Pattern Recognition Letters 000 (2018) 1–10

Contents lists available at ScienceDirect

Pattern Recognition Letters

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

Filters for graph-based keyword spotting in historical handwritten documents

Michael Stauffer[a,d,*], Andreas Fischer[b,c], Kaspar Riesen[a]

[a] University of Applied Sciences and Arts Northwestern Switzerland, Institute for Information Systems, Olten 4600, Switzerland
[b] University of Fribourg, Department of Informatics, Fribourg 1700, Switzerland
[c] University of Applied Sciences and Arts Western Switzerland, Institute for Complex Systems, Fribourg 1705, Switzerland
[d] University of Pretoria, Department of Informatics, Pretoria, South Africa

ARTICLE INFO

Article history:
Available online xxx

MSC:
41A05
41A10
65D05
65D17

Keywords:
Handwritten keyword spotting
Graph representation
Bipartite graph matching
Filter methods
Fast rejection

ABSTRACT

The accessibility to handwritten historical documents is often constrained by the limited feasibility of automatic full transcriptions. Keyword Spotting (KWS), that allows to retrieve arbitrary query words from documents, has been proposed as alternative. In the present paper, we make use of graphs for representing word images. The actual keyword spotting is thus based on matching a query graph with all documents graphs. However, even with relative fast approximation algorithms the shear amount of matchings might limit the practical application of this approach. For this reason we present two novel filters with linear time complexity that allow to substantially reduce the number of graph matchings actually required. In particular, these filters estimate a graph dissimilarity between a query graph and all document graphs based on their node and edge distribution in a polar coordinate system. Eventually, all graphs from the document with distributions that differ to heavily from the query's node/edge distribution are eliminated. In an experimental evaluation on four different historical documents, we show that about 90% of the matchings can be omitted, while the KWS accuracy is not negatively affected.

1. Introduction

In the last decade a vital amount of historical handwritten documents have been made publicly available due to digitisation efforts in private and public institutes. However, the automatic recognition of historical handwriting is often negatively affected by both the degenerative conservation state of scanned manuscripts and variations in the handwriting. Hence, an automatic and full transcription is often not feasible and thus the accessibility — in contrast to the availability — is still a widely unsolved issue.

To bridge this gap between availability and accessibility, *Keyword Spotting (KWS)* as a flexible and more error-tolerant alternative has been proposed [10,22,25,30]. KWS allows to retrieve an arbitrary query word in a certain document without the need of a full transcription. Originally, the concept of KWS has been introduced for speech recordings [31], and was later adapted for printed [1] and handwritten documents [22]. In case of historical handwritten documents the KWS process is inherently *offline* and limited to spatial information only. Generally, offline KWS is regarded as more difficult task when compared to *online* KWS which considers temporal information about the writing process as well. The focus of this paper is on historical documents, and thus, offline KWS can be applied only (referred to as KWS from now on).

KWS approaches are either based on *template-based* or *learning-based* algorithms. In case of template-based KWS, a single instance of a query word image is directly matched against a set of document word images. In the latter case, a trained statistical model is used to detect single characters or words of a given visual or textual query. Roughly speaking, template-based methods result in lower recognition rates when compared to learning-based approaches. Yet, this advantage of learning-based approaches is accompanied by a loss of flexibility and generalisability, which is due to the need of an *a priori* learning of the statistical model. Moreover, the accuracy of learning-based algorithms is crucially depending on the size of the labelled training set. However, the acquisition of labelled training data for historical documents is based on human experts and turns out to be a labour and time-consuming task. Besides, many historical documents consist of few pages only, and thus, the size of labelled training data is often inherently limited. This makes template-based KWS a more feasible approach for historical documents. In particular, as the matching algorithms are independent from both the actual representation formalism and the language of the underlying document.

* Corresponding author.
E-mail address: michael.stauffer@fhnw.ch (M. Stauffer).

https://doi.org/10.1016/j.patrec.2018.03.030
0167-8655/

The first template-based KWS approaches are based on pixel-wise matchings of word images [22]. Likewise, global features (*Gradient, Structural and Concavity* features) as well as *Zones Of Interest*, rather than single pixels, are matched in [50] and [20], respectively. However, these features tend to be negatively affected by noisy word images. Thus, more elaborated and error-tolerant methods for template-based KWS are based on matching sequences of feature vectors rather than single vectors. In many cases these features are used to describe certain characteristics of word images like, for example, projection profiles [25,50], gradients [50], or geometrical characteristics [23], to mention just a few. Yet, also more generic image feature descriptors have become popular in recent years like for example *Histograms of Oriented Gradients* [29,43], *Scale-Invariant Feature Transform* [17], or *Deep Learning* features [48]. Regardless the features actually used, *Dynamic Time Warping (DTW)* is employed as the quasi standard for matching the resulting sequences of feature vectors.

In learning-based KWS similar features as described above are used to train a statistical model. Early and widely used approaches are based on *Hidden Markov Models (HMM)* [8,10,19,30,32,44]. In [8], for instance, *generalised Hidden Markov Models* are trained on character images. More recent HMM approaches are based on feature vectors of word images rather than character-based segmentations which are often error-prone and labour intensive [10,19,30]. However, also other statistical models have been applied like for example *Support Vector Machine (SVM)* [2], or *Recurrent Neural Networks* [13,14] to name just two examples. With the rise of *Convolutional Neural Networks (CNN)*, we recently observe a shift in KWS and related fields from HMMs towards CNNs [21,33,42,45,49].

1.1. Related work

The KWS approach proposed in this paper is template-based and makes use of graphs — rather than feature vectors — for the representation of handwriting. In recent years, graphs gained noticeable interest in many pattern recognition applications [12,27,41]. The increased interest might be due to the introduction of fast approximation algorithms that allow to efficiently measure the dissimilarity of larger and more complex graphs (e.g. [11,28]). Moreover, graphs, in contrast with feature vectors, offer a natural way to represent the inherent topological characteristics of a pattern. Additionally, graphs are capable to adapt both their size and complexity to the underlying pattern. However, only limited attempts can be observed for the representation of handwriting by means of graphs [5,26,37,47]. This is rather surprising, as graphs — in contrast to feature vectors — are well suited to adapt to the high structural variabilities of handwriting. These representational advantages of graphs in conjunction with fast approximation algorithms motivate the method of the present paper.

A first graph-based KWS approach has been proposed in [47], where certain characteristic points (so called *keypoints*) in a word image are represented by nodes, while edges are used to represent strokes between these keypoints. The matching of word graphs is then based on a two step procedure. First, graph dissimilarities are measured between pairs of connected components. Second, an optimal alignment between pairs of connected components is found by means of DTW. The same procedure has been extended by a so-called *coarse-to-fine approach* in [46]. Two similar approaches have been proposed by Bui et al. [5] and Riba et al. [26], where nodes represent prototype strokes (termed *invariants* and *graphemes*, respectively), while edges are used to connect nodes that stem from the same connected component. In all of these approaches graph dissimilarity is measured by means of the Bipartite graph matching algorithm [28].

When compared to these existing approaches, the present graph-based KWS framework distinguishes in many facets. First, word images are represented by single graphs, and thus, no additional alignment between subgraphs of different connected components is necessary. Moreover, the applied graph representations aim to capture the inherent characteristics of word images without the need of a prototype library (as used in [5,26]). Thus, the risk of losing certain characteristics of the handwriting is minimised. Finally, the present paper introduces novel filters for graphs to substantially speed up the KWS procedure (similar in spirit in [46]).

1.2. Contribution

In case of a document (represented by a set of graphs $G = \{g_1, \ldots, g_N\}$) and a number of queries $Q = \{q_1, \ldots, q_n\}$, we need to compute $N \times n$ graph matchings. Even with a fast approximation algorithm for computing graph dissimilarities, this particular setting might limit the general applicability of our method. Thus, we aim to reduce the actual number of graph matchings required by efficiently filtering large parts from G with a low similarity to the current query graph $q \in Q$. This approach is known as *fast rejection* [29,30] and the focus of the present paper. That is, we introduce novel graph filters (with linear time complexity) that substantially reduce the number of graph matchings without negatively affecting the KWS accuracy.

The present paper differs and extends our previous approaches to graph-based KWS in various ways [3,37–40]. In [37], we presented our first graph-based KWS approach, while different combinatorial strategies (so called ensemble methods) are introduced in [38]. In three independent publications we introduced then three strategies to speed up the KWS procedure [3,39,40]. The present article follows this line of research and substantially extends the method for fast rejection presented in [40]. In particular, we present a novel graph dissimilarity measure with linear time complexity that makes use of edges and nodes, rather than nodes only as proposed in [40]. This fast graph dissimilarity measure allows very high filter rates (i.e. a clear speed up when compared to the baseline system can be expected). Moreover, also the empirical evaluation is substantially extended when compared with our preliminary paper. That is, we use two additional datasets of a recent KWS benchmark [24] and thoroughly present and discuss the evaluation of all parameters.

The remainder of this paper is organised as follows. In Section 2, the basic graph-based KWS framework is introduced. The proposed filters to speed up the KWS process are introduced in Section 3, while the actual graph matching is described in Section 4. An experimental evaluation and comparison with several reference systems is given in Section 5. Finally, Section 6 concludes the paper and outlines possible future research activities.

2. Graph-based keyword spotting

The proposed graph-based KWS system includes four basics steps as illustrated in Fig. 1.

First, document word images are preprocessed in order to minimise the influence of noisy and skewed scanning (detailed in Section 2.1). Based on binarised and preprocessed document images, word images are automatically segmented and manually corrected if necessary. Next, single word images are represented with graphs by means of different graph extraction algorithms (see Section 2.2). Rather than matching every query graph $q \in Q$ with all graphs from the set of document graphs G, we apply a specific filtering on G with respect to q based on the spatial distribution of nodes and edges, respectively (see Section 3). On the remaining graphs from G we apply the Bipartite graph matching algorithm [28] in order to compare the dissimilarity from q to the un-

Please cite this article as: M. Stauffer et al., Filters for graph-based keyword spotting in historical handwritten documents, Pattern Recognition Letters (2018), https://doi.org/10.1016/j.patrec.2018.03.030

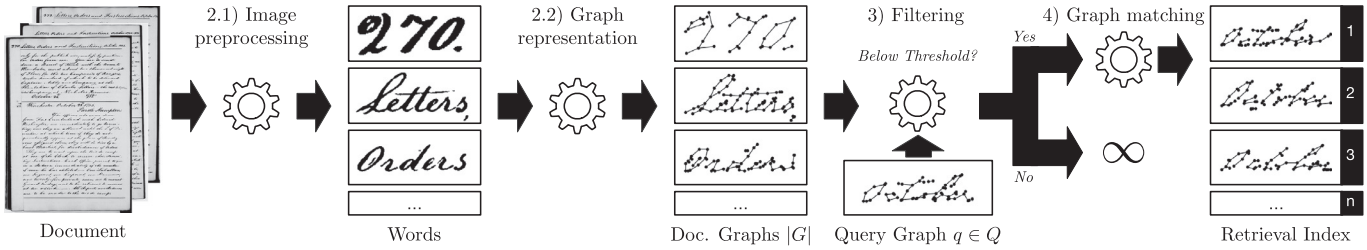


Fig. 1. Process of graph-based keyword spotting of the word "October".

filtered graphs. Finally, the resulting graph dissimilarities are used to form a retrieval index (see Section 4).

The two first steps are described in greater detail in the following subsections, while step 3 and 4 are separately described in Sections 3 and 4, respectively.

2.1. Image preprocessing

Image preprocessing basically aims at reducing undesirable variations which are due to different writers or the digitised document itself (e.g. pixel noise, skewed scanning, or degradation of the document). In our particular case, image preprocessing is focused on document/scanning issues, while variations in the writing style are minimised by graph normalisation (see Eq. (1) in Section 2.2).

The first image preprocessing step addresses the issue of noisy background by means of *Difference of Gaussians* filtering [9]. Subsequently, document images are binarised by a global threshold approach. The present framework focuses on KWS that operates on graphs of perfectly segmented word images. For this reason, document images are first automatically segmented into single word images by means of their projection profiles. Next, the resulting word images are manually inspected and, if necessary, corrected. The present KWS approach neglects any segmentation errors and can therefore be seen as an upper-bound solution. To handle skew, i.e. the inclination of the document, the gradient of the lower baseline of a line of text is estimated and used to deskew single word images [23]. In an optional step, preprocessed word images are skeletonised by means of 3×3 thinning operator [16]. The filtered and binarised word images are denoted by B, while skeletonised word images are denoted by S from now on.[1] In Fig. 4 the influence of the image preprocessing is shown on four exemplary word images stemming from different historical manuscripts.

2.2. Graph representation

Based on binarised and/or skeletonised word images, four different representations are derived [36]. These formalisms aim at extracting the inherent topological characteristics of word images by means of graphs.

In general, 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 labelling 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, respectively. Additionally, graphs are often distinguished into *unlabelled* and *labelled* graphs. In the latter case, both nodes and edges can be labelled 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. $L_V = L_E = \{\}$. For all of our graph representations described below, nodes are labelled with two-dimensional numerical labels, while edges remain unlabelled, i.e. $L_V = \mathbb{R}^2$ and $L_E = \{\}$.

- `Keypoint`: The first graph extraction algorithm makes use of characteristics points (so called keypoints) in skeletonised word images S. These keypoints are represented as nodes that are labelled 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 that is directly connected by a stroke.
- `Grid`: The second graph extraction algorithm is based on a grid-wise segmentation of binarised word images B into equally sized segments. For each segment, a node is inserted into the graph and labelled by the (x, y)-coordinates of its respective centre of mass. Undirected edges are inserted between two neighbouring segments that are actually represented by a node. Finally, the inserted edges are reduced by means of a *Minimal Spanning Tree* algorithm.
- `Projection`: The next graph extraction algorithm works on an adaptive and threshold-based segmentation of binarised word images B. Basically, this segmentation 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 labelled by the (x, y)-coordinates of the corresponding centre of mass. Undirected edges are inserted into the graph for each pair of nodes that is directly connected by a stroke in the original word image.
- `Split`: The fourth graph extraction algorithm is based on an iterative segmentation of binarised 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 labelled by the (x, y)-coordinates of the point on the stroke closest to the centre of mass of each segment. For the insertion of the edges, a similar procedure as for `Projection` is applied.

Finally, the resulting graphs or more precisely the (x, y)-coordinates of the node labels $\mu(v)$ are normalised by a z-score. Formally, we use normalised coordinates $(\hat{x}, \hat{y})$ derived by

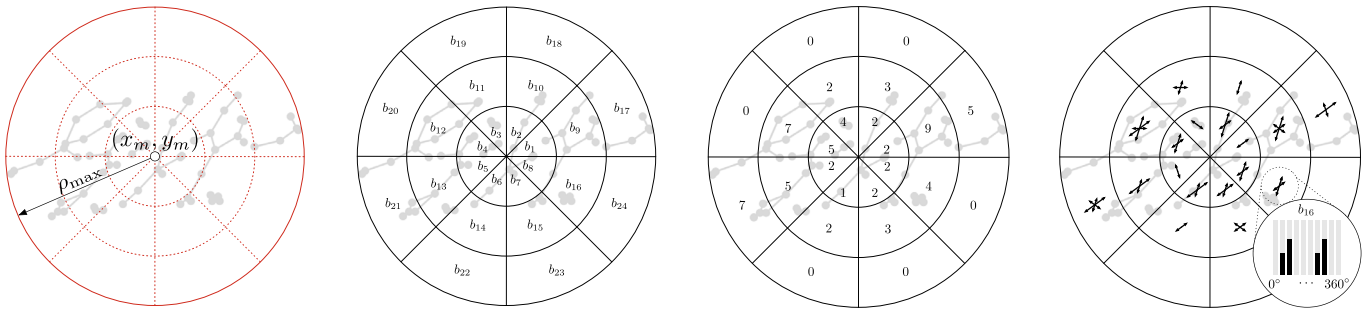
$$\hat{x} = \frac{x - \mu_x}{\sigma_x} \text{ and } \hat{y} = \frac{y - \mu_y}{\sigma_y}, \tag{1}$$

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

3. Filtering

Representing word images with graphs makes a certain graph matching necessary. Generally, graphs can be matched by means of *exact* or *inexact* dissimilarity measures. Inexact graph matching

[1] In case of two datasets, viz. AK and BOT (see Section 5.1 for details), segmented word images are directly taken from the ICFHR2016 benchmark database [24], and thus, only binarisation has been employed. To handle small segmentation errors, we employ an additional image preprocessing step that removes small connected components on these two manuscripts.



(a) Bounding circle C with centre of mass (x_m, y_m) and radius $\rho_{\max}$ (b) Segmentation of C into bins (c) Node-based histogram (d) Edge-based histogram

Fig. 2. Construction of the polar graph dissimilarity.

in contrast with exact graph matching is able to deal with small deviations in both structure and labelling of the graphs under consideration. The problem in either case is the high computational complexity. In KWS, where huge amounts of matchings have to be conducted, this might become a problem.

To reduce the number of graph matchings, we need an efficient and effective measure to estimate whether or not a questioned document graph $g \in G$ might be an instance of a given query graph q. If, and only if, this measure is below a certain threshold we apply the computational demanding graph matching between q and g (otherwise we filter g). Such a filter method needs to fulfil two requirements. First, the estimated graph dissimilarity needs to be coherent with the actual graph dissimilarity in order to filter graphs from G with high precision. Second, the computational complexity of the additional measure needs to be low such that the computational overhead can be compensated.

In order to meet these requirements, we introduce a novel graph dissimilarity measure with linear time complexity. This dissimilarity method measures the distance between histograms of spatial graph segments in a polar coordinate system. We denote this novel graph dissimilarity by *Polar Graph Dissimilarity (PGD)* and the corresponding distance d_{PGD} from now on.

PGD has been inspired by the scale-invariant shape descriptor *Contour Points Distribution Histogram (CPDH)* for matching 2D-shape images [35]. Basically this shape descriptor segments equidistant contour points by means of the polar coordinate system. Thus, a contour image can be formally described by a histogram CPDH $= \{h_1, \ldots, h_i, \ldots, h_n\}$ where h_i consists of the number of contour points in the corresponding segment. Finally, two shape images can be compared with each other by computing the *Earth Mover Distance* (or similar metrics) between shifted and mirrored histograms [34].

In the following paragraphs we explain how this concept is adapted to graphs $g = (V, E, \mu, \nu)$ with $L_V = \mathbb{R}^2$ and $L_E = \{\}$. First, graphs are segmented in a polar coordinate system. Second, histograms are extracted for each segment that represent the node and edge distributions. Finally, two graphs are compared on the basis of the resulting histograms.

Segmentation. We transform a given graph g into a polar coordinate system based on its centre of mass (x_m, y_m) as illustrated Fig. 2a.[2] Formally, the (x, y)-coordinates of each node label $\mu(v) = (x, y) \in \mathbb{R}^2$ are transformed to

$$\rho = \sqrt{(x - x_m)^2 + (y - y_m)^2} \text{ and } \theta_i = \text{atan2}((y - y_m)/(x - x_m)),$$

where ρ denotes the radius from the centre of g to the node position and $-\pi \leq \theta_i < \pi$ refers to the angle from the x-axis to the node position (computed via arctangent function with two arguments in order to return the correct quadrant).

Second, a bounding circle C with the maximum radius $\rho_{\max}$ that surrounds all nodes of graph g is defined. Based on this bounding circle C, the graph is segmented into $P_r \times P_\phi$ bins where P_r and P_ϕ define the number of different radii and angles, respectively. An example is given in Fig. 2b, where a graph is segmented into 24 bins (with $P_r = 3$ and $P_\phi = 8$). Note that every bin b_i is defined by two radii $\rho_{i_{\min}}$ and $\rho_{i_{\max}}$, and two angles $\theta_{i_{\min}}$ and $\theta_{i_{\max}}$. Hence, every node $v \in V$ with polar coordinates (ρ, θ) and $\rho_{i_{\min}} \leq \rho < \rho_{i_{\max}}$ and $\theta_{i_{\min}} \leq \theta < \theta_{i_{\max}}$ can be assigned to the corresponding bin b_i.

Node-based Histograms. The first histogram is created by counting the number of nodes per bin. That is, the histogram $H_i = \{h_1, \ldots, h_n\}$ represents the node frequency per bin b_i (see Fig. 2b and c). Finally, the resulting histograms are normalised by the $l1$-norm.

Edge-based Histograms. The second histogram reflects the distribution of both nodes and edges. To this end, we adapt the concept of *Histogram of Oriented Gradients* to (undirected) graphs to form a histogram with radial directions of the corresponding edges (the same adaptation can be applied to directed graphs as well). In particular, we first define the maximal number of subbins P that defines the radial range of every subbin b_{n_i}. For every edge in a segment, we measure the Euclidean distance d between the two adjacent nodes as well as the angle θ of the edge to the x-axis. Next, d is assigned to the two enclosing subbins b_{n_i} and b_{n_j} with respect to their radial difference to θ. Formally,

$$b_{n_i} \mathrel{+}= 1 - \frac{\theta - \theta_i}{\nu} d \text{ and } b_{n_j} \mathrel{+}= \frac{\theta - \theta_i}{\nu} d.$$

Note that every edge is taken into account in both directions as we make use of undirected edges. Finally, the resulting histogram $\{b_{n_1}, \ldots, b_{n_P}\}$ (i.e. one histogram per segment with P bins) is first concatenated to form one global histogram $H = \{h_1 = \{b_{1_1}, \ldots, b_{1_P}\}, \ldots, h_n = \{b_{n_1}, \ldots, b_{n_P}\}\}$ as illustrated in Fig. 2b and d and then normalised by the $l1$-norm. In our evaluation we set $P = 10$ for all subbins.

Polar Graph Dissimilarity. To measure the dissimilarity between two histograms H_1 and H_2 that represent the node and/or edge distribution of two graphs g_1 and g_2, respectively, we make use of the χ^2 distance. However, rather than comparing two histograms directly, we make use of a *quadtree* segmentation. That is, we segment the graph into smaller subgraphs and measure the dissimilarity between smaller subgraphs as formalised in Algorithm 1. First, the procedure is initialised by an external call with $l = 1$ (i.e. $PGD(1, g_1, g_2)$). On the basis of two graphs g_1 and g_2 the histograms H_1 and H_2 are created with respect to P_r and P_ϕ (see

[2] Node coordinates are *a priori* denormalised by the standard deviation of all node coordinates, for further details we refer to [37].

Algorithm 1 Polar Graph Dissimilarity (PGD).

Input: Graphs g_1 and g_2, number of radii and segments P_r and P_ϕ, recursion depth r
Output: Polar graph dissimilarity between graph g_1 and g_2
1: **function** PGD(l, g_1, g_2)
2: Create H_1 based on g_1, P_r, P_ϕ, and H_2 based on g_2, P_r, P_ϕ
3: Calculate χ^2-distance $d(H_1, H_2)$
4: **if** l equal r **then**
5: **return** d
6: Segment g_1 and g_2 based on quadtree to g_{1_1}, g_{1_2}, g_{1_3}, g_{1_4} and g_{2_1}, g_{2_2}, g_{2_3}, g_{2_4}
7: **return** $d(H_1, H_2) + \left(\sum_{i=1}^{4} \text{PGD}(l+1, g_{1_i}, g_{2_i}) \right)$

line 2 of Algorithm 1).[3] Next, the χ^2-distance between the two histograms is measured (see line 3). If the current recursion level l is equal to the maximal recursion depth r, the distance is returned (see lines 4 and 5). Otherwise, both graphs g_1 and g_2 are segmented into four independent subgraphs. Each of these subgraphs represent the nodes and edges in one of the four quadrants in circle C (see line 6). Eventually, for each subgraph pair, the PGD is measured by means of a recursive function call (see line 7). This procedure is repeated until the current recursion level l is equal to the user-defined maximum depth r.

If the resulting distance $d_{PGD}(q, g_i)$ between a query graph q and document graph g_i is below a certain threshold D, we additionally carry out the computationally more expensive *Bipartite Graph Edit Distance (BP)* (denoted by d_{BP} and thoroughly described in the next section) [28], otherwise we reject graph g_i and assign the graph dissimilarity to be ∞. Formally,

$$d(q, g_i) \begin{cases} d_{BP}(q, g_i), & \text{if } d_{PGD}(q, g_i) < D \\ \infty, & \text{otherwise} \end{cases}.$$

Clearly, if the threshold D is increased the number of filtered document graphs is reduced. Likewise, the number of filtered graphs is increased when threshold D is decreased. Overall we aim at finding a good tradeoff between high filter rates and low error rates.

We denote the graph matching procedure with this fast rejection procedure by BP-FRN in case of node-based histograms and BP-FRE in case of edge-based histograms.

4. Graph matching

We apply *Graph Edit Distance (GED)*, a powerful and flexible graph matching paradigm [6]. The basic idea of GED is 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. A set $\{e_1, \ldots, e_k\}$ of k edit operations e_i that transform g_1 completely into g_2 is called an *edit path* $\lambda(g_1, g_2)$ between g_1 and g_2.

To find the most suitable edit path, one commonly introduces a cost $c(e)$ for every edit operation e, measuring the strength of the corresponding operation. The idea of such a cost is to define whether or not an edit operation e represents a strong modification of the graph. Given an adequate cost model, the graph edit distance $d_{\text{GED}}(g_1, g_2)$, or d_{GED} for short, between g_1 and g_2 is defined by

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

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

[3] Note that P_r and P_ϕ can be defined for every recursion level separately.

For the exact computation of d_{GED}, A*-based search techniques using some heuristics are usually employed [4,15]. However, the search space of possible edit paths is exponential with respect to the number of nodes of the involved graphs. Formally, GED is an instance of a *Quadratic Assignment Problems (QAPs)* [18], which in turn belongs to the class of $\mathcal{NP}$-complete problems.[4]

To tackle the high computational complexity of GED, several fast but suboptimal algorithms have been proposed in the last years (see [12]). These concepts make GED also applicable to larger graphs. A cubic time approximation for GED has been proposed in [28], for instance. This algorithm reduces the QAP of GED to a *Linear Sum Assignment Problem* (*LSAP*) that can be optimally solved in cubic time (see [7] for an exhaustive survey on LSAP solving algorithms). The optimal LSAP solution is eventually used to derive a suboptimal GED.

The employed cost model in our case is based on constant cost for both node and edge deletions/insertions, i.e. $\tau_v \in \mathbb{R}^+$ and $\tau_e \in \mathbb{R}^+$, respectively. The cost for node substitutions reflects the dissimilarity of the associated label attributes, i.e. (x, y)-coordinates. Formally, the cost for substituting node n_i with $\mu(n_i) = (x_i, y_i)$ and node n_j with $\mu(n_j) = (x_j, y_j)$ is given by a weighted Euclidean distance

$$\sqrt{\alpha\, \sigma_x (x_i - x_j)^2 + (1-\alpha)\, \sigma_y (y_i - y_j)^2},$$

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. We additionally use a weighting factor $\beta \in [0, 1]$ to weight the relative importance of node and edge edit costs.

Retrieval Index. For spotting keywords, we build two retrieval indices that are separately optimised for a *local* and *global* threshold scenario. Local thresholds are used in case of a vocabulary of common keywords, while a global threshold is used for arbitrary out-of-vocabulary keywords. In case of local thresholds, the accuracy is independently measured for every query word, while in case of global thresholds, the accuracy is measured for every query word with the same single threshold.[5] Global threshold are thus regarded as more realistic yet also more difficult scenario.

For building the retrieval indices we first normalise the graph edit distances d_{BP} between query graph q and all document graphs $G = \{g_1, \ldots, g_N\}$ by the sum of the maximum cost edit path between q and g_i, i.e. the sum of the edit path that results from deleting all nodes and edges of q and inserting all nodes and edges in g_i. Formally,

$$\hat{d}_{\text{BP}}(q, g_i) = \frac{d_{\text{BP}}(q, g_i)}{(|V_q| + |V_{g_i}|)\, \tau_v + (|E_q| + |E_{g_i}|)\, \tau_e}.$$

In case a query consists of a set of graphs $\{q_1, \ldots, q_t\}$ that represent the same keyword, the normalised graph edit distance $\hat{d}_{\text{BP}}$ is given by the minimal distance achieved on all t query graphs, i.e. $\min_{q_j \in \{q_1, \ldots, q_t\}} \hat{d}_{\text{BP}}(q_j, g_i)$.

Based on normalised distances, the retrieval index for local thresholds is derived by

$$r_1(q, g) = -\hat{d}_{\text{BP}}(q, g),$$

while the retrieval index for global thresholds is derived by

$$r_2(q, g) = -\frac{\hat{d}_{\text{BP}}(q, g)}{\omega},$$

where ω is a linear scaling factor based on the mean distance of q to its ten nearest neighbours (denoted by $\bar{d}_{BP}$ from now on) and

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

[5] In both cases, the threshold defines whether a document word is regarded relevant for a given query word not.

Please cite this article as: M. Stauffer et al., Filters for graph-based keyword spotting in historical handwritten documents, Pattern Recognition Letters (2018), https://doi.org/10.1016/j.patrec.2018.03.030

Table 1
The number of keywords as well as the size of the training and test sets for all four documents.

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

the minimum mean distance of all available query graphs (denoted by $\bar{d}_{BP_{\min}}$). Formally, ω is given by

$$\omega = 1 + m(\bar{d}_{BP} - \bar{d}_{BP_{\min}}),$$

where m is a user defined scaling slope. The basic idea of this procedure is to gradually scale $\hat{d}_{\mathrm{BP}}(q, g)$ depending on the mean distance of q to its ten next neighbours. This is expected to reduce the intraclass variance between different queries and improve the accuracy for global thresholds.

5. Experimental evaluation

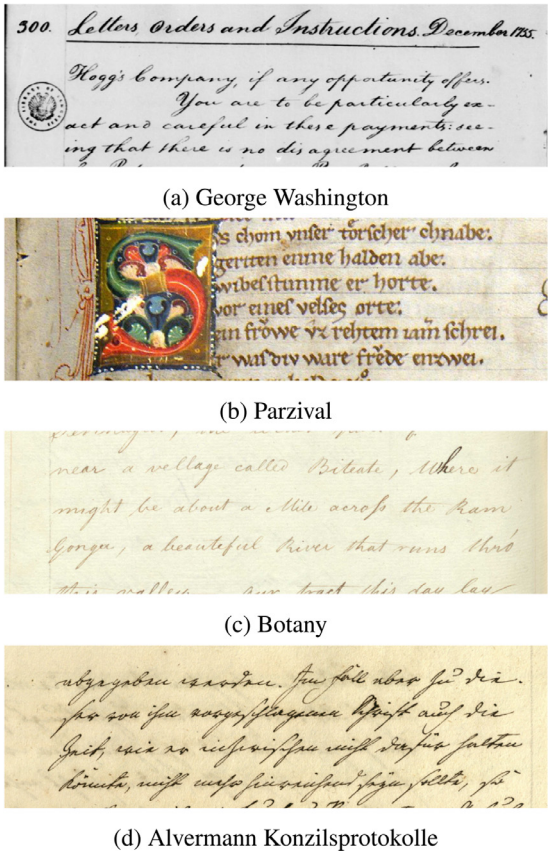
5.1. Datasets

The evaluation is based on two well known manuscripts, viz. *George Washington (GW)*[6] and *Parzival (PAR)*,[7] as well as two recent KWS benchmark datasets,[8] viz. *Alvermann Konzilsprotokolle (AK)* and *Botany (BOT)*. GW is written in English and based on twenty pages with minor variations in writing and degradation. PAR is written in Middle High German and based on 45 pages with low writing variations but markable signs of degradation. AK is written in German and based on 18,000 pages with minor variations and signs of degradation. Finally, BOT is written in English and based on ten pages with high writing variation and markable signs of degradation. On all four documents we extract graphs from segmented word images by means of the graph extraction algorithms introduced in Section 2.2. For AK and BOT, we only consider the two most promising graph extraction algorithms, i.e. `Keypoint` and `Projection`. Small excerpts of all four manuscripts and the corresponding graph representations are shown in Figs. 3 and 4.

On the resulting sets of word graphs, ten different keywords (with different word lengths) are manually selected on all four datasets. Moreover, we define an independent validation set for parameter optimisation that consists of 10 random instances per keyword instance and 900 additional random words (in total 1000 words). The optimised systems are eventually evaluated on the same training and test sets as used in [10] for GW and PAR and [24] for AK and BOT. All templates of a keyword present in the training set are used for KWS. In Table 1 a summary of the datasets is given.

5.2. Optimisation of the parameters

The proposed KWS framework is separately optimised for local and global thresholds. In the global threshold scenario, the *Average Precision (AP)* is measured, which is the area under the Recall-Precision curve for all keywords given a single threshold. In the local threshold scenario, the *Mean Average Precision (MAP)* is computed, that is the mean over the AP of each individual keyword query. To measure the effects of the proposed rejection methods, we compute the relative amount of pairwise matchings that is filtered (termed *Filter Rate (FR)*) and the *Speed-up Factor (SF)* when compared to the matching time of BP.



(a) George Washington

(b) Parzival

(c) Botany

(d) Alvermann Konzilsprotokolle

Fig. 3. Exemplary excerpts of the four historical manuscripts.

The optimisation of the parameters is conducted in three steps on the validation set. First, we optimise the cost functions for the edit distance computation. Second, the parameters for the PGD filters are optimised. Third, threshold D is used to optimally adjust the filter rate. These three steps are described in detail in the next three paragraphs.

Optimisation of BP. The parameters for graph edit distance are individually optimised for both MAP and AP. That is, we evaluate 25 pairs of constants for node and edge deletion/insertion costs ($\tau_v = \tau_e = \{1, 4, 8, 16, 32\}$) in combination with the weighting parameters $\alpha = \{0.1, 0.3, 0.5, 0.7, 0.9\}$ and $\beta = \{0.1, 0.3, 0.5, 0.7, 0.9\}$. Hence, we evaluate a total of 625 parametrisations per graph extraction method and dataset. Using optimised cost functions, we optimised our framework for a global threshold scenario using retrieval index r_2. That is, we optimised the scaling factor $m = \{0.05, \ldots, 10.0\}$ for all graph extraction methods and datasets. For local thresholds no additional parameter tuning has to be conducted. In Table 2 the optimal cost function parameters as well as the scaling factor for global retrieval indices are given for all graph extraction algorithms and datasets.

Optimisation of PGD The parameters of PGD are optimised with respect to AP. On the validation set different polar segmentations (defined via P_r and P_ϕ) are validated for two recursion levels (i.e. we define the maximal recursion depth to $r = 2$). For $l = 1$, the parameter combinations $P_r = \{1, 2, 3, 4, 5, 6\} \times P_\phi = \{4, 8, 12, 16, 20, 24, 28, 32, 36, 40\}$ are evaluated, while for

[6] 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.

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

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

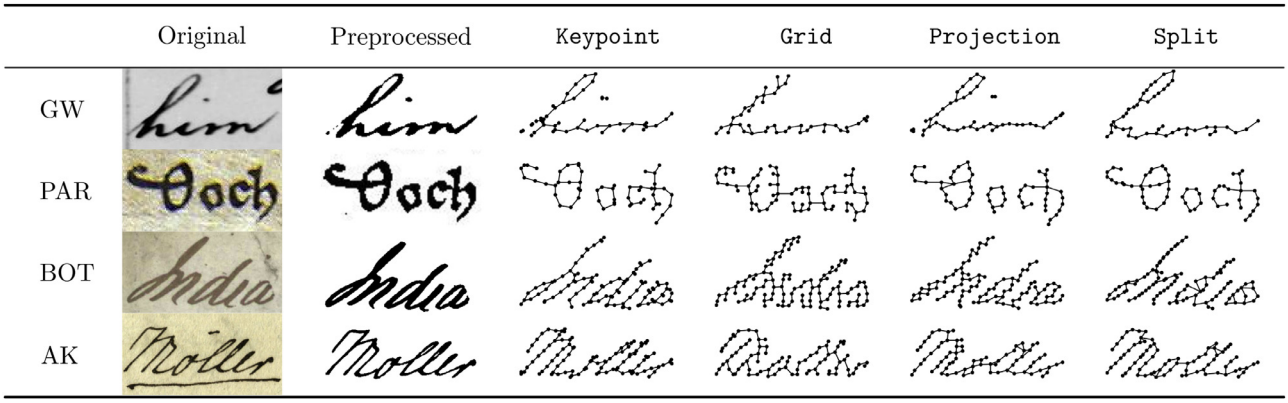


Fig. 4. Exemplary graph representations of the George Washington (GW), Parzival (PAR), Botany (BOT), and Alvermann Konzilsprotokolle (AK) database.

Table 2
Optimal cost function parameters and scaling factors for graph edit distance computation.

Method		τ_v	τ_e	α	β	m
GW	Keypoint	4	1	0.1	0.5	4.55
	Grid	4	1	0.1	0.7	4.70
	Projection	4	1	0.1	0.5	4.90
	Split	4	1	0.1	0.5	4.75
PAR	Keypoint	4	4	0.3	0.5	2.50
	Grid	4	1	0.5	0.7	3.60
	Projection	4	1	0.5	0.5	3.90
	Split	4	1	0.3	0.3	2.65
BOT	Keypoint	32	32	0.1	0.3	3.30
	Projection	8	32	0.3	0.9	6.05
AK	Keypoint	16	16	0.1	0.5	3.35
	Projection	8	32	0.1	0.7	2.35

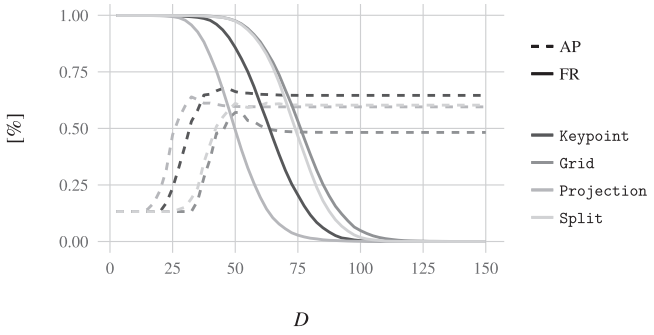


Fig. 5. Average precision (AP) and filter rate (FR) for BP-FRE as function of threshold D.

Table 3
Optimal P_r and P_ϕ for PGD on both recursion levels l in conjunction with node- and edge-based histograms, respectively.

Method		Node				Edge			
		$l=1$		$l=2$		$l=1$		$l=2$	
		P_r	P_ϕ	P_r	P_ϕ	P_r	P_ϕ	P_r	P_ϕ
GW	Keypoint	5	8	1	4	4	16	1	4
	Grid	4	8	1	2	5	40	1	4
	Projection	5	16	1	4	6	4	1	2
	Split	6	24	1	2	5	40	1	4
PAR	Keypoint	6	36	4	8	3	36	3	4
	Grid	1	36	1	8	6	36	4	4
	Projection	6	36	4	4	4	36	4	8
	Split	2	36	3	8	6	8	3	8
BOT	Keypoint	6	40	1	4	1	16	2	4
	Projection	4	40	4	4	1	36	2	4
AK	Keypoint	4	20	1	2	4	4	1	4
	Projection	4	20	1	10	4	20	2	4

Table 4
Optimal D and corresponding filter rate (FR).

Method		BP-FRN		BP-FRE	
		D	FR	D	FR
GW	Keypoint	7.50	88.08	45.0	93.57
	Grid	4.50	89.43	55.0	94.17
	Projection	8.25	91.55	35.0	92.66
	Split	12.75	56.74	62.5	82.75
PAR	Keypoint	24.00	82.49	65.0	94.12
	Grid	6.75	91.91	77.5	94.48
	Projection	17.25	95.51	82.5	91.00
	Split	16.50	88.42	65.0	94.66
BOT	Keypoint	13.50	95.97	92.5	84.59
	Projection	19.50	91.09	105.0	93.78
AK	Keypoint	7.50	96.00	60.0	89.18
	Projection	12.75	92.36	110.0	90.03

$l=2$ the parameter combinations $P_r = \{1,2,3,4\} \times P_\phi = \{2,4,6,8,10\}$ are evaluated. Hence, in total we evaluate $6 \times 10 \times 4 \times 5 = 1200$ parameter combinations for every graph extraction method. In Table 3 the best performing parameters are presented for all graph extraction methods and datasets.

Optimisation of Filtering. Finally, we optimise the threshold D that controls the amount of filtered graphs. For BP-FRN we evaluate thresholds $D = \{0.75, 1.5, \ldots, 29.25, 30\}$ and for BP-FRE we evaluate thresholds $D = \{2.5, 5, \ldots, 147.5, 150\}$. In Fig. 5 the AP and FR for BP-FRE are shown for every tested threshold D on the GW dataset (similar plots can be achieved on all other datasets). By increasing D we observe that the KWS performance is improved in general. Simultaneously, the number of filtered graphs is decreasing (making the KWS process slower in general). Threshold D is determined such that the AP is maximised (if this threshold is actually too restrictive, we choose the next higher threshold where the AP is not further decreasing). In Table 4 the selected threshold D and the corresponding filter rates FR are given for each graph extraction method and all four datasets.

5.3. Reference systems

In order to evaluate the two proposed fast rejection heuristics BP-FRN and BP-FRE, we consider three types of reference systems,

viz. (1) a recent graph-based system, (2) template-based systems using DTW, and (3) learning-based KWS systems.

First, we compare the novel filter methods with the graph-based KWS framework without fast rejection (denoted by BP) proposed in [37] on all datasets.

Next, we compare the proposed approach with four template-based KWS systems using DTW on the GW and PAR datasets.[9] These systems optimally align sequences of feature vectors like geometrical features [23] (denoted by DTW'01), Histogram of Oriented Gradient features [29,43] (denoted by DTW'08 and DTW'09, respectively), and Deep Learning features [48] (denoted by DTW'16).

Finally, we compare our method with three state-of-the-art learning-based methods, viz. CVCDAG [2], PRG [42], and QTOB [49], on the BOT and AK datasets.[10] CVCDAG is based on *Pyramidal Histogram Of Characters labels (PHOC)* features used in conjunction with a SVM [2]. In PRG, the same features are used to train a CNN, the so called PHOCNet [42]. Another CNN is used in QTOB by means of a triplet network approach [49].

5.4. Results and discussion

First, we compare the proposed rejection methods BP-FRN and BP-FRE with the graph-based KWS framework BP without fast rejection [37] on the independent test sets. In Table 5 the MAP for local thresholds, the AP for global thresholds, as well as the FR is given for all methods. On the GW dataset we observe filter rates between 60% and 90% for BP-FRN, and 85% and 95% for BP-FRE (i.e. only 5% to 40% of all comparisons have to be carried out by the bipartite graph matching algorithm). Due to this filtering, we observe speed-up factors of 4 to 21 when compared with the original framework, as shown in Table 6.[11]

Simultaneously both methods BP-FRN and BP-FRE achieve better KWS accuracies than the plain method BP. In particular, the KWS accuracy can be improved by up to 4%. When compared with BP-FRN, BP-FRE generally leads to higher filter rates as well as higher accuracies in both threshold scenarios. In particular, BP-FRE in combination with `Keypoint` graphs lead to promising results.

Given the high accuracies achieved by our filter methods BP-FRN and BP-FRE, one might wonder how the graph dissimilarity PGD performs as a single graph dissimilarity measure for KWS, rather than as filter method. In Table 7, the MAP for PGD with node-based histograms (termed PGD-Node) and edge-based histograms (termed PGD-Edge) is compared against BP using `Keypoint` graphs (for the other graphs similar results are obtained). We observe that PGD-Node results in a lower KWS accuracy when compared to BP, whereas PGD-Edge performs similar or even better than BP. Thus, we conclude that PGD-Edge can be employed for both filtering and for computing a basic dissimilarity measure for KWS.

Next, we compare the two novel graph-based methods BP-FRN and BP-FRE (using the best performing graph extraction method) with four DTW-based reference system for template-based KWS on GW and PAR. In Table 8, the MAP for each dataset as well as their average is given. On GW we observe that both graph-based methods clearly outperform the DTW-based reference systems. Especially, BP-FRE leads to substantial improvements when compared to all DTW systems. On PAR both DTW'09 and DTW'16 are slightly better than our graph-based methods. However, these methods

Table 5
Mean average precision (MAP) using local thresholds, average precision (AP) using a global threshold, and filter rate (FR) for KWS using the original bipartite graph matching without rejection (BP) and with the proposed fast rejection methods BP-FRN and BP-FRE. With ± we indicate the relative percental gain or loss in the accuracy of BP-FR when compared with BP.

Method		GW				PAR				BOT				AK			
		MAP	±	AP	±	FR	MAP	±	AP	±	FR	MAP	±	AP	±	FR
BP	`Keypoint`	66.08		54.37		00.00	62.04		61.36		00.00	45.06		33.84		0.00	77.24		76.32		
	`Grid`	60.02		45.52		00.00	56.50		43.55		00.00	-	-	-	-	-	-	-	-	-	-
	`Projection`	61.43		48.45		00.00	66.23		64.23		00.00	49.57		38.33		0.00	76.02		74.38		
	`Split`	60.23		48.41		00.00	59.44		58.29		00.00	-	-	-	-	-	-	-	-	-	-
BP-FRN	`Keypoint`	69.81	+3.73	56.45	+2.08	86.97	67.28	+5.24	67.04	+5.68	73.60	56.10	+11.04	39.85	+6.01	85.64	**81.51**	+4.27	79.01	+2.69	85.45
	`Grid`	62.85	+2.83	46.39	+0.86	90.92	62.33	+5.83	59.52	+15.97	86.13	-	-	-	-	-	-	-	-	-	-
	`Projection`	65.20	+3.77	51.19	+2.75	93.02	71.09	+4.86	**71.77**	+7.54	93.85	53.77	+4.20	37.75	-0.58	84.20	79.09	+3.07	77.20	+2.82	91.09
	`Split`	63.15	+2.92	51.27	+2.86	58.17	65.43	+5.99	66.80	+8.51	82.84	-	-	-	-	-	-	-	-	-	-
BP-FRE	`Keypoint`	**70.61**	+4.52	**57.04**	+2.67	95.32	68.16	+6.12	67.42	+6.06	92.53	**57.14**	+12.08	**40.48**	+6.64	84.75	**81.51**	+4.27	**79.91**	+3.59	88.57
	`Grid`	62.86	+2.84	45.20	−0.32	95.78	63.12	+6.62	59.85	+16.30	91.61	-	-	-	-	-	-	-	-	-	-
	`Projection`	65.51	+4.08	50.69	+2.25	94.79	**72.03**	+5.80	71.48	+7.25	84.20	52.56	+2.99	36.91	-1.42	86.20	**81.51**	+5.49	79.26	+4.88	92.48
	`Split`	64.01	+3.78	51.78	+3.37	85.38	66.49	+7.05	65.73	+7.44	92.69	-	-	-	-	-	-	-	-	-	-

[9] Template-based reference results are only available for GW and PAR.

[10] Learning-based reference results are only available for BOT and AK.

[11] Actually, we carry out our experiment on a high performance computing cluster with dozens of CPU nodes. Hence, these readings are approximated by means of the average matching time per keyword measured on the validation set in a sequential scenario.

Please cite this article as: M. Stauffer et al., Filters for graph-based keyword spotting in historical handwritten documents, Pattern Recognition Letters (2018), https://doi.org/10.1016/j.patrec.2018.03.030

Table 6
Mean average precision (MAP) for local thresholds and speed-up factor (SF), for the original bipartite graph matching (BP), and the bipartite fast rejection with nodes (BP-FRN) and edges (BP-FRE), respectively, on the `Keypoint` graphs.

	GW	SF	PAR	SF	BOT	SF	AK	SF
BP	66.08		62.04		45.06		77.24	
BP-FRN	69.81	7.68	67.28	3.79	56.10	6.97	**81.51**	6.87
BP-FRE	**70.61**	21.35	68.16	13.38	**57.14**	6.56	**81.51**	8.75

Table 7
Mean average precision for local thresholds for the original bipartite graph matching (BP), and the polar graph dissimilarity with nodes (PGD-Node) and edges (PGD-Edge), respectively, on the `Keypoint` graphs.

	GW	±	PAR	±	BOT	±	AK	±
BP	66.08		62.04		45.06		77.24	
PGD-Node	58.47	−7.62	46.71	−15.33	45.81	+0.75	69.66	−7.58
PGD-Edge	68.78	+2.70	61.47	−0.57	52.11	+7.05	76.17	−1.07

Table 8
Mean average precision (MAP) using local thresholds for graph-based KWS systems in comparison with four 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'01	45.26		46.78		46.02	
	DTW'08	63.39		47.52		55.46	
	DTW'09	64.80		73.49	(1)	69.15	
	DTW'16	68.64	(3)	72.38	(2)	70.51	(2)
Graph	BP-FRN	69.81	(2)	71.09		70.45	(3)
	BP-FRE	70.61	(1)	72.03	(3)	71.32	(1)

Table 9
Mean average precision (MAP) using local thresholds 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		82.15	(2)	68.55	
Graph	BP-FRN	56.10		81.51	(3)	68.81	
	BP-FRE	57.14	(3)	81.51	(3)	69.33	(3)

are based on highly sophisticated features. In particular, DTW'16 makes use of an unsupervised feature learning approach. Overall, we conclude that both graph-based methods are able to outperform, or at least keep up with, the DTW-based reference systems.

Finally, in Table 9 we compare our novel methods for graph-based KWS with three learning-based methods on the two ICFHR2016 benchmark datasets. We observe that both fast rejection methods obtain high accuracy rates. Yet, BP-FRE is slightly better in case of BOT when compared to BP-FRN. Among the learning-based methods we observe that both PHOC-based methods, i.e. CVCDAG and PRG, result in the overall best performance. Especially, the combination of a CNN and PHOC features (PRG) results in remarkable high accuracies on all datasets.

However, the proposed graph-based methods can keep up or even outperform several learning-based methods, especially on AK. This is quite interesting as the reference methods are based on more advanced features than our approach and make use of sophisticated learning-based algorithms (i.e. SVM and CNN).

Considering that manual labelling of historical handwriting is a labour- and cost-intensive process, and, the limited availability of training data, our graph-based methods become a valuable and flexible alternative especially as only one single keyword is required for retrieval without a priori training.[12]

6. Conclusion and outlook

In the present paper two novel filter methods for graph-based KWS are introduced. These filters allow to decide in linear time whether or not a graph from a document is similar enough to a given query graph. In particular, the filter methods compare histograms of the node and edges distributions in a polar coordinate system. If, and only if, two histograms are similar enough, the more powerful and accurate graph matching is actually carried out, otherwise the document graph is rejected. Due to this filtering, more than 90% of all matchings can be omitted. The proposed rejection criterion is computed in linear time, while the graph matching has cubic time complexity. Thus, we observe substantial speed-ups of the complete KWS process. Moreover, the proposed filters also improve the KWS accuracy in most scenarios. Hence, the advantage of our novel framework is twofold: It makes KWS fast *and* more accurate.

Finally, we show that the two novel filters for graph-based KWS can keep up or even outperform several state-of-the-art template- and learning-based systems. This is quite remarkable as some of these reference systems are based on advanced features and use learning-based matching algorithms (e.g. CNN). Moreover, it is worth to note that the learning systems are crucially depending on the size of labelled training data. Yet, such training data is often difficult to acquire in case of handwritten historical documents. This makes our graph-based methods to a viable alternative, especially as only one keyword template is necessary for retrieval.

In future work we aim to optimise and automise the thresholding of the filter methods, such that no additional optimisation step is required. Moreover, as our novel linear time graph comparison achieves quite high accuracies, one might consider to use the proposed graph filter as an independent dissimilarity measure in other graph-based pattern recognition applications.

Acknowledgement

This work has been supported by the Hasler Foundation Switzerland.

References

[1] S.S. Kuo, O. Agazzi, Keyword spotting in poorly printed documents using pseudo 2-d hidden markov models, IEEE Trans. Pattern Anal. Mach. Intell. 16 (1994) 842–848.
[2] J. Almazán, A. Gordo, A. Fornés, E. Valveny, Word spotting and recognition with embedded attributes, IEEE Trans. Pattern Anal. Mach. Intell. 36 (2014) 2552–2566.
[3] M. Ameri, M. Stauffer, K. Riesen, T. Bui, A. Fischer, Keyword spotting in historical documents based on handwriting graphs and hausdorff edit distance, in: Int. Graphonomics Soc. Conf., 2017.
[4] S. Berretti, A. Del Bimbo, E. Vicario, Efficient matching and indexing of graph models in content-based retrieval, IEEE Trans. Pattern Anal. Mach. Intell. 23 (2001) 1089–1105.
[5] Q.A. Bui, M. Visani, R. Mullot, Unsupervised word spotting using a graph representation based on invariants, in: Int. Conf. Doc. Anal. Recognit., 2015, pp. 616–620.
[6] H. Bunke, G. Allermann, Inexact graph matching for structural pattern recognition, Pattern Recognit. Lett. 1 (1983) 245–253.
[7] R. Burkard, M. Dell'Amico, S. Martello, Assignment problems, 2009,
[8] J. Edwards, Y.W. Teh, R. Bock, M. Maire, G. Vesom, D.A. Forsyth, Making latin manuscripts searchable using gHMM's, in: Int. Conf. Neural Inf. Process. Syst., 2004, pp. 385–392.
[9] A. Fischer, E. Indermühle, H. Bunke, G. Viehhauser, M. Stolz, Ground truth creation for handwriting recognition in historical documents, in: Int. Work. Doc. Anal. Syst., 2010, pp. 3–10. New York, New York, USA.

[12] Some of the reference systems need relatively large training sets, i.e. labelled training data (e.g. PRG achieves lower rates on penalised/weighted MAP, see [24] for details).

[10] A. Fischer, A. Keller, V. Frinken, H. Bunke, Lexicon-free handwritten word spotting using character HMMs, Pattern Recognit. Lett. 33 (2012) 934–942.
[11] A. Fischer, C.Y. Suen, V. Frinken, K. Riesen, H. Bunke, Approximation of graph edit distance based on hausdorff matching, Pattern Recognit. 48 (2015) 331–343.
[12] 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 (2014) 1450001.
[13] V. Frinken, A. Fischer, M. Baumgartner, H. Bunke, Keyword spotting for self-training of BLSTM NN based handwriting recognition systems, in: Pattern Recognit., 2014, pp. 1073–1082.
[14] 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 (2012) 211–224.
[15] L. Gregory, J. Kittler, Using graph search techniques for contextual colour retrieval, in: Int. Work. Struct. Syntactic, Stat. Pattern Recognit., 2002, pp. 186–194.
[16] Z. Guo, R.W. Hall, Parallel thinning with two-subiteration algorithms, Commun. ACM 32 (1989) 359–373.
[17] T. Konidaris, A.L. Kesidis, B. Gatos, A segmentation-free word spotting method for historical printed documents, Pattern Anal. Appl. (2015).
[18] T.C. Koopmans, M. Beckmann, Assignment problems and the location of economic activities, Econometrica 25 (1957) 53.
[19] V. Lavrenko, T. Rath, R. Manmatha, Holistic word recognition for handwritten historical documents, in: Int. Work. Doc. Image Anal. Libr., 2004, pp. 278–287.
[20] Y. Leydier, F. Lebourgeois, H. Emptoz, Text search for medieval manuscript images, Pattern Recognit. 40 (2007) 3552–3567.
[21] G. Lluis, M. Rusinol, D. Karatzas, LSDE : levenshtein space deep embedding for query-by-string word spotting, in: Int. Conf. Doc. Anal. Recognit., Kyoto, 2017.
[22] R. Manmatha, H. Chengfeng, E. Riseman, Word spotting: a new approach to indexing handwriting, in: Comput. Vis. Pattern Recognit., 1996, pp. 631–637.
[23] U.V. Marti, H. Bunke, Using a statistical language model to improve the performance of an HMM-based cursive handwriting recognition systems, Int. J. Pattern Recognit. Artif. Intell. 15 (2001) 65–90.
[24] I. Pratikakis, K. Zagoris, B. Gatos, J. Puigcerver, A.H. Toselli, E. Vidal, ICFHR2016 handwritten keyword spotting competition (h-KWS 2016), in: Int. Conf. Front. Handwrit. Recognit., 2016, pp. 613–618.
[25] T. Rath, R. Manmatha, Word image matching using dynamic time warping, Comput. Vis. Pattern Recognit., 2003.
[26] P. Riba, J. Lladós, A. Fornés, Handwritten word spotting by inexact matching of grapheme graphs, in: Int. Conf. Doc. Anal. Recognit., 2015, pp. 781–785.
[27] K. Riesen, Structural pattern recognition with graph edit distance, Advances in Computer Vision and Pattern Recognition, Cham, 2015.
[28] K. Riesen, H. Bunke, Approximate graph edit distance computation by means of bipartite graph matching, Image Vis. Comput. 27 (2009) 950–959.
[29] J.A. Rodríguez-Serrano, F. Perronnin, Local gradient histogram features for word spotting in unconstrained handwritten documents, in: Int. Conf. Front. Handwrit. Recognit., 2008, pp. 7–12.
[30] J.A. Rodríguez-Serrano, F. Perronnin, Handwritten word-spotting using hidden markov models and universal vocabularies, Pattern Recognit. 42 (2009) 2106–2116.
[31] R. Rose, D. Paul, A hidden Markov model based keyword recognition system, in: IEEE Int. Conf. Acoust. Speech, Signal Process., 1990, pp. 129–132.
[32] L. Rothacker, M. Rusinol, G.A. Fink, Bag-of-features HMMs for segmentation-free word spotting in handwritten documents, in: Int. Conf. Doc. Anal. Recognit., 2013, pp. 1305–1309.
[33] L. Rothacker, S. Sudholt, E. Rusakov, M. Kasperidus, G.A. Fink, Word hypotheses for segmentation-free word spotting in historic document images, in: Int. Conf. Doc. Anal. Recognit., Kyoto, 2017.
[34] Y. Rubner, The earth mover's distance as a metric for image retrieval, Int. J. Comput. Vis. 40 (2000) 99–121.
[35] X. Shu, X.J. Wu, A novel contour descriptor for 2d shape matching and its application to image retrieval, Image Vis. Comput. 29 (2011) 286–294.
[36] M. Stauffer, A. Fischer, K. Riesen, A novel graph database for handwritten word images, Int. Work. Struct. Syntactic, Stat. Pattern Recognit., 2016.
[37] M. Stauffer, A. Fischer, K. Riesen, Graph-based keyword spotting in historical handwritten documents, Int. Work. Struct. Syntactic, Stat. Pattern Recognit., 2016.
[38] M. Stauffer, A. Fischer, K. Riesen, Ensembles for graph-based keyword spotting in historical handwritten documents, in: Int. Conf. Doc. Anal. Recognit., 2017.
[39] M. Stauffer, A. Fischer, K. Riesen, Speeding-up graph-based keyword spotting by quadtree segmentations, in: Int. Conf. Comput. Anal. Images Patterns, 2017.
[40] M. Stauffer, A. Fischer, K. Riesen, Speeding-up graph-based keyword spotting in historical handwritten documents, Graph-Based Represent. Pattern Recognit., 2017.
[41] M. Stauffer, T. Tschachtli, A. Fischer, K. Riesen, A survey on applications of bipartite graph edit distance, Graph-Based Represent. Pattern Recognit., 2017.
[42] S. Sudholt, G.A. Fink, PHOCNet: a deep convolutional neural network for word spotting in handwritten documents, in: Int. Conf. Front. Handwrit. Recognit., 2016, pp. 277–282.
[43] K. Terasawa, Y. Tanaka, Slit style HOG feature for document image word spotting, in: Int. Conf. Doc. Anal. Recognit., 2009, pp. 116–120.
[44] S. Thomas, C. Chatelain, L. Heutte, T. Paquet, Y. Kessentini, A deep HMM model for multiple keywords spotting in handwritten documents, Pattern Anal. Appl. 18 (2014) 1003–1015.
[45] J.I. Toledo, S. Sudholt, A. Fornés, J. Cucurull, G.A. Fink, J. Lladós, Handwritten word image categorization with convolutional neural networks and spatial pyramid pooling, in: Int. Work. Struct. Syntactic, Stat. Pattern Recognit., 2016, pp. 543–552.
[46] P. Wang, V. Eglin, C. Garcia, C. Largeron, J. Lladós, A. Fornés, A coarse-to-fine word spotting approach for historical handwritten documents based on graph embedding and graph edit distance, in: Int. Conf. Pattern Recognit., 2014, pp. 3074–3079.
[47] P. Wang, V. Eglin, C. Garcia, C. Largeron, J. Lladós, A. Fornés, A novel learning-free word spotting approach based on graph representation, in: Int. Work. Doc. Anal. Syst., 2014, pp. 207–211.
[48] B. Wicht, A. Fischer, J. Hennebert, Deep learning features for handwritten keyword spotting, in: Int. Conf. Pattern Recognit., 2016.
[49] T. Wilkinson, A. Brun, Semantic and verbatim word spotting using deep neural networks, Int. Conf. Front. Handwrit. Recognit. (2016) 307–312.
[50] B. Zhang, S.N. Srihari, C. Huang, Word image retrieval using binary features, in: Doc. Recognit. Retr., 2003, p. 45.

Please cite this article as: M. Stauffer et al., Filters for graph-based keyword spotting in historical handwritten documents, Pattern Recognition Letters (2018), https://doi.org/10.1016/j.patrec.2018.03.030