W-TSV: weighted topological signature vector for lexicon reduction in handwritten arabic documents

Youssouf Chherawala, Mohamed Cheriet

PII: S0031-3203(12)00108-2
Reference: PR4425

To appear in: Pattern Recognition

Received date: 18 October 2011
Revised date: 1 February 2012
Accepted date: 22 February 2012

Cite this article as: Youssouf Chherawala, Mohamed Cheriet, W-TSV: weighted topological signature vector for lexicon reduction in handwritten arabic documents, Pattern Recognition, doi:10.1016/j.patcog.2012.02.030

This is a PDF file of an unedited manuscript that has been accepted for publication. As a service to our customers we are providing this early version of the manuscript. The manuscript will undergo copyediting, typesetting, and review of the resulting galley proof before it is published in its final citable form. Please note that during the production process errors may be discovered which could affect the content, and all legal disclaimers that apply to the journal pertain.
W-TSV: weighted topological signature vector for lexicon reduction in handwritten Arabic documents

Youssouf Chherawala*, Mohamed Cheriet

Synchromedia Laboratory, École de Technologie Supérieure, 1100 Notre-Dame Ouest, Montreal (QC), Canada

Abstract

This paper proposes a holistic lexicon-reduction method for ancient and modern handwritten Arabic documents. The word shape is represented by the weighted topological signature vector (W-TSV), which encodes graph data into a low-dimensional vector space. Three directed acyclic graph (DAG) representations are proposed for Arabic word shapes, based on topological and geometrical features. Lexicon reduction is achieved by a nearest neighbors search in the W-TSV space. The proposed framework has been tested on the IFN/ENIT and the Ibn Sina databases, achieving respectively a degree of reduction of 83.5% and 92.9% for an accuracy of reduction of 90%.

Keywords:
Lexicon reduction, Arabic handwritten documents, Ancient documents, Weighted topological signature vector (W-TSV), Graph indexing, IFN/ENIT, Ibn Sina database

*Corresponding author

Email addresses: ychherawala@synchromedia.ca (Youssouf Chherawala), mohamed.cheriet@etsmtl.ca (Mohamed Cheriet)

*Tel.: +15143968972; fax: +15143968595.

Preprint submitted to Pattern Recognition March 5, 2012
1. Introduction

Handwritten word recognition systems have improved in a number of ways in recent decades, across many applications, from the recognition of the legal amount on bank checks and of postal addresses [1, 2, 3, 4, 5] to the automated transcription of ancient documents [6, 7, 8, 9, 10]. While the vocabulary for a bank check application is small (fewer than 30 words), it is large for postal applications (1,000 words) and unconstrained for historical documents (several thousand words). A vocabulary of valid words that are expected to be recognized by the system is called a lexicon [11]. A large lexicon generates a high computational complexity, as all the word hypotheses must be tested, and recognition performance decreases as the number of allowed hypotheses grows. To address this problem, lexicon-reduction methods are used. When a query word shape is submitted for recognition, the lexicon is pruned by keeping only the shapes that are most likely to correspond to the query word class [12], or by using application-dependent knowledge [13]. Then, the recognition system considers the word hypotheses remaining in the pruned lexicon. The performance of a lexicon-reduction method is classically evaluated based on its accuracy of reduction $\alpha$ (the probability that the query word class was included in the pruned lexicon), the degree of reduction $\rho$ (the decrease in the size of the lexicon after pruning), and the reduction efficacy $\eta$, which is a combination of the two previous criteria. Computational complexity is also a major factor in lexicon reduction, as one of its goals is to speed up the recognition process. In this paper, we propose a lexicon-reduction method for handwritten Arabic documents, both ancient and modern.
The Arabic language has an alphabet of 28 letters. The script is cursive and written from right to left. One important feature of Arabic letters is that their shapes are context-dependent, which means that a letter shape is usually determined by its position in a word, i.e. initial, medial or final. The letters have no cases and many share the same base shape. They are distinguishable by the addition of diacritical marks. The diacritics used in Arabic for this purpose are dots, one, two, or three of them appearing below or above the base shape. If we ignore the dots, we obtain the archigraphemes (Figure 1), 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 the 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).

![Arabic transliteration table](image)

Figure 1: Arabic transliteration table. If a transliteration is defined in brackets, it is used when the letter is not in the final position in a subword.

The goal of this paper is to provide a lexicon-reduction strategy for Arabic documents, based on the structure of Arabic subword shapes, which is described by their topology and geometry. First, the topological and geometrical properties of the subword shapes are extracted from the shape skeleton.
Then these properties are encoded in a directed acyclic graph (DAG) in order to preserve information about their relationship in the skeleton. Finally, the subword DAG is transformed into a vector using the weighted topological signature vector (W-TSV), which is an extension of the TSV [14] for weighted DAGs. Like the classical TSV, the W-TSV is a powerful tool for encoding structured data, such as a DAG, mapping the DAG to a low-dimensional vector space for fast matching. Also, it has good discriminatory power for DAGs with different topologies, because it preserves their topological properties to some extent. Unlike the TSV, the W-TSV can also discriminate between DAGs sharing the same topology, but with different weights, and it is more robust to topological perturbation than the TSV under small weight perturbation. In this work, lexicon reduction is performed by pruning the reference database of subword/word shapes. This is achieved by selecting the $i$ nearest shapes in the database to a query shape in the W-TSV space. First, the database is indexed by ordering its shapes in ascending order, based on their distance from the query shape; 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 accuracy of reduction level selected for the application. The same $i$ value is then applied for all the query shapes during the lexicon reduction process. From the reduced database of shapes, it is then possible to build a reduced lexicon of subwords/words from the labels of the selected shapes (Figure 2).

This paper is organized as follows. The features of lexicon reduction for ancient and modern Arabic documents are described in section 2. Related work on lexicon reduction is reviewed in section 3. The details of the W-TSV


Figure 2: Lexicon reduction based on the weighted topological signature vector (W-TSV). (a) query shape comparison in the W-TSV space; (b) database indexing based on W-TSV distance; (c) lexicon reduction by selection of the first 3 candidates.

This paper is an extension of the work published in [15]. The underlying methodology, as well as the experimental evaluation, have been significantly improved.

2. Features of ancient and modern Arabic documents for lexicon reduction

The nature of ancient Arabic documents is different from that of the Arabic documents used in modern applications. The study of ancient documents is motivated by their cultural significance, and a vast number of them have been scanned as digital images in order to protect them from aging. Premodern Arabic documents were written during the medieval period. They
can be written in a variety of calligraphic styles, depending on when and where they were copied. The appearance of a 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 3). The diacritics, when they are included at all, tend to float around the subword shape, and their location is more often determined by esthetic considerations than by their immediate proximity to the corresponding letter. This make it difficult to assign the diacritics to the correct subword, especially when the line spacing is reduced. Most of the time, these documents are written by a single author. The lexicon is unconstrained, and the segmentation of Arabic subwords into words is not known a priori.

Arabic word recognition at the subword level is therefore well suited to ancient Arabic documents, as subwords can be easily identified, usually as connected components. In spite of the fact that the diacritical marks, especially the dots, are important cues for discriminating between different letters, this feature is unreliable in these documents for the reasons explained above. They must be ignored in the first stage, so that the correct archigrapheme can be recognized. In this work, the lexicon for ancient documents is composed of a vocabulary of naked subwords (Arabic subwords written with archigraphemes). Lexicon reduction is performed in this step, which it is not in the classical approaches. This is because the number of different subwords is smaller than the number of Arabic words, and also because 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 achieved in a post-processing step by considering the neigh-
boring diacritical marks. A W-TSV is assigned to each subword shape for the lexicon reduction process.

The study of modern Arabic documents is motivated by specific application needs. The recognition system has to deal with a wide variety of writers and the vocabulary is usually large. The segmentation of Arabic text into words can be estimated from the layout of the document, and the diacritics are usually well positioned. Thus, the lexicon for such documents is composed of Arabic words directly, according to the application needs. For
lexicon reduction, a W-TSV is assigned to each connected component of the word image (subwords and diacritics), and these are combined into a single W-TSV for the word shape.

3. Related works

Lexicon reduction can be performed by comparing the optical shapes of the lexicon words to improve recognition speed. When the word’s optical shape is used, the simplest criterion for lexicon reduction, but still efficient, is word length, as this makes it easy to discriminate between long words and short words. More refined knowledge about the word’s shape can also be used. Zimmermann et al. [16] propose the concept of key characters, which are characters that can be accurately identified without a full contextual analysis. Lexicon reduction is performed by considering only the lexicon entries that match the regular expression generated by the key characters. They also estimate the letter count in a word using a neural network for further reduction. A similar approach is proposed by Palla et al. [17], where regular expressions are built from the detection of ascenders and descenders in the query word image. Bertolami et al. [18] propose mapping each character of a word to a shape code. There are fewer shape codes than characters, as they only discriminate between characters based on their ascenders/descenders and basic geometry. The mapping is performed by a hidden Markov model (HMM), which outputs the $n$ best shape-code sequences for a query word. The lexicon is reduced by considering only the words that correspond to one of the shape-code sequences. Kaufmann et al. [19] propose a holistic approach, using the quantified feature vectors as shape descriptors. These
vectors are used by the HMM recognizer, so there is no overhead for the extraction of these features. A model is created for each class of the lexicon, and the word hypotheses are ranked according to the distance between their models and the shape descriptor of the query word. Several other holistic approaches for lexicon reduction extract a string-based descriptor for each shape, which is further matched using dynamic programming, the lexicon entries with the smallest edit distances being considered part of the reduced lexicon. Madhvanath et al. [20] holistic approach is based on using downward pen-strokes descriptors. These pen strokes are extracted from the word shape using a set of heuristic rules, and categorized according to their positions relative to the baseline. Then, lexicon reduction is performed by matching the word descriptors to the ideal descriptors extracted from the lexicon’s ASCII string. Carbonnel et al. [21] compared two lexicon-reduction strategies, one based on lexicon indexing and the other on lexicon clustering. Using ascender/descender-based shape descriptors, the indexing approach showed better performance. Arabic word shapes have a rich structure, with their loops, branches, and diacritics [22, 23, 24]. These structural features have been used for lexicon reduction. Mozaffari et al. [25] propose a two-stage reduction of an Arabic lexicon. In the first stage, the lexicon is reduced based on the number of subwords of the query word. In the second stage, the word’s diacritical mark types and positions are encoded into a string, and the lexicon is reduced based on the string edit distance. Mozaffari et al. [26] extended the previous approach to Farsi handwritten words, which contain more letters than the Arabic alphabet. Wshah et al. [27] propose a similar algorithm, in which the diacritic detection stage is improved by
the use of a convolutional neural network. Farrahi Moghaddam et al. [28] have devised a word-spotting algorithm for pre-modern Arabic documents based on the shape structure of subwords. The first stage of the algorithm consist of lexicon reduction using a self-organizing map (SOM). The SOM is trained using a feature vector of the topological and geometrical properties of the subword skeleton. Once a query shape has been fed to the SOM, only the lexicon of the activated cell and the neighboring cells is considered for further matching. Several lexicon-reduction approaches use application dependent knowledge to improve the system’s recognition rate. For the transcript mapping problem with ancient document images, Tomai et al. [13] propose recognizing each word of a document image by reducing the lexicon to specific lines of the transcript. Morita et al. [29] have taken advantage of the date field structure for the recognition of handwritten dates on bank checks. Milewski et al. [30] use an application-specific lexicon for word recognition on medical forms, while Farooq et al. [31] have proposed estimating the topic of a query document from the output of a word recognizer. As the performance of a word recognizer is very low without a priori knowledge, Farooq et al. used the $n$ best hypotheses for each word, instead of only the first, to infer the document topic. Once the document topic has been found, the query document is submitted again to the word recognizer, but this time with the topic-specific lexicon.
4. Weighted topological signature vector (W-TSV)

4.1. Background

The classical topological signature vector (TSV) is an efficient encoding of
the topology of structured data, such as a directed acyclic graph (DAG). 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 vertex $v_i$ to vertex $v_j$, $A(i, j) = -1$
if an edge goes from vertex $v_j$ to 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:

$$ S_G = |\lambda_1| + \ldots + |\lambda_m| $$  (1)

In order to enrich the signature representation of the graph, such a signature
is extracted from all the subgraphs of $V$, the source of the DAG (vertex with
no incoming edges). 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:

$$ \chi(G) = \left[ S_G, S_{G_1}, \ldots, S_{G_n} \right]^T $$  (2)

The largest signature corresponds to the DAG with the richest topology.
Therefore, the signature of the graph source $S_G$ will always be larger than
the signature of the subgraphs of the source, and will always be the first
dimension of the TSV. As the degree of the source 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, and 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 when needed. The value of $p$ can be set according to the maximum degree of the source 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 $G$ with a source $V$ is presented in Figure 4. The source $V$ has two subgraphs $G_a$ and $G_d$, and so 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.

\[ \chi(G) = \begin{bmatrix} S_G & S_{G_d} & S_{G_a} & 0 & 0 \end{bmatrix} \]

\[ S_G = 8.17 \geq S_{G_d} = 3.46 \geq S_{G_a} = 2.82 \]

Figure 4: Topological signature vector formation for the DAG $G$.

The TSV has many properties that make it well suited to the indexing of DAG databases. First, it is invariant to consistent reordering of the graph branches. Such reordering does not affect the graph’s topology, but it does lead to a different adjacency matrix. In fact, 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. Second, the TSV has been shown to be robust to minor perturbations of the graph structure. 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 (see Section 4.3). 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 substantial decrease in complexity and makes the indexing of a DAG database efficient.

4.2. Generalization to weighted DAG

The TSV only considers the topology of the DAG. Nevertheless, since DAG edges are often weighted, this information can be useful for discrimination. This leads us to propose a new formulation, the weighted TSV (W-TSV), where the weight information is added to the adjacency matrix. One of the main ideas behind the W-TSV is that edges with large weights are more important than edges with small weights. Let $W_G = \{w_{ij}\}$ be the set of edge weights of DAG $G$, such that $w_{ij}$ represents the weight associated with an edge extending from vertex $v_i$ to vertex $v_j$ and $w_{ij} > 0$. The weighted adjacency matrix $A$ of $G$ can be constructed as follows: $A(i, j) = w_{ij}$ for an edge from vertex $v_i$ to vertex $v_j$, $A(i, j) = -w_{ij}$ for an edge from $v_j$ to $v_i$, and $A(i, j) = 0$ otherwise. A weight $w_{ij} = 0$ means that there is no edge between $v_i$ and $v_j$. In the rest of the paper, the weights of $A$ will refer to $W_G$, and $w_{ij}(A)$ will refer to $|A(i, j)|$, i.e. the weight of the edge between $v_i$ and $v_j$, irrespective of its direction. The W-TSV is computed
in a same manner as the TSV, the only difference being that the weighted adjacency matrix is used instead of the classical adjacency matrix. Consider the function $\Gamma : \mathbb{R}^+ \rightarrow \{0, 1\}$:

$$
\Gamma(w) = \begin{cases} 
1 & \text{if } w > 0 \\
0 & \text{otherwise}
\end{cases}
$$

When applied to all the weights of $A$, $\Gamma$ removes the weight information completely, but it preserves the topological property, mapping the weighted adjacency matrix to the classical definition of the adjacency matrix. We note that the classical adjacency matrix is a special case of the weighted adjacency matrix, so from now on we will use the term ‘adjacency matrix’ instead of ‘weighted adjacency matrix’. The discriminative power of the W-TSV over the TSV for weighted DAG is illustrated in Figure 5. The 3 DAGs share the same topology, but they have different edge weights. As a result, their TSV is identical, while their W-TSV is different. The TSV and W-TSV are identical for $G_1$, because all its weights are equal to 1.

![Figure 5: Three DAGs with different weights, but sharing the same topology, and their corresponding W-TSV. They have different W-TSV, but the same TSV (for $G_1$, its TSV and W-TSV are equal).](image)
4.3. Stability and robustness of the W-TSV

The W-TSV uses topological and weight information. In order to be an efficient encoding, it must remain stable and robust under topological and weight perturbations, i.e. the changes in the W-TSV values induced by a perturbation must be commensurate with the perturbation level. In this section, we show the stability of the W-TSV, and its robustness compared to the TSV, under the assumption of small weights perturbation (the notion of ‘small’ is further explained in Proposition 3). The idea here is that noise will be more likely to introduce small weight perturbation than large weight perturbation. The stability of the W-TSV will be studied using graph spectral theory. Consider the graph $G$ and its $m \times m$ adjacency matrix $A$. A lifting operator $\Psi : \mathbb{R}^{+m \times m} \to \mathbb{R}^{+n \times n}$ can be used to create an $n \times n$ adjacency matrix $\Psi(A)$ ($n \geq m$) equivalent to $A$ up to vertex relabeling. This operator will first add $n - m$ zero-valued rows and columns to $A$, forming the matrix $A'$. Then given a permutation matrix $P$, the vertices are relabeled so that $\Psi(A) = PA'P^T$. As $A$ and $A'$ have the same spectrum up to additional 0 elements, and $A'$ and $\Psi(A)$ have the same spectrum, $\Psi()$ is a spectrum preserving operator.

A perturbed graph $H$ can be built from $G$ using the lifting operator and an $n \times n$ perturbation matrix $E$, where $B = \Psi(A) + E$ represents the adjacency matrix of $H$. A weight $w_{ij}(\Psi(A))$ is perturbed by adding (substracting) if $\Psi(A)_{ij}$ and $E_{ij}$ have the same (opposite) sign. We can distinguish three types of perturbation:

- weight perturbation: $w_{ij}(\Psi(A)) > 0$ and $w_{ij}(E) \neq 0$,
- edge addition: $w_{ij}(\Psi(A)) = 0$ and $w_{ij}(E) > 0$, 

15
edge deletion: $w_{ij}(\Psi(A)) = w_{ij}(E) > 0$, and $E_{ij} = -\Psi(A)_{ij}$.

We will assume that $E$ is well conditionned, i.e. $B = \Psi(A) + E$ represents the weighted adjacency matrix of a valid DAG.

We can now show the stability of the W-TSV. Let $\lambda_i(A)$ denote the $i^{th}$ largest element of the set of magnitudes of matrix $A$ eigenvalues. Consider the following result (Shokoufandeh et al. [14]):

**Proposition 1.** If $A$ and $E$ are $n \times n$ antisymmetric matrices, then:

$$|\lambda_i(A + E) - \lambda_i(A)| \leq |\lambda_1(E)|, \text{ for } i \in \{1, \ldots, n\}.$$  

Proposition 1 shows that the eigenvalues of the perturbed matrix $B = \Psi(A) + E$ are bounded by $\lambda_1(E)$. In the case where $E$ represents a topological perturbation matrix (all the weights are equal to 1), $\lambda_1(E)$ is bounded by $\sqrt{k}$, where $k$ is the number of edges of $E$ (Neumaier [32]). We generalize this result to weighted adjacency matrices, as follows:

**Definition 1.** The weight vector $W(E)$ of an adjacency matrix $E$ is the vector formed by concatenation of all the weights of $E$.

**Proposition 2.** If $E$ is an $n \times n$ antisymmetric matrix, then the magnitude of its largest eigenvalue is bounded by the Euclidean norm of its weight vector: $\lambda_1(E) \leq \|W(E)\|$.

**Proof.** $E$ is antisymmetric hence $\lambda_1j$ and $-\lambda_1j$ are eigenvalues of $E$, where $j$ is the imaginary unit. Therefore $2\lambda_1^2 \leq \sum \lambda_i^2 = -\text{tr}(E^2) = \sum_{i,k} w_{ik}^2(E)$. Notice that $\sum_{i,k} w_{ik}^2(E)$ is the sum of all the elements of the matrix $[w_{ik}^2(E)]$ which is symmetric and has the diagonal equal to 0. It can be represented by its upper triangle matrix $U$ such that $[w_{ik}^2(E)] = U + U^T$. The non zero
entries of $U$ are exactly the squared weights of $E$. Therefore the sum of all elements of $U$ is equal to $W(E)^TW(E)$ and $\lambda_1(E) \leq \|W(E)\|$.

Using Proposition 1 and Proposition 2, it is clear that the magnitude of the spectral distortion of a matrix $\Psi(A)$ from a perturbation matrix $E$ is bounded by the magnitude of the weights of $E$. The W-TSV is therefore stable under minor weight perturbation of its corresponding DAG.

We will now show that the W-TSV is more robust to topological perturbation than the TSV, under the assumption of small weight perturbation. For this purpose, the weighted perturbation of $A$ by $E$ will be compared to the equivalent topological perturbation of $\Gamma(A)$ by $\Gamma(E)$. As the TSV is invariant to weight perturbation, we will consider only weighted topological perturbation, represented by a matrix $E$ containing only edge addition and deletion. Nevertheless, the influence of $E$ on the spectrum of $A$ is related to the weights of $A$: it will be larger for small weights of $A$ than for large weights. By contrast, the influence of $\Gamma(E)$ on the spectrum of $\Gamma(A)$ is not related to the weights of $\Gamma(A)$, as all the weights are equal to 1 in the topological case. Therefore $E$ needs to normalized with respect to the weights of $A$, or $\Gamma(A)$ and $\Gamma(E)$ need to be rescaled with the weights of $A$, in order to compare the W-TSV and TSV fairly. We thus introduce the notion of scale for an adjacency matrix, as follows:

**Definition 2.** The scale of an adjacency matrix $A$ is the average value of all its weights.

If the topological perturbation of $\Gamma(A)$ by $\Gamma(E)$ is performed at the same scale as $A$, the effect of the difference in magnitude between $A$ and $\Gamma(A)$ is
removed during the evaluation of their respective spectral distortion. The scale of $\Gamma(A)$ is 1 because all its weights are equal to 1. Let $\mu$ be the scale of $A$, the topological perturbation at this scale is performed by multiplying all the elements of $\Gamma(A)$ and $\Gamma(E)$ by $\mu$. As a result, their respective eigenvalues are also multiplied by $\mu$. The need for the notion of scale becomes obvious if we consider the same topological perturbation but at different scales: $\mu_1 = 1$ and $\mu_2$, such that $\mu_1 \ll \mu_2$; the distortion at scale $\mu_1$ will be lower than at scale $\mu_2$, although they both represent the same topological perturbation. In fact, the rescaling procedure handles such problems. Let $B(A, E)$ denote the upper bound of the magnitude of spectral distortion of $A$ by $E$, then:

**Proposition 3.** $\Psi(A)$ and $E$ are antisymmetric matrices and $E$ represents a weighted topological perturbation. If the root mean square (RMS) of the weights of $E$ is smaller than the scale $\mu$ of $A$, then the upper bound $B(\Psi(A), E)$ is smaller than the upper bound of the equivalent topological perturbation at scale $\mu$: $B(\Psi(A), E) < B(\mu \Gamma(\Psi(A), \mu \Gamma(E))).$

**Proof.** Consider that $E$ represents the weighted addition/deletion of $k$ edges. Then $B(\mu \Gamma(\Psi(A), \mu \Gamma(E))) = \lambda_1(\mu \Gamma(E)) = \mu \sqrt{k}$. Also $B(\Psi(A), E) = ||W(E)|| = \sqrt{k} \alpha$ where $\alpha = ||W(E)|| / \sqrt{k}$ is the RMS of the weights of $E$. Given that $\alpha < \mu$, the result follows. $\square$

From Proposition 3, we can see that if the weights of the perturbation matrix $E$ are small enough compared to the weights of $A$, the W-TSV is more robust than the TSV for topological perturbation at the same scale. An example is shown in Figure 6: in Figure 6(a), the adjacency matrix $A$ of the DAG $G$ is perturbed by $E$, resulting in the DAG $H$ and its adjacency
matrix $B$. In Figure 6(b), $G'$ and $E'$ represent the topological equivalent of $G$ and $E$ at the same scale as $G$. The adjacency matrix $A'$ of the DAG $G'$ is perturbed by $E'$, resulting in the DAG $H'$ and its adjacency matrix $B'$. If we assimilate the distortion of a TSV to the Euclidean distance between the original TSV and its perturbed version, the distortion of the W-TSV (6.59) is smaller than the distortion of the scaled TSV (11.04).

4.4. Proposed fast computation

The topological signature (TS) of a DAG is based on the magnitude of the eigenvalues of its adjacency matrix. For the TSV, the TS is solely based on the structure of the underlying graph, while for the W-TSV it is based on both the weights and structure of the underlying graph. Nevertheless, the computation of the TS involves a singular value decomposition (SVD) of the adjacency matrix, which has a computational complexity of $O(n^3)$ for an $n \times n$ matrix. It is possible to evaluate the TS with a computational complexity of $O(n)$ based only on the weights of the DAG and by ignoring its structure. For fast computation, we simply define the TS of a DAG $G$ as the sum of all its weights:

$$\text{TS} = \sum (w_{ij})$$

The fast computation provides the TS with a better interpretation of its value with respect to its weights. This computation is linear with respect to the weights, so it is stable under minor perturbation by a matrix $E$. As already stated, the cost of the fast computation is the loss of structural information, and the performance of the TSV can be particularly affected by this computation as all its weights are equal. Although the structure of the
Figure 6: Comparison of the perturbation of DAG \( G \) (scale 10) by \( E \) and its topological perturbation at the same scale. (a) Perturbation of \( G \) by \( E \). The W-TSVs of \( G \) and \( H \) are also shown. (b) Perturbation of \( G' \) by \( E' \), the topological equivalent of \( G \) and \( E \) at scale 10. The scaled TSVs of \( G' \) and \( H' \) are shown. The deleted/added edges and vertices are shown respectively in red/green on \( H \) and \( H' \). Here, the distortion of the W-TSV (6.59) is smaller than the distortion of the scaled TSV (11.04).
graph is lost at the TS level, it is retrieved to some extent during construction of the W-TSV.

5. Proposed Arabic subword graph representation

In this section, our holistic method for encoding the structure of Arabic subword shapes into a DAG is presented. We chose the DAG representation because it is more expressive than the vector representation, thanks to the relational information it contains. The saliency of an Arabic subword derives from its topology and its geometry, which are highlighted by the shape skeleton. Therefore, relevant pieces of information are extracted from the shape skeleton, giving rise to 3 DAG representations, each of which integrates more information than the previous one. 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 provide information about the topology of the shape, while the skeletal curves provide information about its geometry. This is because the skeleton approximates the loci of the center of the pen while the subword is being written. A skeletal curve contains information about the geometry of the shape through its length and curvature:

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

The most salient parts of a curve are given by the curvature extrema and inflection points. Once the curvature extrema are obtained, an inflection
point is inserted between two consecutive extrema if their curvature signs
are different.

The first DAG representation is the topological DAG (T-DAG), which
only contains information about the shape topology. The end points and
branch points of the skeletal graph are set as the vertices of the T-DAG,
and each skeletal curve will represent an edge of the DAG, connecting two
vertices if they were connected by the skeletal curve in the skeleton image.
The second DAG representation is the length DAG (L-DAG), which further
integrates information about the skeletal curve lengths, by weighting the
edges of the T-DAG by the length of the corresponding skeletal curve. The
last DAG representation is the curvature DAG (C-DAG), which contains
additional information about the curvature of the skeletal curve. For this,
each skeletal curve is split at the position of the extrema and inflection points.
The curvature extrema and the inflection points of the skeletal curves are
added as additional vertices of the L-DAG, where the weight of the edges is
equal to the length of the split curves.

The three graphical representations defined previously are, in fact, undi-
rected graphs. In order to transform them into DAGs, a partial order is
declared over the graph vertices. This is done by assigning a formation time
to each vertex that is equal to its distance from the nearest end point of
the skeleton. The distance between two vertices is 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. For this transformation, the length of
the corresponding skeletal curves will temporarily be assigned to the T-DAG
edges as weight. The distance from 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:

\[ u \leq v : d_u \geq d_v \]  

(4)

where \( u \) and \( v \) are vertices of the graph and \( d_u \) and \( d_v \) are their shortest distances from an end point respectively. A path of directed edges between \( u \) and \( v \) exists only if the partial ordering \( u \leq v \) is respected. This ordering puts vertices corresponding to the skeleton’s end points as leaves of the graph, because their nearest end point is themselves, and so the distance is zero. The goal of this ordering is to make the central part of the subword the source, which can be any type of vertex, even an end point, if the graph only contains end points. The process of formation of the subword DAGs from a subword shape is illustrated in Figure 7. First, the shape skeleton is computed. Then, the graph’s topological vertices are identified on the skeleton. The shape in the example contains two end points and no branch points. The T-DAG and the L-DAG are extracted from this set of vertices and skeletal curves.

The curvature-based vertices are also identified from the skeletal curve. In this example, we have two curvature extrema and one inflection point. The C-DAG is extracted from this new set of vertices.

6. Experiments

6.1. Databases

We evaluated this approach on the Ibn Sina database [33] for ancient Arabic documents and the IFN/ENIT database [34] for modern Arabic documents. The Ibn Sina database is based on a commentary on an important
philosophical work by the famous Persian scholar Ibn Sina. This database consists of 60 pages and approximately 25,000 Arabic subword shapes written in the Naskh style (Figure 3(b)). The document images were binarized with a dedicated algorithm [35] to preserve the shape’s topology. Each page contains approximately 500 subword shapes. There are 1,200 different classes, but the distribution of the database is highly unbalanced; some classes have up to 5,000 entries, while others have fewer than 5. The diacritics are ignored, and a W-TSV is assigned to each subword shape. The W-TSV size is set to \( p = 3 \) as most of the skeletal points have at most 3 neighbors.

The IFN/ENIT database was built for a postal application, and contains the names of 946 Tunisian towns and villages spread over 26,459 word images. Approximately four hundred writers participated in its creation. For each
connected component of a word shape (subwords and diacritics), a W-TSV of size \( p = 1 \) is computed. Then all these individual W-TSVs are sorted in descending order and concatenated, in order to form the word shape W-TSV (size \( p = 10 \)). The size of the W-TSV is set according to the maximum number of subwords in a word.

6.2. Experimental protocol

The W-TSV is extracted in the following way from a single connected component shape image. First the skeletal graph of the shape is obtained using the divergence ordered thinning algorithm [36], with the threshold parameter, which is used to discard irrelevant skeletal branches, set to -7. In order to prevent the formation of loops in the DAG, the holes of the shape are filled in prior to this step. The fork points of the graph are merged into a single point once the graph is extracted. The curvature extrema points are found using the algorithm described in [37], and the extrema near the ends of the skeletal curve (distance less than 5 pixels) are ignored. If a shape’s DAG contains more than one source, the W-TSV is computed for each source, and all the W-TSVs are added to form the final shape’s W-TSV. For simplicity, the curve length is set to its number of pixels; for an 8-connected curve, it corresponds to the \( L_\infty \) metric. With the 3 DAG representations and the fast and classical computation of the W-TSV, 6 different W-TSVs are evaluated (fast TSV, fast L-TSV, fast C-TSV, TSV, L-TSV, C-TSV).

Some examples from the Ibn Sina database of archigraphemic subwords C-DAG and their fast C-TSV are shown in Figure 8. For each shape, the skeleton image is labeled by the C-DAG graph-vertex index. The vertices of the C-DAG are labeled by two numbers. The first number represents the
index of the vertex in the C-DAG, and the second number after the colon represents the point type. The meaning of the point type value and its corresponding color on the skeleton image is detailed in Table 1. Notice that the C-DAG and the fast C-TSV of the subword shapes are quite different.

<table>
<thead>
<tr>
<th>Vertex type</th>
<th>Value</th>
<th>Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>End point</td>
<td>1</td>
<td>red</td>
</tr>
<tr>
<td>Branch point</td>
<td>3</td>
<td>yellow</td>
</tr>
<tr>
<td>Curvature</td>
<td>10</td>
<td>blue</td>
</tr>
<tr>
<td>Inflection</td>
<td>11</td>
<td>green</td>
</tr>
</tbody>
</table>

The lexicon-reduction method is evaluated on the degree of reduction of the shape database, as well as on the degree of reduction of the lexicon, over the entire query database, achieved for a given accuracy of reduction. A leave-one-out strategy is used for the evaluation; each shape is selected alternately as the query shape and the remaining shapes are considered as constituting the shape database. The results are averaged over the entire database. For the Ibn Sina database, only the first 50 pages are used for this experiment.

6.3. Results and discussion

The lexicon reduction performance on the Ibn Sina and IFN/ENIT databases is shown in Figure 9 and Figure 10. The trend of the curves is the same for both databases, but also for the shape database reduction, as is the case for the degree of reduction of the lexicon. The performances of the W-TSVs,
Figure 8: Arabic archigraphemic subword skeletal graphs, C-DAG (edges weights not shown) and fast C-TSV.
including geometrical information (L-TSV and C-TSV), are very similar, and better than the performances of the pure topological TSVs, as they achieve a higher degree of reduction for a given accuracy of reduction. Detailed results for specific accuracies of reduction are shown in Table 2 and Table 3. On the Ibn Sina database, the best performance is achieved by the fast L-TSV, with a database degree of reduction $\rho = 90.96\%$ and lexicon degree of reduction $\rho = 83.33\%$ for an accuracy of reduction of 95%. On the IFN/ENIT database, the best performance is achieved by the fast C-TSV with a database degree of reduction $\rho = 94.97\%$ and lexicon degree of reduction $\rho = 71.33\%$ for an accuracy of reduction of 95%.

On a 2.30 GHz processor and for fully preprocessed shapes, the lexicon reduction time for each query shape against the lexicon database is approximately 7.5 milliseconds for the Ibn Sina database and 10 milliseconds for the IFN/ENIT database. The preprocessing time on the Ibn Sina database is, on average, 2.5 milliseconds, and 53 milliseconds on the IFN/ENIT database.

The W-TSV approach shows better performance for Arabic documents than the classical TSV, both for database pruning and vocabulary reduction. Indeed, most of the subwords share the same topology, despite having different shapes. Geometrical information is thus needed to improve the discriminative power of the TSV. For W-TSVs including geometrical information, the fast computation show slightly better results than the classical computation, showing the importance of the weights over the structure of the DAG for Arabic word databases. As expected, the fast computation decrease the performances of the TSV. It can be noted that the performance of the L-TSV and C-TSV are very similar. The curvature feature, which
modify the structure of the DAG, doesn’t significantly improve the W-TSV performance once the DAG is weighted by the length of the curves. This further shows the importance of the length feature over the structure of Arabic
Table 2: Lexicon reduction performance on the Ibn Sina database

<table>
<thead>
<tr>
<th>W-TSV type</th>
<th>$\alpha = 90%$</th>
<th>$\alpha = 95%$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Database $\rho$ (%)</td>
<td>Lexicon $\rho$ (%)</td>
</tr>
<tr>
<td>TSV</td>
<td>95.00</td>
<td>86.43</td>
</tr>
<tr>
<td>L-TSV</td>
<td>97.38</td>
<td>91.94</td>
</tr>
<tr>
<td>C-TSV</td>
<td>97.36</td>
<td>91.66</td>
</tr>
<tr>
<td>Fast TSV</td>
<td>94.96</td>
<td>86.05</td>
</tr>
<tr>
<td>Fast L-TSV</td>
<td>97.83</td>
<td>92.94</td>
</tr>
<tr>
<td>Fast C-TSV</td>
<td>97.65</td>
<td>92.47</td>
</tr>
</tbody>
</table>

subwords. The main source of error is the variability in the appearance of the word/subword, either because of the writing variations allowed by the Arabic script style or because of the large panel of writers.

The impact of lexicon reduction on a 1-NN archigraphemic subword shape classifier (Appendix A) has been tested on the Ibn Sina database. The first 50 pages form the shape reference database, and the last 10 pages form the test database. The lexicon was reduced using the fast L-TSV representation, and by keeping the $i$ nearest shapes needed to achieve a given accuracy of reduction (on average, over the cross validation) in the previous experiment. The results, detailed in Table 4, show that the decrease in the recognition rate is in the same order as the decrease in the accuracy of reduction, which means that the decrease in the recognition rate is effectively controlled by the accuracy of reduction.
6.4. Comparison with other methods

The proposed method has been compared to existing approaches for Arabic script. These approaches first reduce the lexicon based on the subword
Table 3: Lexicon reduction performance on the IFN/ENIT database

<table>
<thead>
<tr>
<th>W-TSV type</th>
<th>Accuracy of reduction</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>$\alpha = 90%$</td>
<td>$\alpha = 95%$</td>
</tr>
<tr>
<td></td>
<td>Database $\rho$ (%)</td>
<td>Lexicon $\rho$ (%)</td>
<td>Database $\rho$ (%)</td>
</tr>
<tr>
<td>TSV</td>
<td>96.85</td>
<td>76.45</td>
<td>93.01</td>
</tr>
<tr>
<td>L-TSV</td>
<td>97.63</td>
<td>81.19</td>
<td>94.23</td>
</tr>
<tr>
<td>C-TSV</td>
<td>97.75</td>
<td>81.97</td>
<td>94.46</td>
</tr>
<tr>
<td>Fast TSV</td>
<td>95.39</td>
<td>67.36</td>
<td>90.87</td>
</tr>
<tr>
<td>Fast L-TSV</td>
<td>97.93</td>
<td>83.56</td>
<td>94.89</td>
</tr>
<tr>
<td>Fast C-TSV</td>
<td>98.01</td>
<td>84.03</td>
<td>94.97</td>
</tr>
</tbody>
</table>

counts, and then use a dot descriptor string. On the Ibn Sina database, only the dot descriptor is used, as the recognition is performed at the subword level. First, the dot string matching is evaluated, under the assumption that the dot descriptor extraction from the subword images has an ideal behavior. This experiment is referred to as ideal diacritic matching and will provide upper bound results for the dot-based approaches. Then, a rule based method, similar to that of Mozaffari et al. [25] was used to extract the dot descriptor from the images. Single, double and triple dots are detected and represented by a two-character label representing the number of dots and their positions (up or down) with respect to their base shape. Finally, all the labels are concatenated into a string. The lexicon is reduced based on the string-edit distance from the ideal dot descriptors of the lexicon. The edit cost is 1 for each missing/additional dot, and the value 2 is added to the cost in case of...
Table 4: Impact of lexicon reduction on the archigraphemic subword shape classifier

<table>
<thead>
<tr>
<th>Accuracy of reduction $\alpha$ (%)</th>
<th>Classifier recognition rate (%)</th>
<th>Reduced lexicon size $i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>86.23</td>
<td>20681</td>
</tr>
<tr>
<td>95</td>
<td>84.57</td>
<td>1869</td>
</tr>
<tr>
<td>90</td>
<td>79.07</td>
<td>449</td>
</tr>
</tbody>
</table>

position mismatch. For an ease of comparison, the reduction efficacy measure $\eta = \alpha^k \cdot \rho$ is also used, with $k = 1$, in order to give equal importance to $\alpha$ and $\rho$. The results on the Ibn Sina database are shown in Table 5. The proposed method performs better than the dot-based approach, even for the ideal matching. This result shows the low discriminative power of the dot descriptor at the subword level. This is because most of the subwords have only one diacritical mark, or none at all. On the IFN/ENIT database, the performance of the proposed method is in the range of the other approaches (Table 6). The W-TSV approach uses the subword shape and the subword count in each word, as each connected component represents an element of the word W-TSV. The W-TSV approach is therefore complementary to the dot descriptor approach, and a combination of the two would improve the results. Inspite of the lower performance of the W-TSV than the best method on IFN/ENIT, it has some advantages. First, no a priori knowledge is needed, while the other approaches must perform the identification of subwords and the recognition of diacritics. Also, for lexicon reduction, it has a computational complexity of the order of $O(N)$, where $N$ is the length of the W-TSV vector, while the dot-based approaches have a complexity of the
order of $O(MN)$ due to the string-edit distance, where $M$ and $N$ are the lengths of the strings.

Table 5: Comparison with a dot matching lexicon-reduction method on the Ibn Sina database

<table>
<thead>
<tr>
<th>Method</th>
<th>$\alpha$ (%)</th>
<th>$\rho$ (%)</th>
<th>$\eta$ (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ideal diacritics matching</td>
<td>100</td>
<td>74.96</td>
<td>74.96</td>
</tr>
<tr>
<td>Diacritics matching</td>
<td>75.38</td>
<td>72.88</td>
<td>54.94</td>
</tr>
<tr>
<td>Proposed method (Fast L-TSV)</td>
<td>90.0</td>
<td>92.94</td>
<td>83.64</td>
</tr>
</tbody>
</table>

Table 6: Comparison with other lexicon-reduction methods on the IFN/ENIT database

<table>
<thead>
<tr>
<th>Method</th>
<th>$\alpha$ (%)</th>
<th>$\rho$ (%)</th>
<th>$\eta$ (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subword count and diacritics matching [25]</td>
<td>74</td>
<td>92.5</td>
<td>68.5</td>
</tr>
<tr>
<td>Improved subword count and diacritics matching [27]</td>
<td>94.6</td>
<td>85.6</td>
<td>81.0</td>
</tr>
<tr>
<td>Proposed method (Fast L-TSV)</td>
<td>90.0</td>
<td>83.6</td>
<td>75.2</td>
</tr>
</tbody>
</table>

7. Conclusion

In this paper, we proposed the W-TSV representation, a generalization of the TSV for weighted DAG indexing. The stability and robustness to small weights perturbation of the W-TSV have been studied. The W-TSV has been applied for holistic lexicon reduction of handwritten Arabic words/subwords. The topology and the geometry of the word/subword shape is first converted into a DAG and then transformed into a low dimensional vector using the
W-TSV representation. Three different DAG representations and a fast W-TSV computation approach have been proposed. The W-TSV has shown better performances than the original TSV. This approach is complementary to the dot based lexicon reduction approaches for Arabic documents. The processing speed of this approach can be further improved by parallelizing the thinning algorithm and the nearest neighbors search. In future work, this approach will be extended to other scripts such as Chinese, the main challenge being to properly encode the shape loops into the DAG representation. The proposed DAG representations are invariant to shape rotation, and so directional information will be added to improve performance. Moreover, the combination of the W-TSV representation and other shape representations, such as geometrical moments, will be explored.

8. Acknowledgments

The authors thank the NSERC and SSHRC of Canada for their financial support.

Appendix A. Archigraphemic subword shape classifier

The archigraphemic subword shape classifier is a holistic classifier, based on a nearest-neighbor strategy (1-NN). A contour-based representation is chosen for its complementarity with the skeleton representation used for lexicon reduction. The subword contour is represented using the square root velocity (SRV) representation [38, 39], where the contour is considered as a simple (non self-intersecting) closed curve. The curve is defined on the $\mathbb{L}^2$ Hilbert space, and has value in the $\mathbb{R}^2$ Euclidean space. This representation
allows shape matching, while being invariant to translation and scaling by embedding the contour curve of the shapes on an appropriate manifold. The curve \( f \) is parameterized by \( t \) over the domain \( D = [0, 1] \). First, \( f \) is normalized to unit length, in order to remove the effect of scale. The curve is then represented using the SRV representation:

\[
q(t) = \frac{\dot{f}(t)}{\sqrt{\|\dot{f}(t)\|}}
\]  

(A.1)

This representation is invariant to translation as it uses the derivation of \( f \). It also preserves the unit length constraint on \( f \):

\[
\int_{D} \|q(t)\|^2 dt = \int_{D} \|\dot{f}(t)\| dt = 1
\]  

(A.2)

Therefore, the set of all curves under the SRV representation forms a unit hypersphere in \( L^2 \). Furthermore, the original curve \( f \) can be recovered up to translation from \( q \):

\[
f(t) = \int_{0}^{t} q(s) \|q(s)\| ds
\]  

(A.3)

The geodesic distance between two curves \( q_1 \) and \( q_2 \) is defined as \( d(q_1, q_2) = \text{acos} \left( \langle q_1, q_2 \rangle \right) \). The best curve alignment is sought, in order to decrease the influence of handwriting variability on the recognition process. As the contour curves are closed, the best origin of the curve parameterization is found first, and then the curves are aligned using dynamic programming. After lexicon reduction, only the shapes contained in the reduced lexicon are considered by the 1-NN classifier. Other values of \( k \) have been tested for this \( k \)-NN classifier, but without significant improvement.
References


handwritten document images, in: Proceedings of the Eighth Interna-
tional Workshop on Frontiers in Handwriting Recognition (IWFHR ’02),

[14] A. Shokoufandeh, D. Macrini, S. Dickinson, K. Siddiqi, S. W. Zucker,
Indexing hierarchical structures using graph spectra, IEEE Transactions

[15] Y. Chherawala, R. Wisnovsky, M. Cheriet, TSV-LR: topological signa-
ture vector-based lexicon reduction for fast recognition of pre-modern
Arabic subwords, in: Proceedings of the 2011 Workshop on Historical

[16] M. Zimmermann, J. Mao, Lexicon reduction using key characters in
cursive handwritten words, Pattern Recognition Letters 20 (1999) 1297–
1304.

[17] S. Palla, H. Lei, V. Govindaraju, Signature and lexicon pruning tech-
niques, in: Proceedings of the Ninth International Workshop on Fron-
tiers in Handwriting Recognition, IWFHR ’04, IEEE Computer Society,

[18] R. Bertolami, C. Gutmann, H. Bunke, A. Spitz, Shape code based
lexicon reduction for offline handwritten word recognition, in: Proceed-
ings of the Eighth IAPR International Workshop on Document Analysis


[31] F. Farooq, A. Bhardwaj, V. Govindaraju, Using topic models for OCR


