Pattern Recognition Letters 000 (2018) 1–7

Contents lists available at ScienceDirect

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec

Graph-based keyword spotting in historical manuscripts using Hausdorff edit distance

Mohammad Reza Ameri[a,*], Michael Stauffer[b,c], Kaspar Riesen[b], Tien D. Bui[a], Andreas Fischer[d,e]

[a] Concordia University, Computer Science and Software Engineering Department, 1455 de Maisonneuve Blvd West, Montréal H3G 1M8, Canada
[b] University of Applied Sciences and Arts Northwestern Switzerland, Institute for Information Systems, 4600 Olten, Switzerland
[c] University of Pretoria, Department of Informatics, Pretoria 0083, South Africa
[d] University of Fribourg, Department of Informatics, Fribourg 1700, Switzerland
[e] University of Applied Sciences and Arts Western Switzerland, Institute for Complex Systems, Fribourg 1705, Switzerland

ARTICLE INFO

Article history:
Available online xxx

MSC:
68T10
68T45
05C62
05C85

Keywords:
Keyword spotting
Handwriting graphs
Graph matching
Hausdorff edit distance

ABSTRACT

Keyword spotting enables content-based retrieval of scanned historical manuscripts using search terms, which, in turn, facilitates the indexation in digital libraries. Recent approaches include graph-based representations that capture the complex structure of handwriting. However, the high representational power of graphs comes at the cost of high computational complexity for graph matching. In this article, we investigate the potential of Hausdorff edit distance (HED) for keyword spotting. It is an efficient quadratic-time approximation of the graph edit distance. In a comprehensive experimental evaluation with four types of handwriting graphs and four benchmark datasets (George Washington, Parzival, Botany, and Alvermann Konzilsprotokolle), we demonstrate a strong performance of the proposed HED-based method when compared with the state of the art, both, in terms of precision and speed.

1. Introduction

In recent years we have seen increasing efforts worldwide by libraries and archives to digitize handwritten historical documents. To integrate scanned manuscript images into digital libraries based on their content, automatic handwriting recognition is needed. However, modeling and recognition of handwriting is far more challenging than optical character recognition (OCR) for printed text, mainly due to the variable character shapes. When facing ancient scripts and languages, automatic transcription is often not feasible because of a lack of training data. For such situations, keyword spotting (KWS) offers an alternative to index scanned manuscripts without performing a complete transcription [17].

Two general approaches to keyword spotting can be distinguished, viz. *template-based* and *learning-based* methods.[1] While template-based methods match one or several instances of a keyword image directly with the scanned manuscript, learning-based methods aim to learn word or subword models from labeled training samples. Examples include learning with hidden Markov models (HMM) [7,19,26], support vector machines (SVM) [1], recurrent neural networks (RNN) [12], and convolutional neural networks (CNN) [33,38]. In general, learning-based methods are able to achieve a significantly better performance than template-based methods. However, they are less flexible because they require a considerable amount of labeled training data.

In this article, we focus on template-based methods, which do not require any learning and can be applied even if only a single template image of the keyword is provided to the spotting system. This is particularly useful for historical manuscripts, which typically require human experts for obtaining labeled training data in a time-consuming and costly process.

Early approaches to template-based KWS include pixel-by-pixel matchings of word images [17]. Later on, different feature descriptors have been investigated, including projection profiles [21], histograms of oriented gradients (HOG) [25,27,34], and features extracted from unlabeled data by deep neural networks [37], to name just a few. For coping with the variable width of the handwriting, a widely adopted approach is to use a sliding window for extracting a sequence of feature vectors from word images and match two sequences by means of dynamic time warping (DTW) [21]. To avoid an explicit segmentation of the scanned document page into

* Corresponding author.
E-mail address: mo_amer@encs.concordia.ca (M.R. Ameri).

[1] Another commonly used distinction related to the query type is *query-by-example* and *query-by-string*. See for example [1].

https://doi.org/10.1016/j.patrec.2018.05.003
0167-8655/

word images, segmentation-free methods have been proposed as well [27].

Two general limitations of feature vector descriptors relate to their representational power. Firstly, they have to capture the structure of handwriting with a fixed number of real-valued features regardless of the complexity of the given instance. Secondly, they cannot represent binary relations between parts of the handwriting in a straight-forward way. Both limitations can be overcome by means of graph-based representations which model parts of an object with nodes and relations between the parts with edges [5]. In recent work, several graph-based methods have been proposed in the context of template-based keyword spotting, using keypoints as nodes [14,30,35,36] or basic strokes as nodes [3,22], and connecting them with edges if there is a connection in the image.

The main drawback of graphs, however, is that their high representational power comes at the cost of high computation complexity. Most of the aforementioned methods for graph-based keyword spotting use the well-known bipartite approximation (BP) [24] of the graph edit distance (GED) [4]. Although BP reduces the $\mathcal{NP}$-complete problem of GED to a polyomial-time assignment problem, it still has a cubic time complexity with respect to the graph size, which imposes significant computational constraints for keyword spotting.

In this article, we investigate the potential of a recently introduced more efficient approximation of GED, namely the Hausdorff edit distance (HED) [9]. It has a quadratic time complexity with respect to the graph size – similar to DTW, which has a quadratic time complexity with respect to the sequence length. Unlike DTW, HED is not constrained to sequence matching. Instead it is able to match arbitrary handwriting graphs without constraints as regards the graph structure and the label alphabets for nodes and edges.

A preliminary version of this article has been published as an extended abstract in the proceedings of the 18th International Graphonomics Society Conference (IGS2017) [2]. The present article substantially extends the conference paper with a more detailed description and discussion of the method, a more comprehensive experimental evaluation with three additional benchmark datasets, a study on the combination of HED and DTW, and an extended comparison with the current state of the art. The main focus and contribution of the present work is the graph matching method. For graph-based handwriting representation, we consider four types of handwriting graphs introduced in earlier work [29].

In the remainder, we first describe the four graph-based handwriting representations in Section 2, introduce the HED-based keyword spotting system in Section 3, present our experimental evaluation on four benchmark datasets in Section 4, and conclude the article in Section 5 with an outlook on future lines of research.

2. Handwriting graphs

The proposed template-based keyword spotting approach makes use of graphs for the representation and retrieval of word images as illustrated in Fig. 1. In the first step, handwritten document images are binarized and segmented into word images (detailed in Section 2.1). Graphs are then extracted from single word images by means of four different representations (see Section 2.2). Next, the graphs are normalized by a z-score to minimize intraclass variations (see Section 2.3). Finally, keyword spotting is performed by computing all graph dissimilarities (see Sections 3.1 and 3.2) between a certain query graph q and all document graphs $g \in G$ to build a retrieval index (see Section 3.3).

In the following sections, the first three steps are described in greater detail, while the actual graph-based keyword spotting method is detailed in Section 3.

2.1. Image preprocessing

To remove noise, a *Difference of Gaussians* edge enhancement is first applied to scanned document images [6]. Next, the locally enhanced document images are binarized by means of global thresholding. As the proposed keyword spotting framework operates on isolated word images, document images are segmented into text lines and subsequently into words by means of projection profiles. If necessary, the automatic segmentation result is manually corrected. That is, segmentation errors are neglected in the evaluation and the measured precision can be seen as an upper bound on the end-to-end spotting performance. Image preprocessing also includes a skew correction [18], i.e. a correction of the inclination of the document, which is applied at word-level. Optionally, word images are skeletonized by means of a 3×3 thinning operator [13]. The binarized word images are denoted by B, while skeletonized word images are denoted by S from now on.

2.2. Graph extraction

A graph g is defined as a four-tuple $g = (V, E, \mu, \nu)$ where V and E are finite sets of nodes and edges, and $\mu\colon V \to L_V$ as well as $\nu\colon E \to L_E$ are labeling functions for nodes and edges, respectively. Graphs can be divided into *undirected* and *directed* graphs, where pairs of nodes are either connected by undirected or directed edges. Additionally, graphs are often distinguished into *unlabeled* and *labeled* graphs. In the latter case, both nodes and edges can be labeled with an arbitrary numerical, vectorial, or symbolic label from L_V or L_E, respectively. In the former case we assume empty label alphabets, i.e. $L_V = L_E = \{\}$. In the present work, we consider four types of graph representations that have been introduced by Stauffer et al. [29]. They result in nodes that are labeled with two-dimensional numerical labels, while edges remain unlabeled, i.e. $L_V = \mathbb{R}^2$ and $L_E = \{\}$. Fig. 2 illustrates the handwriting graphs for the manuscripts used in our experimental evaluation (see Section 4.1). In the following, we briefly describe the graph extraction procedures. For a more detailed account, we refer to Stauffer et al. [29].

`Keypoint`. The first graph extraction algorithm makes use of characteristics points (so-called keypoints) in skeletonized word images S. These keypoints are represented as nodes that are labeled with the corresponding (x, y)-coordinates. Between pairs of keypoints (which are connected on the skeleton) further intermediate points are converted to nodes and added to the graph at equidistant intervals. Finally, undirected edges are inserted into the graph for each pair of nodes directly connected by a stroke.

`Grid`. The second graph extraction algorithm is based on a grid-wise segmentation of binarized word images B into equally sized segments. For each segment, a node is inserted into the graph and labeled with the (x, y)-coordinates of its respective center of mass. Undirected edges are inserted between two neighboring segments that are actually represented by a node. Finally, the inserted edges are reduced to the minimal spanning tree.

`Projection`. The next graph extraction algorithm is computed on the horizontal and vertical projection profiles of B. The resulting segmentation is further refined in the horizontal and vertical direction by means of two distance-based thresholds. A node is inserted into the graph for each segment and labeled by the (x, y)-coordinates of the corresponding center of mass. Undirected edges are inserted into the graph for each pair of nodes directly connected by a stroke in the original word image.

`Split`. The fourth graph extraction algorithm is based on an iterative segmentation of binarized word images B. That is, segments are iteratively split into smaller subsegments until the width and height of all segments are below certain thresholds. A node is inserted into the graph and labeled by the (x, y)-coordinates of

Please cite this article as: M.R. Ameri et al., Graph-based keyword spotting in historical manuscripts using Hausdorff edit distance, Pattern Recognition Letters (2018), https://doi.org/10.1016/j.patrec.2018.05.003

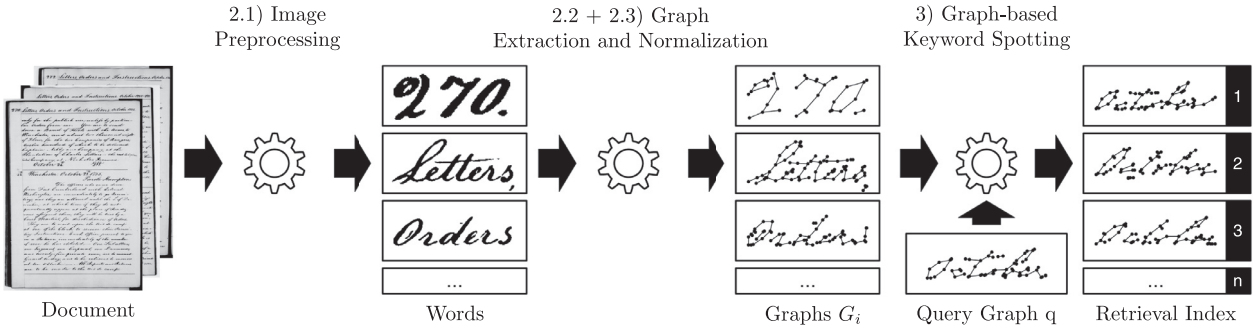

Fig. 1. Process of graph-based keyword spotting of the word "October".

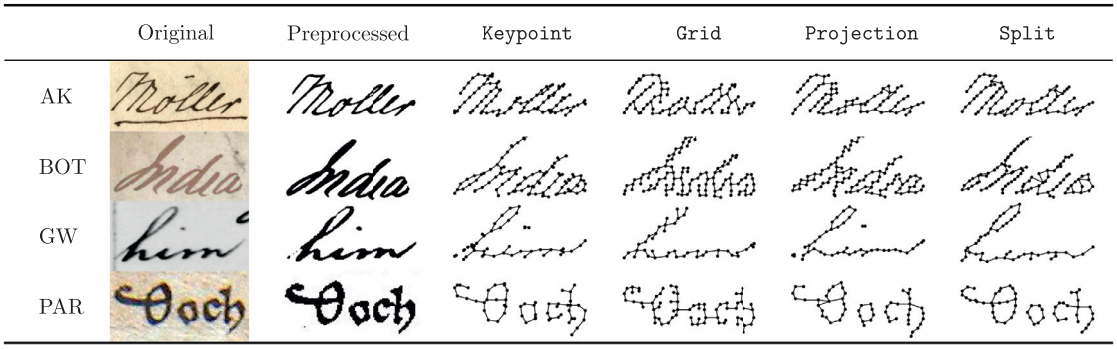

Fig. 2. Exemplary graph representations of the Alvermann Konzilsprotokolle (AK), Botany (BOT), George Washington (GW), and Parzival (PAR) dataset.

the point on the stroke closest to the center of mass of each segment. For the insertion of the edges, the same procedure as for `Projection` is applied.

2.3. Graph normalization

To mitigate the influence of intraclass writing variations, the resulting set of graphs is normalized with respect to the (x, y)-coordinates of their node labels $\mu(v)$. Formally, we use a z-score to derive normalized coordinates $(\hat{x}, \hat{y})$ by

$$\hat{x} = \frac{x - \mu_x}{\sigma_x} \text{ and } \hat{y} = \frac{y - \mu_y}{\sigma_y} \quad ,$$

where (μ_x, μ_y) and (σ_x, σ_y) are the mean and standard deviation of all (x, y)-coordinates in the graph under consideration.

3. Graph-based keyword spotting

For spotting keywords, a query graph q (used to represent a certain keyword) is pairwise matched against all document graphs $G = \{g_1, \ldots, g_N\}$. Generally, graphs can either be matched by means of exact or inexact approaches [5,10]. In the case of graph-based KWS, graphs are used to represent the inherent characteristic of handwriting, and thus, affected by (subtle) variations in both their structure and labels. For this reason, inexact graph matching can be applied only.

3.1. Graph edit distance

Several approaches have been proposed for inexact graph matching [5,10]. Yet, graph edit distance (GED) is regarded as one of the most flexible and powerful paradigms [4,23]. In particular, GED measures the amount of distortion needed to transform graph g_1 into graph g_2 using a sequence of edit operations like *insertions, deletions*, and *substitutions* of both nodes and edges (called *edit path* $\lambda(g_1, g_2)$ between g_1 and g_2).

To find the most suitable edit path, one commonly introduces a certain cost function $c(e)$ for every edit operation e. This cost function should correspond to the strength of a certain graph modification. Formally, the graph edit distance $d_{\text{GED}}(g_1, g_2)$, or d_{GED} for short, between g_1 and g_2 is given by

$$d_{\text{GED}}(g_1, g_2) = \min_{\lambda \in \Upsilon(g_1, g_2)} \sum_{e_i \in \lambda} c(e_i) \quad ,$$

where $\Upsilon(g_1, g_2)$ is the set of all edit paths between g_1 and g_2.

For the representation of domain knowledge, one commonly makes use of a certain cost model. Following the cost model of [30], we use constant costs for both node and edge deletions/insertions, i.e. $\tau_v \in \mathbb{R}^+$ and $\tau_e \in \mathbb{R}^+$. Unlabeled edges are substituted without costs, i.e. $c(p \to q) = 0$. For the substitution of nodes $(u \to v)$, we make use of a weighted Euclidean distance between the corresponding node labels,

$$c(u \to v) = \sqrt{\alpha\, \sigma_x (x_i - x_j)^2 + (1 - \alpha)\, \sigma_y (y_i - y_j)^2} \quad ,$$

where $\alpha \in [0, 1]$ denotes a parameter to weight the importance of the x- and y-coordinate of a node, while σ_x and σ_y denote the standard deviation of all node coordinates in the current query graph. Moreover, we make use of a weighting factor $\beta \in [0, 1]$ between the node and edge edit costs.

3.2. Hausdorff edit distance

The exact computation of d_{GED} is exponential with respect to the number of nodes of the involved graphs. Formally, GED is an instance of a *Quadratic Assignment Problem (QAP)* [15], which in turn belongs to the class of $\mathcal{NP}$-complete problems.[2] Hence, several fast but suboptimal algorithms have been proposed in the last years (see [10]).

[2] That is, an exact and efficient algorithm for the graph edit distance problem can not be developed unless $\mathcal{P} = \mathcal{NP}$.

Please cite this article as: M.R. Ameri et al., Graph-based keyword spotting in historical manuscripts using Hausdorff edit distance, Pattern Recognition Letters (2018), https://doi.org/10.1016/j.patrec.2018.05.003

In this article, we consider the recently introduced Hausdorff edit distance (HED) [9], which is a lower bound of graph edit distance $d_{\text{HED}} \leq d_{\text{GED}}$ that can be computed in quadratic time with respect to the graph size. It reduces the problem of graph edit distance to a set matching problem between local substructures (nodes and their adjacent edges).

The Hausdorff edit distance $d_{HED}(g_1, g_2)$ between two graphs g_1 and g_2 is formally defined as:

$$d_{HED}(g_1, g_2) = \sum_{u \in V_1} \min_{v \in V_2 \cup \{\epsilon\}} f(u, v) + \sum_{v \in V_2} \min_{u \in V_1 \cup \{\epsilon\}} f(u, v).$$

Similar to the Hausdorff distance between finite subsets of a metric space, the two summation terms compute nearest neighbor distances between the node sets according to the node function

$$f(u, v) = \begin{cases} \tau_n + \sum_{i=1}^{|P|} \frac{\tau_e}{2} & \text{for node deletion } (u \to \epsilon) \\ \tau_n + \sum_{i=1}^{|Q|} \frac{\tau_e}{2} & \text{for node insertion } (\epsilon \to v) \\ \frac{c(u \to v) + \frac{d_{HED}(P,Q)}{2}}{2} & \text{for node substitution } (u \to v) \end{cases},$$

where P and Q are the set of edges adjacent to u and v, respectively. Note that only half of the implied edge cost is added to the node cost and only half of the substitution cost is considered in general, to ensure the lower bound property.

The edge cost, which is implied by node substitution, is estimated based on the edge sets P and Q with a similar Hausdorff matching function

$$d_{HED}(P, Q) = \sum_{p \in P} \min_{q \in Q \cup \{\epsilon\}} g(p, q) + \sum_{q \in Q} \min_{p \in P \cup \{\epsilon\}} g(p, q)$$

according to the edge function

$$g(p, q) = \begin{cases} \tau_e & \text{for edge deletion } (p \to \epsilon) \\ \tau_e & \text{for edge insertion } (\epsilon \to q) \\ \frac{c(p \to q)}{2} & \text{for edge substitution } (p \to q) \end{cases}$$

The underestimation of $d_{HED} \leq d_{GED}$ is limited by a minimum edit cost, which is $||V_1| - |V_2|| \cdot \tau_n$ for $d_{HED}(g_1, g_2)$ and $||P| - |Q|| \cdot \tau_e$ for $d_{HED}(P, Q)$. For more details on HED, we refer to Fischer et al. [9].

3.3. Keyword spotting score

For building the KWS score, the approximate graph edit distances d_{HED} between query graph q and all document graphs $G = \{g_1, \ldots, g_N\}$ is normalized by the maximum cost edit path between q and g_i, i.e. the cost of the edit path that results from deleting all nodes and edges of q and inserting all nodes and edges in g_i. Formally,

$$r(q, g) = -\frac{d_{\text{HED}}(q, g_i)}{(|V_q| + |V_{g_i}|)\,\tau_v + (|E_q| + |E_{g_i}|)\,\tau_e},$$

If a query consists of a graph collection $\mathcal{Q} = \{q_1, \ldots, q_t\}$ that represents the same keyword (possibly in different writing styles), the minimal distance is considered

$$r(\mathcal{Q}, g) = \min_{q \in \mathcal{Q}}(r(q, g)).$$

4. Experimental evaluation

We evaluate the proposed HED-based method on four benchmark datasets for keyword spotting in historical manuscripts, which are described in Section 4.1. We compare the performance of the proposed method with three template-based reference methods, namely BP, BP2, and DTW, all of which are detailed in Section 4.2. Finally, we also put our method into context with learning-based approaches to keyword spotting.

4.1. Datasets

For the experimental evaluation we consider two well known manuscripts, viz. *George Washington (GW)*[3] and *Parzival (PAR)*,[4] as well as two documents of a very recent KWS benchmark competition,[5] viz. *Alvermann Konzilsprotokolle (AK)* and *Botany (BOT)*. GW consists of letters of George Washington and his associates during the American Revolutionary War in 1755. The letters are written in English and based on twenty pages with minor variations in writing and degradation. PAR is based on stories of the German poet Wolfgang von Eschenbach in the 13th century. The manuscript is written in Middle High German and based on 45 pages with low writing variations but markable signs of degradation. AK consists of minutes of formal meetings held by the central administration of the University of Greifswald in the period of 1794 to 1797. The notes are written in German and based on 18,000 pages with minor variations and signs of degradation. Finally, BOT is based on botanical records made in British India in the 18th and 19th century. The records are written in English and based on ten pages with high writing variation and markable signs of degradation.

On all four manuscripts, we extract graphs by means of the graph representation formalisms proposed in Section 2. Note that for AK and BOT, only the two most promising graph representations (`Keypoint` and `Projection`) are considered. Fig. 2 shows an exemplary word of each manuscript and the corresponding graph representations.

4.2. Reference methods

In order to assess the potential of the proposed HED-based graph matching approach, we compare it with three related reference methods for matching graphs (BP and BP2) and sequences (DTW), respectively.

BP. The first reference is the bipartite graph matching method (BP) proposed by Riesen and Bunke [24] for approximating the graph edit distance. It is widely used for graph-based pattern recognition (for a survey, see [32]) and has also been considered in a number of graph-based keyword spotting systems, including [3,22,30,31,36]. BP reduces the problem of graph edit distance to a linear sum assignment problem (LSAP) and returns a valid – but not necessarily optimal – edit path between two graphs. The cost of this edit path gives an upper bound of graph edit distance and can be used to compute a spotting score. The main constraint of BP is its cubic time complexity with respect to the graph size, which imposes computational limits regarding the size of the handwriting graphs as well as the number of handwriting graphs that can be matched.

BP2. The second reference is the recently introduced quadratic time variant of BP called BP2 [8]. It solves the bipartite matching problem in quadratic time and returns, similar to BP, a valid edit path between two graphs and thus an upper bound of graph edit distance.

DTW. The third reference is the well-established Dynamic Time Warping (DTW) method for sequence matching, which has often been used for keyword spotting in historical manuscripts [12,21,34,37]. By moving a sliding window over the handwriting a sequence of feature vectors is extracted. DTW finds an optimal alignment of two sequences along a common time

[3] George Washington Papers at the Library of Congress, 1741–1799: Series 2, Letterbook 1, pp. 270–279 & 300–309, http://memory.loc.gov/ammem/gwhtml/gwseries2.html

[4] Parzival at IAM historical document database, http://www.fki.inf.unibe.ch/databases/iam-historical-document-database/parzival-database

[5] Alvermann Konzilsprotokolle and Botany at ICFHR2016 benchmark database, http://www.prhlt.upv.es/contests/icfhr2016-kws/data.html

Please cite this article as: M.R. Ameri et al., Graph-based keyword spotting in historical manuscripts using Hausdorff edit distance, Pattern Recognition Letters (2018), https://doi.org/10.1016/j.patrec.2018.05.003

Table 1
Number of keywords and number of word images in the training and test sets of the four datasets.

Dataset	Keywords	Train	Test
GW	105	2447	1224
PAR	1217	11,468	6869
BOT	150	1684	3380
AK	200	1849	3734

axis such that the sum of feature vector distances is minimal. This sum of distances can then be used to compute a keyword spotting score. Using dynamic programming, an optimal DTW alignment can be obtained in quadratic time with respect to the sequence length. Note that sequences are a special case of graphs, in which the nodes are ordered and have at most one successor.

4.3. Experimental setup

On all benchmark datasets, individual word images are considered for experimental evaluation. The word segmentation is manually corrected, hence the results obtained on these benchmarks can be seen as an upper bound on the spotting performance. In a real-world scenario, errors stemming from automatic word segmentation may decrease the end-to-end performance.

Experiments are conducted in two stages. First, during the validation stage, several system parameters are fine-tuned on a small validation set, which consists of 10 random instances of 10 manually selected keywords (with different word lengths) and 900 additional, randomly selected words (1000 words in total). Secondly, during the testing stage, the optimized system is evaluated on the same training and test sets as used in [7] for GW and PAR and Pratikakis et al. [20] for AK and BOT. All templates of a keyword present in the training set are used for keyword spotting. In Table 1 a summary of the datasets is presented.

To evaluate the keyword spotting performance, we consider Recall and Precision for each keyword query and compute the Mean Average Precision (MAP) over all queries using the `trec_eval`[6] software.

4.4. Comparison with graph edit distance approximations

In the first experiment, we compare HED with other approximation methods of graph edit distance, namely BP and BP2. All three methods can be applied to any type of graph, without constraints on the graph structure or the node and edge label alphabets. The approximate graph edit distance is divided by the maximum graph edit distance to derive a normalized keyword spotting score, as described in Section 3.3.

We consider the four graph-based handwriting representations discussed in Section 2 and adopt optimal graph parameters from previous work [29,30]. Parameters of the keyword spotting system include the cost for node deletion/insertion τ_n, the cost for edge deletion/insertion τ_e, and the weights α, β of the cost function (see Section 3.1). They are optimized over the range of τ_n, $\tau_e \in \{1, 4, 8, 16, 32\}$ and α, $\beta \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$ for each method individually on the validation set.

Table 2 presents the MAP results on the test set of GW and PAR for the three methods and the four graph representations. Confirming the observations in [8], BP2 performs very similar to BP, outperforming BP in five out of eight cases. These results indicate that, in this scenario, the quadratic-time BP2 method is not only significantly more efficient than the cubic-time BP method but it can also achieve similar performance.

[6] http://trec.nist.gov/trec_eval

Table 2
Mean average precision (MAP) for graph-based KWS systems on the George Washington (GW) and Parzival (PAR) datasets.

	Method	GW		PAR	
		MAP	±	MAP	±
BP	`Keypoint`	66.08		62.04	
	`Grid`	60.02		56.50	
	`Projection`	61.43		66.23	
	`Split`	60.23		59.44	
BP2	`Keypoint`	68.42	+2.33	55.03	-7.01
	`Grid`	62.10	+2.07	57.00	+0.50
	`Projection`	60.83	−0.60	63.35	−2.88
	`Split`	64.24	+4.02	68.69	+9.25
HED	`Keypoint`	69.28	+3.19	69.23	+7.19
	`Grid`	62.78	+2.75	60.74	+4.24
	`Projection`	66.71	+5.28	72.82	+6.59
	`Split`	65.12	+4.89	72.79	+13.35

Table 3
Mean average precision (MAP) for graph-based KWS systems on the Botany (BOT) and Alvermann Konzilsprotokolle (AK) datasets.

	Method	BOT		AK	
		MAP	±	MAP	±
BP2	`Keypoint`	45.06		77.24	
	`Projection`	49.57		76.02	
BP2	`Keypoint`	50.94	+5.88	74.86	−2.38
	`Projection`	50.49	+0.92	75.46	−0.56
HED	`Keypoint`	51.74	+6.68	79.72	+2.48
	`Projection`	51.69	+2.12	81.06	+5.04

Table 4
Median and maximum number of nodes, mean runtime per graph pair in milliseconds for BP and HED, and speedup factor on the George Washington (GW) dataset.

Method	$\lvert V\rvert_{med}$	$\lvert V\rvert_{\max}$	T_{BP}	T_{HED}	Speedup
`Keypoint`	74	366	303.0	3.2	95.3
`Grid`	90	509	707.9	6.1	116.0
`Projection`	74	391	344.1	3.9	88.1
`Split`	80	434	480.2	4.4	108.1

HED achieves the best results, outperforming BP in eight out of eight cases. Hence, it not only allows to reduce the computational complexity but also improves the keyword spotting performance. Unlike BP and BP2, HED allows multiple assignments among substructures in the handwriting graphs. We assume that this property of HED is beneficial in the context of handwriting because it allows a kind of "warping" between characters of different size, similar to DTW (see Section 4.2) but in two dimensions rather than only one.

The results shown in Table 3 for the two other datasets, BOT and AK, confirm the findings. On these datasets, HED outperforms BP in four out of four cases.

Finally, Table 4 reports the speedup that can be achieved with the quadratic-time HED method when compared to the cubic-time BP method. On the GW dataset, the handwriting graphs have a median size between 74 and 90 and a maximum size between 366 and 509. For this graph size, HED-based keyword spotting is about hundred times faster than BP-based keyword spotting.

Note that the BP reference method has also been used in the context of other handwriting graphs, including graphs based on keypoints labeled with their shape context [36], graphs based on graphemes extracted from shape convexities [22], and graphs based on invariants corresponding to prototypical strokes [3]. These systems achieve comparable spotting results on a qualitative

Please cite this article as: M.R. Ameri et al., Graph-based keyword spotting in historical manuscripts using Hausdorff edit distance, Pattern Recognition Letters (2018), https://doi.org/10.1016/j.patrec.2018.05.003

Table 5
Mean average precision (MAP) for graph-based KWS systems in comparison with three template-based reference systems on the George Washington (GW) and Parzival (PAR) dataset. The first, second, and third best systems are indicated by (1), (2), and (3).

Method		GW		PAR		Average	
Reference (Template)	DTW'08	63.39		47.52		55.46	
	DTW'09	64.80		73.49	(1)	69.15	(3)
	DTW'16	68.64	(2)	72.38	(3)	70.51	(2)
Graph (Template)	BP	66.08		66.23		66.16	
	BP2	68.42	(3)	68.69		68.55	
	HED	69.28	(1)	72.82	(2)	71.05	(1)

level[7] and can potentially profit from the proposed HED-based approach as an alternative to BP-based graph matching.

4.5. Comparison with dynamic time warping

Table 5 shows a comparison with the state of the art for template-based keyword spotting using DTW. Three reference methods are considered for the GW and PAR benchmark datasets. DTW'08 [25] and DTW'09 [34] employ SIFT-like gradient features, while DTW'16 [37] is based on convolutional neural network (CNN) features that are extracted from the datasets without supervision (without labeled training data) using deep belief networks. All results are taken from Wicht et al. [37]. For HED, we show the results for the best performing graph representations found in Section 4.4, which in most cases is `Keypoint`.

The results indicate that the template-based keyword spotting methods achieve performance results in the same ballpark. DTW'09 and DTW'16 tend to outperform BP and BP2, while HED achieves the overall best results on these benchmarks.

The strong performance of HED is rather astonishing when comparing the sophisticated CNN features of DTW'16 with the relatively simple coordinate labels used for the handwriting graphs. It underlines the representational power of graphs for capturing relevant structures of the handwriting.

Regarding runtime, HED has a quadratic time complexity with respect to the graph size and DTW has a quadratic time complexity with respect to the sequence length. In our experimental setting, the graph size is typically smaller than the sequence length. On the GW dataset, for example, the median graph size is 74, while the median sequence length is 134. In this scenario, HED also reduces the computational effort when compared with DTW.

4.6. Combination of HED and dynamic time warping

In the next experiment, we investigate the potential of combining HED and DTW. Since the two methods are quite different, one matching two-dimensional graphs and the other matching one-dimensional sequences, they have complementary properties and thus a high potential to support each other in a multiple classifier system (MCS). In such an MCS setting, ideally, one method is able to correct errors of the other method [16].

We have implemented our own DTW reference method, following the general ideas of Rath and Manmatha [21] and using the features proposed by Marti and Bunke [18]. Image preprocessing includes skew and slant correction as well as height and width normalization. Afterwards, a sliding window of one pixel width extracts a sequence of nine geometric features. They are aligned by means of DTW using a Sakoe-Chiba band [28] with a width of Ω percent to speedup the alignment and to exclude unusual warping paths. The parameter Ω is optimized on the validation set over a range of $\Omega \in \{0.20, 0.25, \ldots, 0.70\}$. The resulting cost of the warping path is normalized with the length of the warping path to obtain a keyword spotting score. This DTW system achieves a MAP of 64.00 on GW and 71.74 on PAR, which is comparable with the other reference methods listed in Table 5.

[7] A direct quantitative comparison is not feasible due to different experimental conditions.

Table 6
Mean average precision (MAP) for the combination of DTW and HED on the George Washington (GW) and Parzival (PAR) datasets.

Method		GW		PAR	
Individual	DTW	64.00		71.74	
	HED	69.28		72.82	
Combined	DTW+HED	77.83	+8.55	77.00	+4.18

Table 7
Mean average precision (MAP) for graph-based KWS systems in comparison with three state-of-the-art learning-based reference systems on the Alvermann Konzilsprotokolle (AK) and Botany (BOT) datasets. The first, second, and third best systems are indicated by (1), (2), and (3).

Method		BOT		AK		Average	
Reference (Learning)	CVCDAG	75.77	(2)	77.91		76.84	(2)
	PRG	89.69	(1)	96.05	(1)	92.87	(1)
	QTOB	54.95	(3)	82.15	(2)	68.55	(3)
Graph (Template)	BP	49.57		77.24		63.41	
	BP2	50.94		75.46		63.20	
	HED	51.74		81.06	(3)	66.40	

After normalizing the HED and the DTW scores to zero mean and unit standard deviation, they are combined with a weighted sum $hed + \omega \cdot dtw$. The weight ω is optimized on the validation set over a range of $\omega \in \{0.1, 0.2, \ldots, 2.0\}$.

Table 6 reports the combination result on the GW and PAR test sets. Although DTW has a lower performance than HED, the combination leads to a significant increase in MAP by 8.55% and 4.18%, respectively, emphasizing the complementary properties of the two methods.

4.7. Comparison with learning-based keyword spotting

Our proposed HED method follows the template-based approach to keyword spotting, which has minimum requirements regarding human interaction. Even if only a single template image of the keyword is provided to the system, it can search for it in a collection of scanned documents without requiring a human to annotate part of the collection. The low requirements of template-based keyword spotting are especially useful in the context of historical manuscripts, where obtaining labeled training data often requires human experts and thus becomes time-consuming and costly.

However, if labeled training data can be made available to the system, learning-based approaches can profit from this knowledge and build more robust spotting systems. In Table 7, we compare our proposed template-based method with recent learning-based methods from the ICFHR2016 competition [20], viz. CVCDAG [1], PRG [33], and QTOB [38]. CVCDAG is based on Pyramidal Histogram Of Characters (PHOC) features in conjunction with an SVM, PRG is based on the same features in conjunction with a Convolutional Neural Network (CNN), called PHOCNet, and QTOB is based on another CNN following a triplet network approach.

As expected, the learning-based methods achieve a higher performance in general and especially PRG significantly outperforms the proposed HED-based method. Nevertheless, it is interesting to observe that HED can keep up with the performance of QTOB and outperforms CVCDAG in one out of three cases, despite the fact that no learning has been performed for HED. This observation demonstrates the high potential of HED as a template-based keyword spotting method.

Please cite this article as: M.R. Ameri et al., Graph-based keyword spotting in historical manuscripts using Hausdorff edit distance, Pattern Recognition Letters (2018), https://doi.org/10.1016/j.patrec.2018.05.003

Note that template-based and learning-based methods have complementary properties and can be used together in the digitalization process of historical manuscripts. At the beginning, when no labeled data is available, template-based method can be used to cluster similar words that are then labeled conjointly and efficiently by a human expert. As soon as enough training samples become available, learning-based methods can be trained to perform a more accurate search. Finally, when enough labeled data is available to train robust character models, a full transcription can be attempted together with a word dictionary [11].

5. Conclusion and outlook

The HED-based keyword spotting approach presented in this article has demonstrated several promising properties. First, it approximates the graph edit distance and hence is *flexible* in the sense that it allows to represent handwriting with any type of graph, without constraints on the graph structure or the label alphabets for nodes and edges. Secondly, it can be computed in quadratic time with respect to the graph size and hence is *efficient* for matching large graphs and large numbers of graphs. Thirdly, the experimental evaluation on four benchmark datasets for keyword spotting in historical manuscripts has demonstrated that it is *effective* in terms of mean average precision and compares favorably with other template-based keyword spotting systems.

Unlike dynamic time warping, which considers handwriting as a sequence of feature vectors, HED considers the two-dimensional global structure of the handwriting. The two perspectives are different and complementary, which could be demonstrated by combining the two methods into a multiple classifier system that outperformed the individual methods.

There are several promising lines of future research. First, it would be interesting to investigate other, potentially more abstract graph-based representations of handwriting. Secondly, it may be rewarding to include more information about the global handwriting structure when matching local substructures with HED. Finally, given labeled training data is available, an intriguing open question is how to perform machine learning on graph-based representations and graph matching in order to profit from the labeled data.

Acknowledgments

This work has been supported by the Hasler Foundation (grant no. 14047) Switzerland and the Natural Sciences and Engineering Research Council of Canada (NSERC) (grant no. RGPIN-2016-05467).

References

[1] J. Almazan, A. Gordo, A. Fornes, E. Valveny, Word spotting and recognition with embedded attributes, IEEE Trans. Pattern Anal. Mach. Intell. 36 (12) (2014) 2552–2566.
[2] M. Ameri, M. Stauffer, K. Riesen, T. Bui, A. Fischer, Keyword spotting in historical documents based on handwriting graphs and Hausdorff edit distance, in: International Graphonomics Society Conference, 2017, pp. 105–108.
[3] Q.A. Bui, M. Visani, R. Mullot, Unsupervised word spotting using a graph representation based on invariants, in: International Conference on Document Analysis and Recognition, 2015, pp. 616–620.
[4] H. Bunke, G. Allermann, Inexact graph matching for structural pattern recognition, Pattern Recognit. Lett. 1 (4) (1983) 245–253.
[5] D. Conte, P. Foggia, C. Sansone, M. Vento, Thirty years of graph matching in pattern recognition, Int. J. Pattern Recognit. Artif. Intell. 18 (03) (2004) 265–298.
[6] A. Fischer, E. Indermühle, H. Bunke, G. Viehhauser, M. Stolz, Ground truth creation for handwriting recognition in historical documents, in: International Workshop on Document Analysis Systems, 2010, pp. 3–10.
[7] A. Fischer, A. Keller, V. Frinken, H. Bunke, Lexicon-free handwritten word spotting using character HMMs, PRL 33 (7) (2012) 934–942.
[8] A. Fischer, K. Riesen, H. Bunke, Improved quadratic time approximation of graph edit distance by combining Hausdorff matching and greedy assignment, Pattern Recognit. Lett. 87 (2017) 55–62.
[9] A. Fischer, C.Y. Suen, V. Frinken, K. Riesen, H. Bunke, Approximation of graph edit distance based on Hausdorff matching, PR 48 (2) (2015) 331–343.
[10] P. Foggia, G. Percannella, M. Vento, Graph matching and learning in pattern recognition in the last 10 years, Int. J. Pattern Recognit. Artif. Intell. 28 (01) (2014) 1450001.
[11] V. Frinken, A. Fischer, M. Baumgartner, H. Bunke, Keyword spotting for self-training of BLSTM NN based handwriting recognition systems, in: Pattern Recognition, 47, 2014, pp. 1073–1082.
[12] V. Frinken, A. Fischer, R. Manmatha, H. Bunke, A novel word spotting method based on recurrent neural networks, IEEE Trans. Pattern Anal. Mach. Intell. 34 (2) (2012) 211–224.
[13] Z. Guo, R.W. Hall, Parallel thinning with two-subiteration algorithms, Commun. ACM 32 (3) (1989) 359–373.
[14] N.R. Howe, Part-structured inkball models for one-shot handwritten word spotting, in: Proceedings of the International Conference on Document Analysis and Recognition, ICDAR, 2013, pp. 582–586.
[15] T.C. Koopmans, M. Beckmann, Assignment problems and the location of economic activities, Econometrica 25 (1) (1957) 53.
[16] L.I. Kuncheva, Combining Pattern Classifiers, John Wiley & Sons, Inc., Hoboken, NJ, USA, 2004.
[17] R. Manmatha, Chengfeng Han, E. Riseman, Word spotting: a new approach to indexing handwriting, in: Computer Vision and Pattern Recognition, IEEE, 1996, pp. 631–637.
[18] U.-V. Marti, H. Bunke, Using a statistical language model to improve the performance of an HMM-based cursive handwriting recognition systems, IJPRAI 15 (01) (2001) 65–90.
[19] F. Perronnin, J.A. Rodríguez-Serrano, Fisher kernels for handwritten word-spotting, in: International Conference on Document Analysis and Recognition, 2009, pp. 106–110.
[20] I. Pratikakis, K. Zagoris, B. Gatos, J. Puigcerver, A.H. Toselli, E. Vidal, ICFHR2016 handwritten keyword spotting competition (H-KWS 2016), in: ICFHR, IEEE, 2016, pp. 613–618.
[21] T.M. Rath, R. Manmatha, Word spotting for historical documents, Int. J. Document Anal.Recognit. 9 (2–4) (2007) 139–152.
[22] P. Riba, J. Llados, A. Fornes, Handwritten word spotting by inexact matching of grapheme graphs, in: International Conference on Document Analysis and Recognition, 2015, pp. 781–785.
[23] K. Riesen, Structural Pattern Recognition with Graph Edit Distance, Advances in Computer Vision and Pattern Recognition, Springer International Publishing, Cham, 2015.
[24] K. Riesen, H. Bunke, Approximate graph edit distance computation by means of bipartite graph matching, Image Vision Comput. 27 (7) (2009) 950–959.
[25] J.A. Rodríguez-Serrano, F. Perronnin, Local gradient histogram features for word spotting in unconstrained handwritten documents, in: International Conference on Frontiers in Handwriting Recognition, 2008, pp. 7–12.
[26] L. Rothacker, G.A. Fink, Segmentation-free query-by-string word spotting with Bag-of-Features HMMs, in: International Conference on Document Analysis and Recognition, IEEE, 2015, pp. 661–665.
[27] M. Rusiñol, D. Aldavert, R. Toledo, J. Lladós, Efficient segmentation-free keyword spotting in historical document collections, PR 48 (2) (2015) 545–555.
[28] H. Sakoe, S. Chiba, Dynamic programming algorithm optimization for spoken word recognition, IEEE Trans. Acoust. Speech Signal Process. 26 (1) (1978) 43–49.
[29] M. Stauffer, A. Fischer, K. Riesen, A novel graph database for handwritten word images, in: International Workshop on Structural, Syntactic, and Statistical Pattern Recognition, 2016, pp. 553–563.
[30] M. Stauffer, A. Fischer, K. Riesen, Graph-based keyword spotting in historical handwritten documents, in: International Workshop on Structural, Syntactic, and Statistical Pattern Recognition, 2016, pp. 564–573.
[31] M. Stauffer, A. Fischer, K. Riesen, Ensembles for graph-based keyword spotting in historical handwritten documents, in: International Conference on Document Analysis and Recognition, 2017, pp. 714–720.
[32] M. Stauffer, T. Tschachtli, A. Fischer, K. Riesen, A survey on applications of bipartite graph edit distance, in: Graph-Based Representations in Pattern Recognition, 2017, pp. 242–252.
[33] S. Sudholt, G.A. Fink, PHOCNet: adeep convolutional neural network for word spotting in handwritten documents, in: International Conference on Frontiers in Handwriting Recognition, IEEE, 2016, pp. 277–282.
[34] K. Terasawa, Y. Tanaka, Slit style HOG feature for document image word spotting, in: International Conference on Document Analysis and Recognition, 2009, pp. 116–120.
[35] P. Wang, V. Eglin, C. Garcia, C. Largeron, J. Llados, A. Fornes, A coarse-to-fine word spotting approach for historical handwritten documents based on graph embedding and graph edit distance, in: International Conference on Pattern Recognition, IEEE, 2014, pp. 3074–3079.
[36] P. Wang, V. Eglin, C. Garcia, C. Largeron, J. Llados, A. Fornes, A novel learning-free word spotting approach based on graph representation, in: DAS, 2014, pp. 207–211.
[37] B. Wicht, A. Fischer, J. Hennebert, Deep learning features for handwritten keyword spotting, in: International Conference on Pattern Recognition, 2016, pp. 3423–3428.
[38] T. Wilkinson, A. Brun, Semantic and verbatim word spotting using deep neural networks, in: International Conference on Frontiers in Handwriting Recognition, 2016, pp. 307–312.

Please cite this article as: M.R. Ameri et al., Graph-based keyword spotting in historical manuscripts using Hausdorff edit distance, Pattern Recognition Letters (2018), https://doi.org/10.1016/j.patrec.2018.05.003