to $l = 10$ for these figures. It can be noticed that the subword graph and the TSV of all the shapes are quite different.

The candidates-set size $i$ needed to achieve a given reduction accuracy for different values of $l$ is found by cross-validation over the validation set. Each page alternatively becomes a query page, and the lexicon is formed by the remaining 49 pages. The indexing of the lexicon for a given query shape is based on the worst-case indexing. For example, if the query shape label is ‘A’ and two lexicon shapes with labels ‘D’ and ‘A’ have the same TSV distance from the query shape, the lexicon shape ‘A’ will be ranked after the shape ‘D’. The following values for $l$ have been tested: $\{1,5,10,15,20\}$. The best value is selected as the one with the lowest value for $i$ for a reduction accuracy of 95% on the validation set after the cross validation. If the label of a query shape is not found in the lexicon, the query shape is ignored for the computation of the reduction accuracy. The test set is then indexed against the whole validation set as lexicon, in order to evaluate the value of $i$ needed to achieve the reduction accuracy.

The candidates-set size $i$ needed to achieve a given reduction accuracy for different values of $l$ is shown in figure 8. It can be seen that the trend of the graph is the same for the validation database and the test database. We can also notice that the curve that is overall lower than the other curves (and thus performs a better lexicon reduction) is the one for $l = 1$, closely followed by the curve for $l = 5$. In table 1 the average first correct index and the candidate set size $i$ needed to achieve a reduction accuracy of 95% are shown for different values of $l$. The lowest $i$ for the 95% reduction
Table 1: Average first correct index and candidates set size $i$ for a 95% reduction accuracy, for different reference length $l$

<table>
<thead>
<tr>
<th>$l$</th>
<th>Validation set</th>
<th>Test set</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>avg. $i$ for 95%</td>
<td>avg. $i$ for 95%</td>
</tr>
<tr>
<td>1</td>
<td>464</td>
<td>2361</td>
</tr>
<tr>
<td>5</td>
<td>560</td>
<td>2693</td>
</tr>
<tr>
<td>10</td>
<td>954</td>
<td>4275</td>
</tr>
<tr>
<td>15</td>
<td>990</td>
<td>3902</td>
</tr>
<tr>
<td>20</td>
<td>1092</td>
<td>4467</td>
</tr>
</tbody>
</table>

Table 2: Impact of the lexicon reduction on the subword shape classifier

<table>
<thead>
<tr>
<th>Subword shape classifier recognition rate</th>
<th>full lexicon</th>
<th>reduced lexicon</th>
</tr>
</thead>
<tbody>
<tr>
<td>avg.</td>
<td>86.16%</td>
<td>82.20%</td>
</tr>
</tbody>
</table>

7. CONCLUSION

A lexicon-reduction approach based for pre-modern Arabic document has been proposed. The lexicon is formed from Arabic naked subwords, where the distinction given by the diacritical marks is ignored. The lexicon-reduction approach is based on the Arabic subwords’ structure. A DAG representation has been used to encode the topological and geometrical features of the subword shape, which are extracted from the shape skeleton. Then the TSV has been used to map the DAG into a low-dimensional vector space for fast indexing of the lexicon DAGs. Lexicon reduction is achieved by keeping only the first $i$ elements of the indexed lexicon.
Special care has been taken to encode the length information of the skeletal curves by the definition of a reference length, and by testing several values for it. This framework has been tested on the Ibn Sina database of pre-modern Arabic subword shapes. Experimental results are encouraging both for the lexicon reduction performance and for the computational time by achieving a reduction of more than 90% of the database with a 95% reduction accuracy in 30 milliseconds for each query shape. The impact of the lexicon-reduction method on the recognition rate of the subword shape classifier is a decrease of 3.94%. The results suggest that the length information has a great impact on the lexicon reduction performance. In future work, a better encoding of the curve’s length will be investigated, in order to improve the stability of the TSV. Also, the recovery of the subwords’ label from the recognized naked subwords using the diacritical information will be considered. This approach will be extended for pre-modern Chinese documents.

8. ACKNOWLEDGMENTS

The authors would like to thank NSERC of Canada for their financial support.

9. REFERENCES


