Pattern Recognition 81 (2018) 240–253

Contents lists available at ScienceDirect

Pattern Recognition

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

Keyword spotting in historical handwritten documents based on graph matching

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 of Complex Systems, Fribourg 1705, Switzerland
[d] University of Pretoria, Department of Informatics, Pretoria, South Africa

ARTICLE INFO

Article history:
Received 4 September 2017
Revised 6 January 2018
Accepted 2 April 2018
Available online 6 April 2018

Keywords:
Handwritten keyword spotting
Graph representation
Bipartite graph matching
Ensemble methods

ABSTRACT

In the last decades historical handwritten documents have become increasingly available in digital form. Yet, the accessibility to these documents with respect to browsing and searching remained limited as full automatic transcription is often not possible or not sufficiently accurate. This paper proposes a novel reliable approach for template-based keyword spotting in historical handwritten documents. In particular, our framework makes use of different graph representations for segmented word images and a sophisticated matching procedure. Moreover, we extend our method to a spotting ensemble. In an exhaustive experimental evaluation on four widely used benchmark datasets we show that the proposed approach is able to keep up or even outperform several state-of-the-art methods for template- and learning-based keyword spotting.

1. Introduction

In order to bridge the gap between availability and accessibility of ancient handwritten documents *handwriting recognition* is often employed for an automatic and complete transcription. In the case of historical documents this process is inherently an *offline* task, and as such, more complex than *online* handwriting recognition where temporal information is available [1]. Moreover, the recognition rates of handwriting recognition systems applied to ancient documents is often negatively affected by both the degenerative conservation state of scanned documents [2] and different writing styles [3].

In order to overcome the prior obstacles of automatic full transcriptions of historical handwritten documents, *Keyword Spotting (KWS)* as a more error-tolerant, flexible, and suitable approach has been proposed [4–7]. KWS refers to the task of retrieving any instance of a given query word in a certain document. The concept of KWS was originally proposed for speech documents [8] and later adapted for printed [9] and handwritten documents [4].

To date, KWS is still highly relevant in different application domains. This is particularly due to the global trend towards digitalisation of paper-based archives and libraries in both private and public institutions. Clearly, KWS techniques can make these documents accessible for searching and browsing [5]. Similar to handwriting recognition, textual KWS can be divided into *online* and *offline* KWS. The focus of this paper is on historical documents, and thus, offline KWS, referred to as KWS from now on, can be applied only.

Most of the KWS methodologies available are based on *template-based* or *learning-based* algorithms (similar to the corresponding subfields in handwriting recognition). Early approaches of template-based KWS are based on pixel-by-pixel matchings of word images [4] by either Euclidean distance measures or affine transformations by the *Scott and Longuet–Higgins* algorithm [10]. More elaborated and error-tolerant approaches to template-based KWS are based on the matching of feature vectors that numerically describe certain characteristics of the word images like projection profiles [5,11,12], gradients [11], contours [13], or geometrical characteristics [14]. Also more generic image feature descriptors have been used like *Histograms of Oriented Gradients* [15–17], *Local Binary Patterns* [17,18], or *Deep Learning* features [19], to name just a few. Regardless the features actually employed, *Dynamic Time Warping (DTW)* is the most frequently used algorithm for matching two sequences of features in KWS [12–16,19,20].

Learning-based KWS is based on statistical models that have to be trained *a priori* on a (relatively large) training set of word or character images. Many approaches of learning-based KWS are based on *Hidden Markov Models (HMM)* [6,7,21–26]. Early approaches are based on *generalised Hidden Markov Models* that are

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

https://doi.org/10.1016/j.patcog.2018.04.001
0031-3203/© 2018 Elsevier Ltd. All rights reserved.

trained on character images, i.e. images of Latin [21] or Arabic [23] characters. However, character-based approaches are negatively affected by an error-prone segmentation step [7]. More elaborated approaches rely on feature vectors of word images [22], for example by means of *Continuous-HMM* [6] or *Semi-Continuous-HMM* [6], i.e. HMMs with a shared set of *Gaussian Mixture Models*. Furthermore, the use of a *Fisher Kernel* has been employed in conjunction with HMMs in [24], while a line-based and lexicon-free HMM-approach is proposed in [7]. Recently, HMMs have been used in combination with *Bag-of-Features* [25] or *Deep Neural Networks* [26].

Further learning-based approaches are based on *Recurrent Neural Networks* [20,27], *Support Vector Machines (SVM)* [28–30], or *Latent Semantic Analysis* [31–33]. Moreover, we observe a clear shift towards *Convolutional Neural Network (CNN)* in the last years [34–38]. In most cases, CNNs are used to learn a certain word string embedding like *Pyramid Histogram of Characters (PHOC)* [35–38] or *Discrete Cosine Transform of Words* [36–38] that allows the retrieval of visual and textual queries in the same feature space.

It is known that template-based matching algorithms generally result in a lower recognition accuracy when compared to learning-based approaches. Yet, this advantage is accompanied by a loss of flexibility, which is due to the need for learning the parameters of the actual model. In particular, learning-based methods are depending on the acquisition of labelled training data by means of human experts. This is a costly and time-intensive process, especially in case of handwritten historical documents. In contrast with learning-based approaches, template-based algorithms are independent from both the actual representation formalism and the language of the underlying document. Thus, only a single instance of a keyword image is needed for the whole retrieval process.

1.1. Related work

The vast majority of KWS algorithms are based on statistical representations of words by certain numerical features (regardless whether template- or learning-based approaches are used). However, in recent years a clear tendency towards structural pattern representation formalisms can be observed in various domains [39,40]. *Structural Pattern Recognition* is based on more sophisticated data structures than feature vectors such as strings, trees, or graphs (whereby strings and trees can be seen as special cases of graphs). Graphs are, in contrast with feature vectors, flexible enough to adapt their size to the size and complexity of individual patterns. Moreover, graphs are capable to represent binary relationships that might exist in different subparts of the underlying patterns.

In the last four decades various procedures for evaluating the dissimilarity of graphs, commonly known as *graph matching*, have been proposed [41,42]. Although graphs gained noticeable attention in various fields, we observe only limited attempts where graphs have been used for the analysis and recognition of handwriting [43–45]. This is particularly interesting as graphs offer a natural and comprehensive way to represent handwritten characters or words. Moreover, in the last decade substantial progress has been made in speeding up different graph matching algorithms [42]. These facts build the main motivation of the present paper that researches the benefits of graph and template-based KWS.

A first approach to graph-based KWS has been proposed in [43], where certain keypoints are represented by nodes, while edges are used to represent strokes between these keypoints. The matching of words is then based on two separate procedures. First, assignment costs between all pairs of connected components (represented by graphs) are computed by means of a bipartite graph matching algorithm [46]. Second, optimal assignment costs between all pairs of connected components are found by means of a DTW implementation. The same matching procedure is improved by a so-called *coarse-to-fine approach* in [47].

Another framework for graph-based KWS has been introduced in [44], where nodes represent prototype strokes (so-called invariants), while edges are used to connect nodes which stem from the same connected component. The same matching procedure as in [47] is finally used for computing graph dissimilarities.

A third graph-based KWS approach has been proposed in [45], where nodes represent prototype stokes (so-called graphemes), while edges are used to represent the connectivity between graphemes. The matching is based on a coarse-to-fine approach. Formally, potential subgraphs of the query graph are determined first. These subgraphs are subsequently matched against a query graph by means of a similar graph matching procedure as used in [44,46,47].

1.2. Contribution

In the present paper we employ four novel approaches for the representation of handwritten words by means of graphs. A first approach is based on the representation of characteristic points by nodes, while edges represent strokes between these points. Another approach is based on a grid-wise segmentation of word images, where each segment is eventually represented by a node. Finally, two representation formalisms are based on vertical and horizontal segmentations of word images by means of projection profiles. For matching graphs we adopt the concept of graph edit distance which can be approximated in cubic time complexity by means of the Bipartite graph edit distance algorithm [46].

When compared to existing graph-based KWS approaches [43–45], the present approach distinguishes manifold. First, our graph representations results in a single graph for every word image. Hence, no additional assignment between graphs of different connected components is necessary during the matching process. Second, no prototype library (as used in [44,45]) is necessary for our graph representations. Thus, the risk of losing the main characteristics of handwriting is mitigated in our approach. Last but not least, besides single matchings we make use of ensemble methods [48] to combine the graph dissimilarities resulting from the the different graph representations.

The present article combines several lines of research and substantially extends three preliminary conference papers [49–51]. Moreover, in the empirical evaluation we use two additional datasets of a very recent KWS benchmark [52] and thoroughly present and discuss the evaluation of all parameters.

The remainder of this paper is organised as follows. In Sections 2 and 3, the proposed graph-based KWS approach is introduced. An experimental evaluation against template- and learning-based reference systems is given in Section 4. Finally, Section 5 concludes the paper and outlines possible future research activities.

2. Graph-based word representation

The proposed system for KWS is based on representing ancient handwritten documents by means of a set of single word images, which are in turn represented by graphs. Thus, a keyword can be retrieved in a document by matching a corresponding query graph against the complete set of document graphs. More formally, a specific graph matching algorithm computes the dissimilarities between the questioned keyword graph and all document graphs. Based on these graph dissimilarities a retrieval index can be derived. In the best case, this index represents all n instances of a given keyword as the final top-n results.

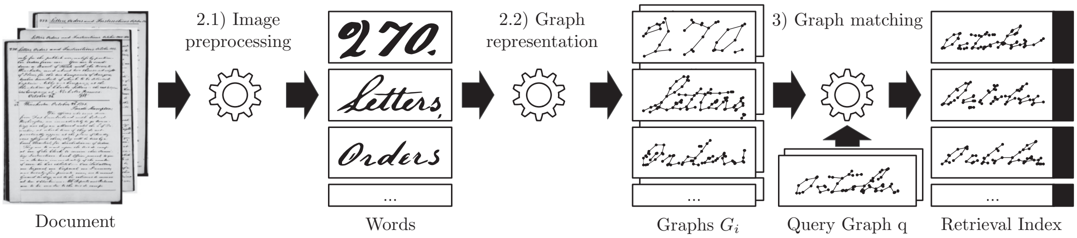


Fig. 1. Process of graph-based keyword spotting of the word “October”.

The proposed KWS system includes four basics steps (illustrated in Fig. 1). During the image preprocessing (1) the original document images are processed in order to minimise the influence of variations that are caused, for instance, by noisy backgrounds, skewed scanning, or document degradation. Subsequently, the preprocessed document images are automatically segmented into word images. On the basis of this particular word images, graphs are extracted by means of different graph extraction algorithms (2) and eventually normalised (3). Finally, the query graphs are matched with all document graphs. The resulting graph dissimilarities can either be used directly to compute the retrieval index, or they can be combined by means of different ensemble methods in order to create a retrieval index (4).

In the following subsections the first three steps are described in greater detail. In Section 2.1 we describe the image preprocessing and the word segmentation algorithm that are actually used. Section 2.2 introduces a unique arsenal of algorithms to extract graphs from word images. Finally, Section 2.3 covers the step of graph normalisation. The graph matching algorithm and ensemble methods for word graphs are thoroughly introduced in a separate section (Section 3).

2.1. Image preprocessing

Image preprocessing basically aims at reducing undesirable variations which are due to different writers (i.e. interpersonal variations), as well as due to the digitised document itself (e.g. pixel noise, skewed scanning, or general degradation of the document). In our particular case, the preprocessing is focused on the latter problem as variations in the writing style are minimised by graph normalisation in one of the next steps (see Section 2.3).

The first preprocessing step locally enhances edges by a *Difference of Gaussians* in order to address the issue of noisy background [53]. Next, document images are binarised by a global threshold. In the present paper we focus on KWS that operates on perfectly segmented word images. Thus, single word images are first automatically segmented based on projection profiles. Next, the segmentation result is manually inspected and, if necessary, manually corrected.[1] Moreover, the skew, i.e. the inclination of the document, is estimated on the lower baseline of a line of text and then corrected on single word images [14]. Finally, binarised and normalised word images are skeletonised by a 3 × 3 thinning operator [54]. We denote segmented word images that are binarised by *B*. If the image is additionally skeletonised we use the term *S* from now on.

2.2. Graph representation

On the basis of segmented, binarised, and possibly skeletonised word images, we define four graph representation formalisms that are built via graph extraction algorithms. These graph extraction algorithms aim at representing certain characteristics of a word image by means of a graph.

Table 1
Parameters of the four graph extraction algorithms.

Method	Parameter	
`Keypoint`	D = Distance threshold	
`Grid`	w = Segment width	h = Segment height
`Projection`	D_v = Vertical threshold	D_h = Horizontal threshold
`Split`	D_w = Width threshold	D_h = Height threshold

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 to be defined in the following four subsection, nodes are labelled with two-dimensional numerical labels, while edges remain unlabelled, i.e. $L_V = \mathbb{R}^2$ and $L_E = \{\}$.

In the following paragraphs four graph representation formalisms (so-called graph extraction algorithms) as well as their corresponding parameters (see Table 1) are introduced. For further details and a more thorough introduction we refer to Stauffer et al. [49].

- `Keypoint`: The first graph extraction algorithm makes use of keypoints in skeletonised word images *S* such as start, end, and junction points. 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 *D*. 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 with of binarised word images *B* into equally sized segments of width *w* and height *h*. For each segment, a node is inserted into the graph and labeled by the (*x*, *y*)-coordinates of the centre of mass of this segment. 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 [55].
- `Projection`: The next graph extraction algorithm works in a similar way as `Grid`. However, rather than a static grid this method is based 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 horizon-

[1] The present KWS approach neglects any segmentation errors and can therefore be seen as an upper-bound solution.

tal and vertical direction by means of two distance thresholds D_h and D_v, respectively. A node is inserted into the graph for each segment and labeled 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 D_w and D_h, respectively. A node is inserted into the graph and labeled by the (x, y)-coordinates of the point closest to the centre of mass of each segment. For the insertion of the edges, a similar procedure as for `Projection` is applied.

2.3. Graph normalisation

In order to improve the comparability between graphs of the same word class, the labels $\mu(v)$ of the nodes $v \in V$ are normalised. In our case the node label alphabet is defined by $L_v = \mathbb{R}^2$. We aim to reduce variations caused by position and size of the underlying word images. In particular, the (x, y)-coordinates of each node label $\mu(v) = (x, y) \in \mathbb{R}^2$ are centralised and scaled. Formally, we compute

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

where $\hat{x}$ and $\hat{y}$ denote the new node coordinates, while x and y denote the original node position. The value pairs (μ_x, μ_y) and (σ_x, σ_y) represent the means and standard deviations of all (x, y)-coordinates in the graph under consideration.

3. Matching word graphs

For spotting keywords, a query graph q (used to represent a certain keyword) is pairwise matched against all graphs $G = \{g_1, \dots, g_N\}$ stemming from the underlying document. Basically, graphs can either be matched with exact or inexact methods [41,42]. In case of exact graph matching, it is verified whether two graphs are identical (*isomorphic*) with respect to both their structure and labels. In our application graphs that represent the same word class might have (subtle) variations in their structure and their labelling making exact graph matching not feasible. Hence, we focus on inexact graph matching.

In the following three paragraphs the concept of using inexact graph matching for KWS is thoroughly discussed. In Section 3.1, the inexact graph matching paradigm of *Graph Edit Distance (GED)*, as well as a fast, yet suboptimal algorithm for this particular distance model is discussed. A selection of ensemble methods based on different graph dissimilarities is introduced in Section 3.2. Finally, Section 3.3 explains how graph dissimilarities are transformed into a retrieval index for KWS.

3.1. Inexact graph matching by means of Graph Edit Distance (GED)

Inexact graph matching allows matchings between two non-identical graphs by endowing the matching process with a certain error-tolerance with respect to labels and structure. Several approaches for inexact graph matching have been proposed [41,42]. Yet, GED is widely accepted as one of the most flexible and powerful paradigms available [56].

Given two graphs, the source graph $g_1 = (V_1, E_1, \mu_1, \nu_1)$ and the target graph $g_2 = (V_2, E_2, \mu_2, \nu_2)$, the basic idea of graph edit distance is to transform g_1 into g_2 using some edit operations. A standard set of edit operations is given by *insertions*, *deletions*, and *substitutions* of both nodes and edges. We denote the substitution of two nodes $u \in V_1$ and $v \in V_2$ by $(u \to v)$, the deletion of node $u \in V_1$ by $(u \to \varepsilon)$, and the insertion of node $v \in V_2$ by $(\varepsilon \to v)$, where ε refers to the empty node. For edge edit operations we use a similar notation. A set $\{e_1, \dots, e_k\}$ 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.

Let $\Upsilon(g_1, g_2)$ denote the set of all edit paths between two graphs g_1 and g_2. To find the most suitable edit path out of $\Upsilon(g_1, g_2)$, 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 $d_{\lambda_{\min}}(g_1, g_2)$, or $d_{\lambda_{\min}}$ for short, between $g_1 = (V_1, E_1, \mu_1, \nu_1)$ and $g_2 = (V_2, E_2, \mu_2, \nu_2)$ is defined by

$$d_{\lambda_{\min}}(g_1, g_2) = \min_{\lambda \in \Upsilon(g_1, g_2)} \sum_{e_i \in \lambda} c(e_i).$$

For our cost model we use a weighting parameter $\alpha \in [0, 1]$ that controls whether the edit operation cost on the nodes or on the edges is more important. That is, the cost of any node operation is multiplied by α. In the case of edge operations the costs are multiplied by $(1 - \alpha)$. Thus, a setting of $\alpha = 0.5$ leads to balanced importance between node and edge operation cost.

In our framework a constant cost for node deletions and insertions is defined by $c(u \to \varepsilon) = c(\varepsilon \to v) = \tau_v \in \mathbb{R}^+ (u \in V_1$ and $v \in V_2)$. The same accounts for the edges (constant cost $\tau_e \in \mathbb{R}^+$ for edge deletions and insertions). The cost for node substitutions should reflect the dissimilarity of the associated label attributes. In our application the nodes are labelled with (x, y)-coordinates and we use a weighted Euclidean distance on these labels to model the substitution cost. Formally, the cost for a node substitution $(u \to v)$ with $\mu_1(u) = (x_i, y_i)$ and $\mu_2(v) = (x_j, y_j)$ is defined by

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

where $\beta \in [0, 1]$ 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. The larger the deviation in x- or y-direction, the more important is the particular direction (and weighted accordingly).

In order to compute the graph edit distance $d_{\lambda_{\min}}(g_1, g_2)$ often A* based search techniques using some heuristics are employed [57–60]. Yet, graph edit distance computation based on A* is exponential in the number of nodes of the involved graphs. Formally, for graphs with m and n nodes we observe a time complexity of $\mathcal{O}(m^n)$. This means that for large graphs the computation of exact edit distance is intractable. In fact, graph edit distance belongs to the family of *Quadratic Assignment Problems (QAPs)* [61], which in turn belong to the class of $\mathcal{NP}$-complete problems.[2]

QAPs basically consist of a linear and a quadratic term which have to be simultaneously optimised. In case of graph edit distance, the linear term of QAPs can be used to model the sum of node edit costs, while the latter is commonly used to represent the sum of edge edit costs (see Riesen [39] for further details). The graph edit distance approximation framework introduced in [46] reduces the QAP of graph edit distance computation to an instance of a *Linear Sum Assignment Problem* (*LSAP*). Similar to QAPs, LSAPs deal with the question how the entities of two sets can be optimally assigned to each other.

For solving LSAPs, a large number of efficient algorithms exist (see Burkard et al. [62] for an exhaustive survey). The time complexity of the best performing exact algorithms for LSAPs is cubic in the size of the problem. Hence, LSAPs can be – in contrast with QAPs – quite efficiently solved.

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

Basically, the framework proposed in [46] optimally solves the LSAP which can be stated on assignments of local structures (viz. nodes and adjacent edges). This assignment can eventually be used to infer a complete set of globally consistent node and edge edit operations, i.e. we can derive a valid edit path $\lambda \in \Upsilon(g_1, g_2)$. The sum of costs of this – not necessarily optimal – edit path gives us an upper bound on the exact distance $d_{\lambda_{\min}}$ [63]. We refer to the approximated distance as d_{BP} from now on.

Finally, d_{BP} is normalised by the sum of the maximum cost edit path between the query graph q and the document word graph g, 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. In case a query consists of a set of graphs $\{q_1, \dots, q_t\}$ that represents the same keyword, the normalised graph edit distance is given by the minimal distance achieved on all t query graphs.

3.2. Ensemble methods for graphs

Rather than representing word images by a single graph representation, one might represent both the query graph q as well as the document graphs $g_i \in G$ with all graph representations as introduced in Section 2.2. Hence, a query word is actually represented by four graphs q_K, q_G, q_P, and q_S, i.e. one query graph per graph formalism (`Keypoint (K)`, `Grid (G)`, `Projection (P)`, and `Split (S)`). The same accounts for all document words which are now represented by four sets of document graphs $\{G_K, G_G, G_P, G_S\}$. Hence, rather than matching one query graph against one document graph, one could also match q_K, q_G, q_P, and q_S with the corresponding set of document graphs. Consequently, four graph dissimilarities are obtained for each pair (q, g) of a query word q and a document word g. Based on these dissimilarities, we use different combination strategies in order to build a KWS ensemble.

The first strategy considers all four graph extraction methods by either choosing the minimal, maximal, or mean GED returned on the four representations (termed $d_{\min}$, $d_{\max}$, and d_{mean} from now on). Formally, for one query word q represented by q_K, q_G, q_P, and q_S and one document word g represented by g_K, g_G, g_P, and g_S we define

$$d_{\min}(q,g) = \min_{i\in\{K,G,P,S\}} d_{\text{BP}}(q_i, g_i),$$

$$d_{\max}(q,g) = \max_{i\in\{K,G,P,S\}} d_{\text{BP}}(q_i, g_i),$$

$$d_{\text{mean}}(q,g) = \frac{1}{4}\sum_{i\in\{K,G,P,S\}} d_{\text{BP}}(q_i, g_i).$$

The second strategy only considers the two most promising individual graph extraction methods as proposed in [49], viz. `Keypoint` and `Projection`. Two different weighted sums are applied to combine the respective distances with each other (termed d_{sum_α} and $d_{\text{sum}_{\text{map}}}$ from now on). Formally,

$$d_{\text{sum}_\alpha}(q,g) = \gamma\, d_{\text{BP}}(q_K, g_K) + (1-\gamma)\, d_{\text{BP}}(q_P, g_P),$$

$$d_{\text{sum}_{\text{map}}}(q,g) = \delta\, d_{\text{BP}}(q_K, g_K) + \epsilon\, d_{\text{BP}}(q_P, g_P),$$

where γ denotes a user defined weighting factor, and δ and ϵ denote weighting factors based on the mean average precision of the individual KWS systems operating on `Keypoint` and `Projection` graphs, respectively.

The individual distances d_{BP} (on single representations) as well as all combinations can now be used to create retrieval indices for KWS (for the remainder of this section d stands for single distances or any of the five combined distances).

3.3. Computing the retrieval index

Keyword spotting is either based on a *local* or *global* threshold scenario. In a real world 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. Generally, global thresholds are regarded as the more realistic but also more difficult scenario. That is, in case of local thresholds, the KWS accuracy is independently measured for every keyword, while in case of global thresholds, the KWS accuracy is measured for every keyword with one single threshold. In the following, we introduce two retrieval indices for local and global thresholds, respectively.

First, d is used to derive a retrieval index for local thresholds by

$$r_1(q,g) = -d(q,g).$$

Second, the distance d is normalised to form a retrieval index r_2 for global thresholds by using the average distance of a query graph q to its k nearest document graphs, i.e. the document graphs $\{g_{(1)}, \dots, g_{(k)}\}$ with smallest distance values to q. Formally, we use

$$\bar{d}_k(q) = \frac{1}{k}\sum_{i=1}^{k} d(q, g_{(i)}).$$

to derive

$$\hat{d}(q,g) = \frac{d(q,g)}{\bar{d}_k(q)}.$$

Eventually, $\hat{d}$ is used to derive the second retrieval index by

$$r_2(q,g) = -\hat{d}(q,g).$$

Rather than defining k as a constant, we dynamically adapt k to every query graph q. Formally, we define k such that the distance $d(q, g_{(k)})$ of q to its kth nearest document graph $g_{(k)}$ is equal to

$$d(q, g_{(k)}) = \bar{d}_m(q) + \theta\,(\bar{d}_N(q) - \bar{d}_m(q)),$$

where $m \in \mathbb{N}$ and $\theta \in [0, 1]$ are user defined parameters and N refers to the number of document graphs. The value of $\bar{d}_m(q)$ refers to the mean distance of q to its m nearest neighbours and $\bar{d}_N(q)$ refers to the mean distance to all document graphs available. This sum reflects the level of dissimilarities of q to the graphs in its direct neighbourhood. If the sum is large, k is automatically defined large, too. This in turn increases $\bar{d}_k(q)$, which ultimately increases the scaling for $\hat{d}$.

4. Experimental evaluation

4.1. Datasets

The experimental evaluation is carried out on 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)*. In Fig. 2 small excerpts of all four documents are shown.

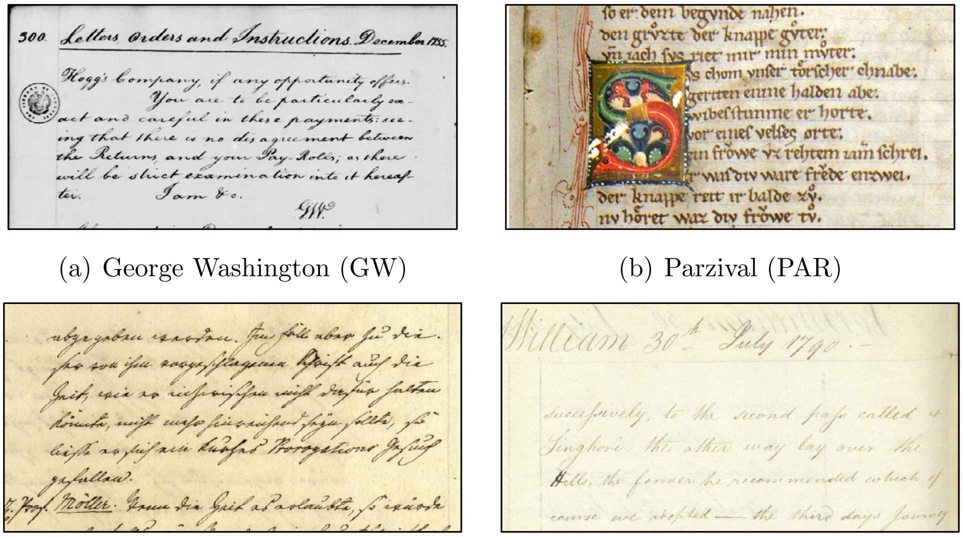
- GW consists of twenty pages stemming from handwritten letters written by George Washington and his associates during the American Revolutionary War in 1755.[6] The letters are written in English and contain only minor signs of degradation. The variations in the writing style is low, even though different writers have been involved in their creation.

[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

[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



(a) George Washington (GW)

(b) Parzival (PAR)

(c) Alvermann Konzilsprotokolle (AK)

(d) Botany (BOT)

Fig. 2. Exemplary excerpts of the two datasets.

- PAR consists of 45 pages stemming from handwritten letters written by the German poet Wolfgang Von Eschenbach in the 13th century.[7] The manuscript is written in Middle High German on parchment with markable signs of degradation. The variations in the writing are low, even though three different writers have been involved.
- AK consists of 18000 pages stemming from handwritten 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 contain only minor signs of degradation. The variations in the writing styles is rather low.
- BOT consists of more than 100 different botanical records made by the government in British India in the period of 1800 to 1850. The records are written in English and contain certain signs of degradation and especially fading. The variations in the writing style are noticeable especially with respect to scaling and intraword variations.

Note that only a subset of the datasets is used in case of BOT and AK. Moreover, the word segmentation on these datasets is imperfect [52], and thus, an additional image preprocessing step is applied to filter small connected components. In Fig. 3 three example words per dataset and their corresponding graph representations are shown.[8]

4.2. Reference systems

We compare the proposed graph-based KWS approach with two types of reference systems, viz. four different template-based systems using DTW [14–16,19], and three learning-based KWS system using SVM and CNN [28,35,36]. Note that reference results of the DTW-based systems are available on GW and PAR only. Likewise, the results of the learning-based reference systems are available on BOT and AK only.

DTW optimally aligns (warps) two sequences of features vectors $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ and $Y = \{\mathbf{y}_1, \ldots, \mathbf{y}_n\}$ along one common time axis using a dynamic programming approach. In the current case these feature vectors either consist of nine different geometrical features [14], Histogram of Oriented Gradient features [15,16], or Deep Learning features [19]. Finally, the alignment cost $d(\mathbf{x}, \mathbf{y})$ between each vector pair $(\mathbf{x}, \mathbf{y}) \in \mathbb{R}^n \times \mathbb{R}^n$ is given by the squared Euclidean distance. The DTW distance $D(X, Y)$ between two sequences of feature vectors is then given by the minimum alignment cost found by dynamic programming [5]. For speeding up the alignment a *Sakoe-Chiba band* that constrains the warping path [64] is applied.

Table 2
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	11468	6869
BOT	150	1684	3380
AK	200	1849	3734

We take three state-of-the-art learning-based methods into account, viz. CVCDAG [28], PRG [35], and QTOB [36]. CVCDAG is based on PHOC features used in conjunction with an SVM [28]. PHOC is a word string embedding approach based on five different splitting levels, i.e. on level n a word image is split into n subparts for which a histogram for the number of character occurrences is created. In PRG, the same features are used to train a CNN, the so-called PHOCNet [35]. Another CNN is used in QTOB by means of a triplet network approach [36].

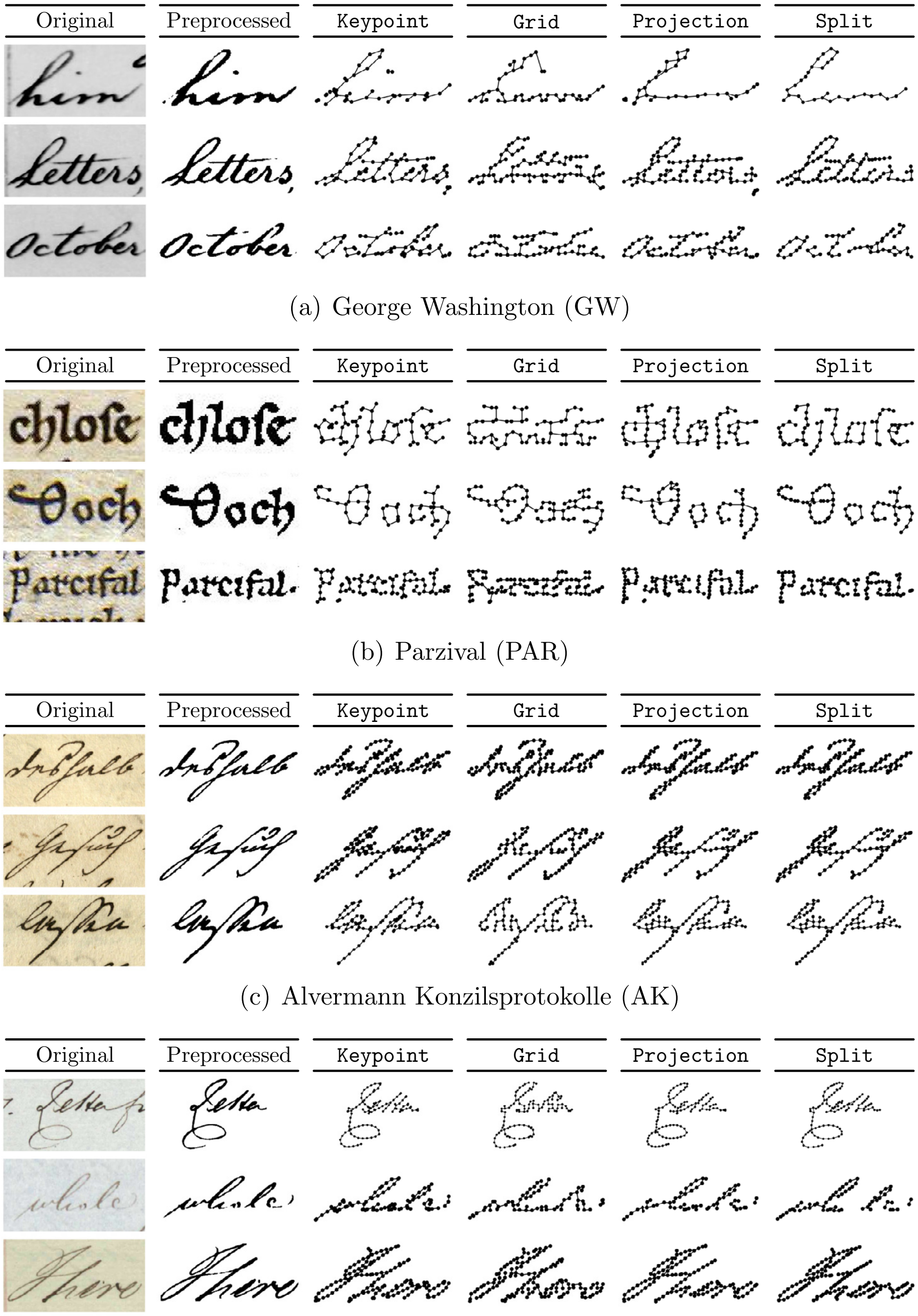
4.3. Experimental setup

All parameters are optimised on ten different keywords (with different word lengths), as shown in Fig. 4. To this end, we define a validation set that consists of 10 random instances per keyword and 900 additional random words (in total 1000 words). The details regarding the parameter optimisation are thoroughly described in Appendix A.

The optimised systems are eventually evaluated on the same training and test sets as used in [7,52] (for all datasets). All templates of a keyword present in the training set are used for KWS. In Table 2 a summary of all datasets can be found. In Section 4.4 we first compare our novel framework with the four template-based

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

[8] Graphs are available at http://www.histograph.ch



(a) George Washington (GW)

(b) Parzival (PAR)

(c) Alvermann Konzilsprotokolle (AK)

(d) Botany (BOT)

Fig. 3. Different graph representations of the four datasets.

systems on GW and PAR. Eventually, the comparison with the three learning-based systems on AK and BOT is carried out in Section 4.5.

For measuring the KWS performance of the different systems we compute the *Recall (R)* and *Precision (P)*

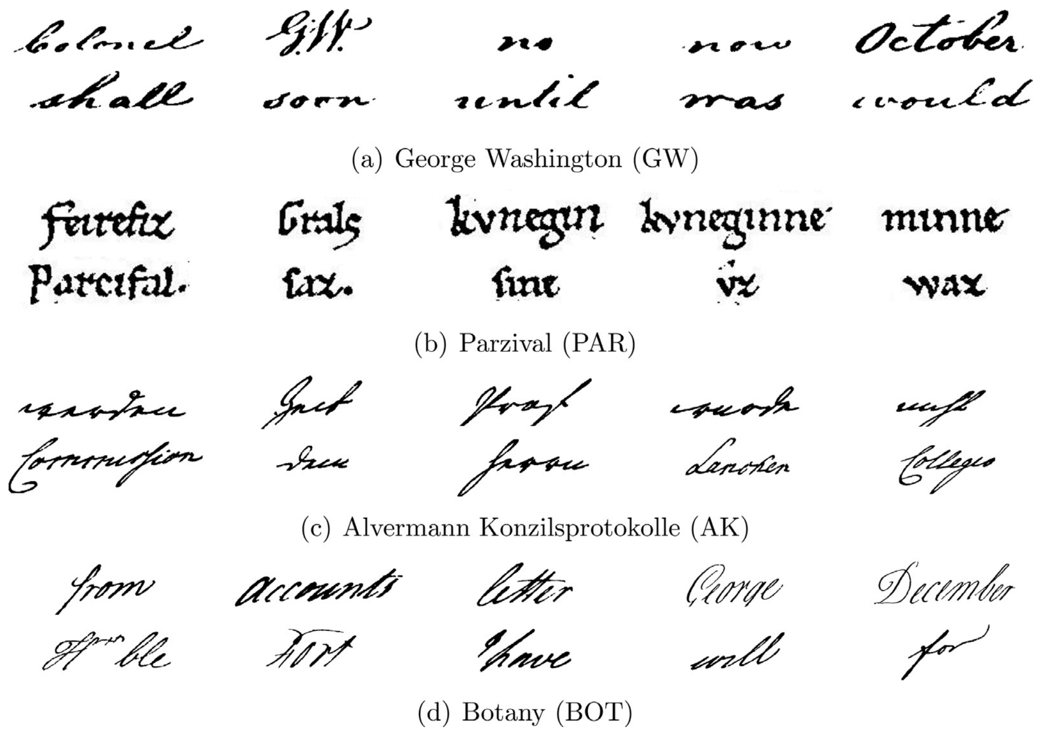
$$R = \frac{TP}{TP + FN} \text{ and } P = \frac{TP}{TP + FP},$$

which are both based on the number of *True Positives (TP)*, *False Positives (FP)*, and *False Negatives (FN)*.

Both recall and precision can be computed for two types of thresholds, viz. *local* and *global* thresholds. In the case of local thresholds, the KWS performance is measured for each keyword individually and then averaged over all keyword queries. In the case of global thresholds, the same thresholds are used for all keywords. This scenario is more practical for a real-world KWS system but requires individual keyword scores to be comparable with each other. Hence, the global threshold scenario is more challenging in general.[9]

Eventually, two metrics are used to evaluate the quality of the KWS system. For global thresholds, the *Average Precision (AP)* is measured, which is the area under the Recall-Precision curve for

[9] Global threshold results are not available for BOT and AK as the ICFHR2016 competition is based on local thresholds only.



(a) George Washington (GW)

(b) Parzival (PAR)

(c) Alvermann Konzilsprotokolle (AK)

(d) Botany (BOT)

Fig. 4. Selected keywords of the four datasets used for optimisation.

Table 3
Mean average precision (MAP) using local thresholds, average precision (AP) using a global threshold as well as their average for all graph-based KWS systems in comparison with four DTW-reference systems according to [19]. The first, second, and third best systems are indicated by (1), (2), and (3).

	GW				PAR					
Method	MAP		AP		MAP		AP		Average	
Reference (Template)										
DTW'01 [14]	45.26		33.24		46.78		50.67		43.99	
DTW'08 [15]	63.39		41.20		47.52		55.82		51.98	
DTW'09 [16]	64.80		43.76		73.49		69.10		62.79	
DTW'16 [19]	68.64		56.98	(3)	72.38		72.71	(3)	67.68	(3)
Graph (Ensemble)										
min	70.56	(1)	56.82		67.90		62.33		64.40	
max	62.58		47.94		67.57		50.59		57.17	
mean	69.16	(3)	57.11	(2)	79.38	(1)	73.77	(1)	69.85	(1)
sum_{α}	68.44		55.78		74.51	(3)	68.12		66.71	
sum_{map}	70.20	(2)	57.38	(1)	76.80	(2)	73.56	(2)	69.48	(2)

all keywords given a single (global) threshold. For local thresholds, we compute the *Mean Average Precision (MAP)*, that is the mean over the AP of each individual keyword query. The values are computed using the `trec_eval`[10] software.

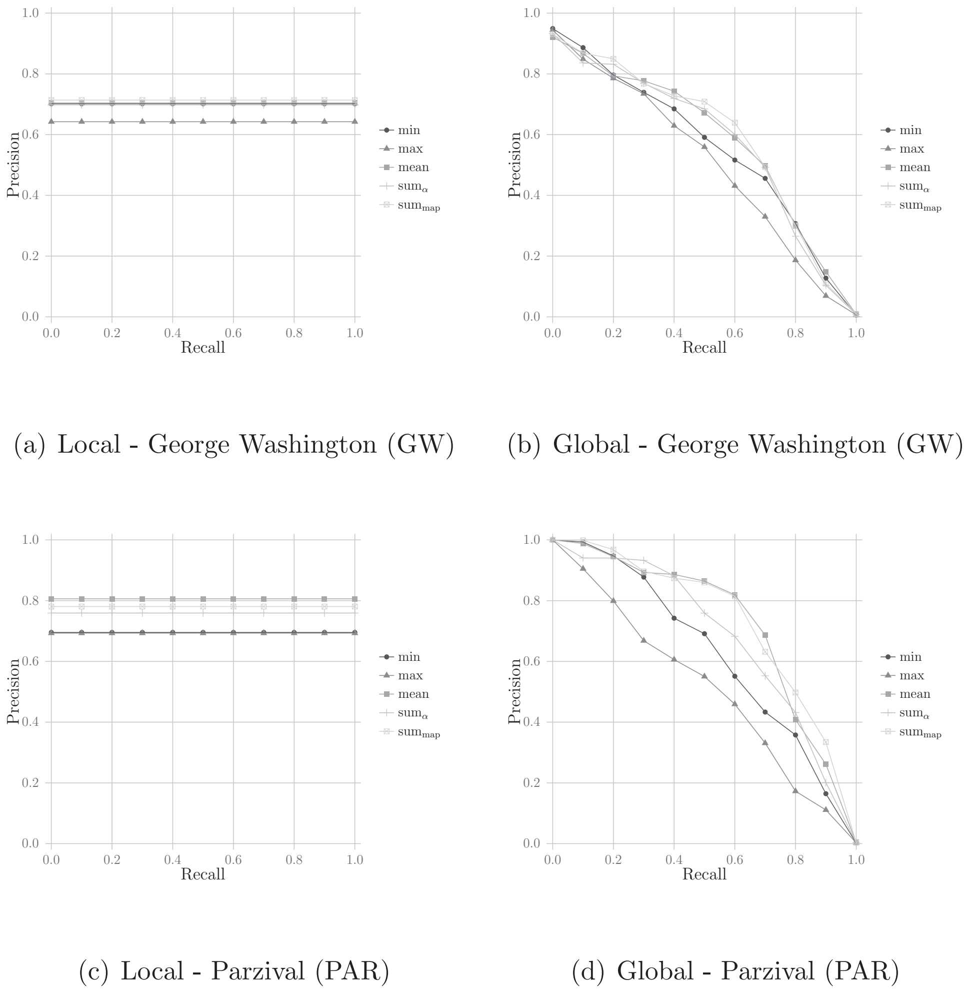
4.4. Graph-based vs. template-based KWS

The novel graph-based ensemble for KWS is compared on the independent test set with four DTW-reference systems for both benchmark datasets GW and PAR. Note that the evaluation of the single graph representations, underlying our ensemble, can be found in Appendix B. The MAP (for local thresholds), the AP (for global thresholds) as well as their average are given in Table 3. These results are also reflected in Fig. 5 where the recall-precision curves are plotted for local and global thresholds on GW and PAR, respectively.

We observe that the ensemble method mean achieves in two out of four cases the best and and in two cases the second and third best result, while sum_{map} achieves once the best result and three times the second best result. Hence, we conclude that mean and sum_{map} are the best performing ensembles. On the other hand, we observe that the ensemble max is not a well suited strategy in our specific scenario as it achieves the worst result of all ensembles in all four cases.

Overall, we see a clear performance improvement of our graph- and ensemble based approach when compared to state-of-the-art DTW-based reference systems. Especially, the ensemble methods mean and sum_{map} achieve a higher accuracy on both datasets and threshold scenarios. This is in particular interesting as two reference systems [16,19] are using advanced feature sets,[11] while our graph-based methods are based on coordinate labels only.

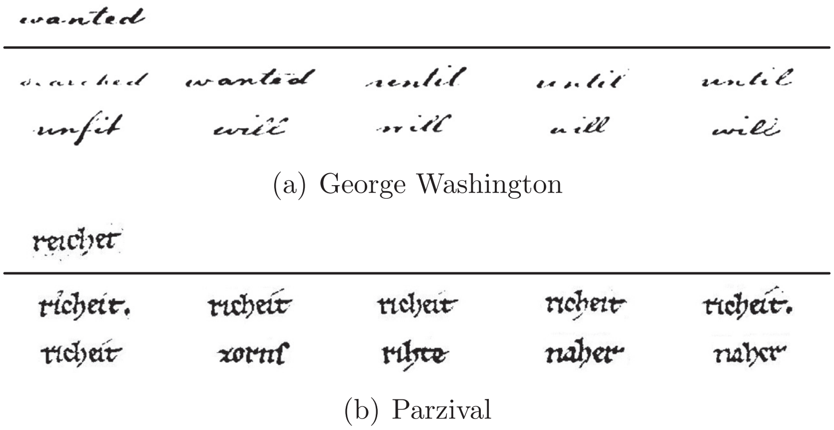
[10] http://trec.nist.gov/trec_eval



(a) Local - George Washington (GW) (b) Global - George Washington (GW)

(c) Local - Parzival (PAR) (d) Global - Parzival (PAR)

Fig. 5. Comparing the keyword spotting performance for both local and global thresholds on both datasets using recall-precision curves.



(a) George Washington

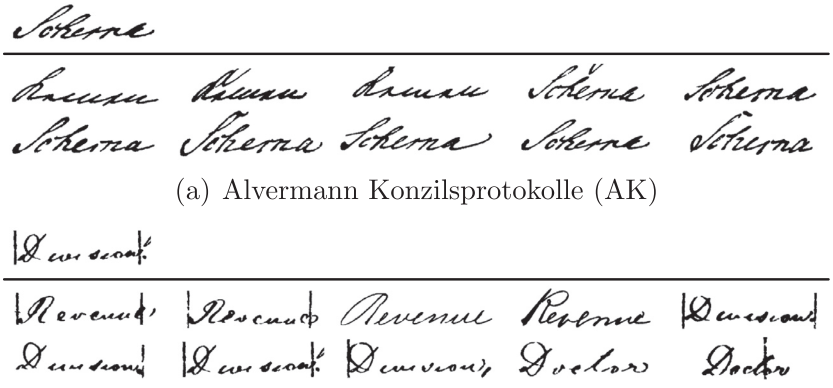
(b) Parzival

Fig. 6. Top ten results for given keywords using the min ensemble method.

A qualitative evaluation of our graph ensemble is given in Fig. 6, where the top ten results for one random keyword of each dataset is given (using the ensemble method min). On GW we observe that the first retrieved keyword (*searched*) does not correspond to the actual query word (*wanted*). Yet, the word images are visually similar to each other. The single instance available is retrieved second. Similar effects of false positives can be observed on PAR, where the first retrieved keyword does not correspond to the actual query word. Note the slight differences in orthography, which highlight a particular challenge of the medieval PAR dataset.

We provide an empirical performance evaluation on the GW dataset in Table 4. We show the average matching time per keyword for all four DTW-based systems and two graph-based systems, i.e. the proposed cubic time framework BP and a more recent quadratic time algorithm *Hausdorff Edit Distance (HED)* [65].

[11] In particular, [19] performs an unsupervised feature learning step using unlabelled data.



(a) Alvermann Konzilsprotokolle (AK)

(b) Botany (BOT)

Fig. 7. Top ten results for given keywords using the min ensemble method.

Table 4
Average matching time per keyword pair in ms.

Reference Systems	ms
DTW'01 [14]	0.7
DTW'08 [15]	5.6
DTW'09 [16]	10.2
DTW'16 [19]	7.7
Graph Matching Algorithms	
BP	303.0
HED [65]	3.2

Table 5
Mean average precision (MAP) using local thresholds for graph-based KWS systems in comparison with three state-of-the-art learning-based reference systems of the ICFHR2016 competition [52]. The first, second, and third best systems are indicated by (1), (2), and (3).

Method	AK		BOT		Average	
Reference (Learning)						
CVCDAG [28]	77.91		75.77	(2)	76.84	(2)
PRG [35]	96.05	(1)	89.69	(1)	92.87	(1)
QTOB [36]	82.15		54.95		68.55	
Graph (Ensemble)						
min	82.75		65.19		73.97	
max	82.09		67.57		74.83	
mean	84.25	(3)	68.88	(3)	76.57	
sum_{α}	84.77	(2)	68.77		76.77	(3)
sum_{map}	84.25	(3)	68.88	(3)	76.57	

Comparing the average matching time per keyword for BP with the DTW methods, we observe a clear performance loss. However, if we compare the average matching time for HED, we see lower matching times in three out of four cases. These results highlight the potential of graph matching also with a view to runtime considerations. Only DTW'01, where low dimensional feature vectors are used, is faster than HED.

4.5. Graph-based vs. learning-based KWS

Last but not least, we provide a comparison of graph-based ensemble methods[12] with recent results of learning-based approaches achieved on the ICFHR2016 benchmark dataset [52]. Note that the evaluation of the single graph representations, underlying our ensemble, can be found in Appendix B. In Table 5, the MAP on both datasets as well as their average is given for all graph-based ensemble methods and the learning-based reference systems. In case of graph-based ensemble methods, we observe that the ensemble methods mean, sum_{α}, and sum_{map} outperform min and max. In case of the learning-based reference systems, we observe that the PHOCNet leads to an astonishing accuracy. However, the ensemble method sum_{α} achieves almost the same overall KWS accuracy as the learning-based PRG-approach. In particular, on the AK dataset our graph-based methods can keep up or even outperform PRG.

We conclude that our graph-based methods can keep up with most state-of-the-art learning-based methods. This is especially remarkable as these reference methods are based on more advanced features than our approach (e.g. PHOC), and use learning-based algorithms (i.e. SVM, and CNN). In particular, 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 Pratikakis et al. [52] for details). Yet, the manual labelling of historical handwriting with the help of human experts is a labour- and cost-intensive process. This makes our novel graph-based methods especially valuable as only a single instance of a keyword is required.

To conclude this section, we provide a qualitative comparison in Fig. 7, where the top ten results for one random keyword of each dataset is given (using the ensemble method min). On the AK dataset, we observe false positives on the first three retrieval results for the query *Schema*. However, all false positives are visually very similar to the actual query word. On the BOT dataset, a similar effect can be observed for the query *Division* and the false positives *Revenue*. Note the additional challenge of KWS due to the imperfect word segmentation.

5. Conclusion and future work

A novel graph-based framework is presented for the task of keyword spotting in historical handwritten documents. The proposed approach allows to search arbitrary keywords in handwritten documents, which is particularly interesting for historical documents where an automatic transcription is not feasible or not sufficiently accurate. The basic algorithm for keyword spotting is template-based and can thus be used without a priori learning of a statistical model. Hence, no labour- and time-intensive labelling of historical handwriting is necessary. This makes the presented graph-based framework not only more universally applicable but also more flexible when compared to learning-based approaches.

To represent individual words we make use of four different graph extraction methods. The first extraction method is based on keypoints detected in a word image, which are eventually repre-

[12] For this comparison, we only consider the two most promising graph representations `Keypoint` and `Projection`.

sented by nodes, while the edges are represent the line strokes between two keypoints. The second extraction method is based on grid-wise segments of a word image, which are eventually represented by nodes, while the edges model a minimal spanning tree on the adjacency structure of these segments. The third and fourth extraction method make both use of vertical and horizontal projection profiles to segment a word image. The resulting segments are then represented by nodes, while edges are inserted between nodes based on the line strokes between segments. The actual keyword spotting is then based on combined graph dissimilarities returned by bipartite graph matching.

For the experimental evaluation the proposed method is first compared with four state-of-the-art template-based DTW reference systems on two well-known benchmark datasets. For both datasets the task of keyword spotting is evaluated using both local and global thresholds. On both datasets and thresholds the novel graph-based methods clearly outperform all of the reference systems. Moreover, our novel graph-based ensemble method can keep up or even outperform several learning-based approaches on the very recent ICFHR2016 benchmark datasets. This indicates the power and flexibility of graphs in conjunction with template- and ensemble-based KWS.

A limitation of the proposed approach is the need for a segmentation of document pages into single word images. In future work we therefore aim at extending our word-based approach to a line-based approach, where a query graph can be found in a line (represented as a line graph) by means of error-tolerant subgraph isomorphism. Moreover, the current graph representations and edit functions are based on the Euclidean space $\mathbb{R}^2$. More meaningful label functions for both nodes and edges (based on texture descriptors for instance) could further improve the performance of our keyword spotting framework. Finally, the performance of the novel framework could be improved w.r.t. runtime by graph matching algorithms with a lower computational complexity [65] or other heuristics [66,67].

Acknowledgments

This work has been supported by the Hasler Foundation Switzerland.

Appendix A. Optimisation of the parameters

The optimisation of the parameters is conducted in four subsequent steps. The first two steps are actually used for optimising the systems based on local thresholds, while the third step is solely used for global thresholds. Note that global thresholds are optimised for GW and PAR only, as the ICFHR2016 benchmark (i.e. AK and BOT) is based on local thresholds only. Finally, the fourth step is used for the ensemble methods only.

First step: The parameters for each graph extraction algorithm are optimised with respect to the MAP on the validation set using different node and edge deletion/insertion costs $\tau_v = \tau_e = \{1, 4, 8, 16, 32\}$, fixed weighting parameters $\alpha = \beta = 0.5$ and retrieval index r_1. In Table A.6, an overview of the tested parameters is given for all four datasets. The best performing parameters are marked with an asterisk.

Second step: Using the optimal parameters for each graph extraction method, the parameters for graph edit distance are further optimised. That is, we evaluate the 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 (resulting in 5000 settings in total). In Table A.7 the optimal cost function parameters are given for all graph extraction algorithms.

Third step: For KWS systems using global thresholds we employ retrieval index r_2 rather than r_1. Hence, the parameter m and threshold scaling factor θ are individually optimised for each system (N is defined by the number of document graphs). We tested 10000 parameters pairs $(m, \ \theta)$ with $m \in \{10, 20, \ldots, 990, 1000\}$ and $\theta \in \{0.01, 0.02, \ldots, 0.99, 1.00\}$. In Table A.8, the optimal parameter settings for r_2 are given for all systems.

The differences of the optimal parameter settings are due to different distributions of the graph edit distances for GW and PAR, respectively. In case of GW, the graph edit distances for global thresholds are optimised by considering a rather large neighbourhood m and small weighting factor θ. In case of PAR, we can observe the opposite case.

Table A.6
Parameter optimisation for the graph extraction algorithms during the first step of optimisation. Optimal parameters are marked with an asterisk.

	Method	Parameter				
GW	Keypoint	D	= {4*, 6,8,10,12}			
	Grid	w	= {6*, 8,10,12,14}	×	h	= {6*, 8,10,12}
	Projection	D_v	= {4*, 6,8,10}	×	D_h	= {4,6*, 8,10}
	Split	D_w	= {4,6*, 8,10}	×	D_h	= {4*, 6,8,10}
PAR	Keypoint	D	= {2*, 4,6,8,10,12}			
	Grid	w	= {4*, 6,8,10,12}	×	h	= {4*, 6,8,10}
	Projection	D_v	= {2*, 4,6,8,10}	×	D_h	= {4*, 6,8,10}
	Split	D_w	= {2*, 4,6,8,10}	×	D_h	= {4*, 6,8,10}
AK	Keypoint	D	= {16*, 20,24,28,32,36}			
	Projection	D_v	= {10,12,14,16*}	×	D_h	= {10*, 12,14,16}
BOT	Keypoint	D	= {16*, 20,24,28,32,36}			
	Projection	D_v	= {10,12,14*, 16}	×	D_h	= {10,12*, 14,16}

Table A.7
Optimal cost function parameter for graph edit distance computation.

	GW				PAR				AK				BOT			
Method	τ_v	τ_e	α	β	τ_v	τ_e	α	β	τ_v	τ_e	α	β	τ_v	τ_e	α	β
Keypoint	4	1	0.5	0.1	4	4	0.5	0.3	16	16	0.5	0.1	32	32	0.3	0.1
Grid	4	1	0.7	0.1	4	1	0.7	0.5	-	-	-	-	-	-	-	-
Projection	4	1	0.5	0.1	4	1	0.5	0.5	8	32	0.7	0.1	8	32	0.9	0.3
Split	4	1	0.5	0.1	4	1	0.3	0.3	-	-	-	-	-	-	-	-

Table A.8
Optimal m and θ for retrieval index r_2.

	GW		PAR	
Method	m	θ	m	θ
Keypoint	60	0.02	1000	0.20
Grid	90	0.01	820	0.19
Projection	60	0.03	980	0.20
Split	80	0.03	990	0.18
min	70	0.08	10	0.72
max	40	0.10	10	0.61
mean	60	0.06	10	0.64
sum$_\alpha$	70	0.04	10	0.61
sum$_{map}$	90	0.02	10	0.61

Table A.9
Optimal γ for the sum$_\alpha$ ensemble.

	GW	PAR	AK	BOT
γ	MAP	MAP	MAP	MAP
0.1	73.21	**100.00**	30.65	51.20
0.2	73.34	99.19	**30.74**	51.38
0.3	**75.23**	99.19	30.25	51.56
0.4	71.84	99.19	29.74	**51.74**
0.5	71.75	99.19	29.12	50.06
0.6	71.34	96.33	29.49	50.50
0.7	72.00	94.65	29.91	49.73
0.8	72.10	94.28	28.59	46.49
0.9	72.05	93.95	27.91	46.71

Fourth step: The weighting factor $\gamma \in \{0.1, \ldots, 0.9\}$ for the ensemble sum$_\alpha$ is the sole parameter that needs to be optimised (all other ensemble strategies need no parameter tuning).

In Table A.9, the MAP is given for the tested parameter settings for γ on all benchmark datasets. Note that the best performing parameter setting is indicated in bold face.

Appendix B. Results graph-based representations

We compare the novel graph representations with each other in Table B.10. We observe that either Keypoint or Projection achieve the best results among all graph extraction methods.

Table B.10
Mean average precision for all graph-based KWS systems. The first, second, and third best systems are indicated by (1), (2), and (3).

	GW		PAR		AK		BOT	
Keypoint	66.08	(1)	62.04	(2)	77.24	(1)	45.06	(2)
Grid	60.02		56.50		-		-	
Projection	61.43	(2)	66.23	(1)	76.02	(2)	49.57	(1)
Split	60.23	(3)	59.44	(3)	-		-	

References

[1] R. Plamondon, S. Srihari, Online and off-line handwriting recognition: a comprehensive survey, IEEE Trans. Pattern Anal. Mach.Intell. 22 (1) (2000) 63–84.
[2] A. Antonacopoulos, A.C. Downton, Special issue on the analysis of historical documents, Int. J. Doc. Anal.Recognit. 9 (2-4) (2007) 75–77.
[3] H. Bunke, T. Varga, Off-line roman cursive handwriting recognition, in: Digital Document Processing, 2007, pp. 165–183.
[4] R. Manmatha, Chengfeng Han, E. Riseman, Word spotting: a new approach to indexing handwriting, in: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1996, pp. 631–637.
[5] T.M. Rath, R. Manmatha, Word image matching using dynamic time warping, in: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2, 2003.
[6] J.A. Rodríguez-Serrano, F. Perronnin, Handwritten word-spotting using hidden Markov models and universal vocabularies, Pattern Recognit. 42 (9) (2009) 2106–2116.
[7] A. Fischer, A. Keller, V. Frinken, H. Bunke, Lexicon-free handwritten word spotting using character HMMs, Pattern Recognit. Lett. 33 (7) (2012) 934–942.
[8] R. Rose, D. Paul, A hidden Markov model based keyword recognition system, in: IEEE International Conference on Acoustics, Speech, and Signal Processing, 1990, pp. 129–132.
[9] S.-S. Kuo, O.E. Agazzi, Keyword spotting in poorly printed documents using pseudo 2-D hidden Markov models, IEEE Trans. Pattern Anal. Mach.Intell. 16 (8) (1994) 842–848.
[10] G.L. Scott, H.C. Longuet-Higgins, An algorithm for associating the features of two images, Proc. R. Soc. B 244 (1309) (1991) 21–26.
[11] B. Zhang, S.N. Srihari, C. Huang, Word image retrieval using binary features, in: Document Recognition and Retrieval, 2003, pp. 45–54.
[12] T.M. Rath, R. Manmatha, Word spotting for historical documents, Int. J. Doc. Anal.Recognit. 9 (2-4) (2007) 139–152.
[13] T. Adamek, N.E. O'Connor, A.F. Smeaton, Word matching using single closed contours for indexing handwritten historical documents, Int. J. Doc. Anal.Recognit. 9 (2-4) (2006) 153–165.
[14] 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 (01) (2001) 65–90.
[15] 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.
[16] 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.
[17] A. Silberpfennig, L. Wolf, N. Dershowitz, S. Bhagesh, B.B. Chaudhuri, Improving OCR for an under-resourced script using unsupervised word-spotting, in: International Conference on Document Analysis and Recognition, 2015, pp. 706–710.
[18] S. Dey, A. Nicolaou, J. Lladós, U. Pal, Local binary pattern for word spotting in handwritten historical document, in: International Workshop on Structural, Syntactic, and Statistical Pattern Recognition, 2016, pp. 574–583.
[19] B. Wicht, A. Fischer, J. Hennebert, Deep learning features for handwritten keyword spotting, in: International Conference on Pattern Recognition, 2016, pp. 3434–3439.
[20] 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.
[21] J. Edwards, Y.W. Teh, R. Bock, M. Maire, G. Vesom, D.A. Forsyth, Making latin manuscripts searchable using gHMM's, in: International Conference on Neural Information Processing Systems, 17, 2004, pp. 385–392.
[22] V. Lavrenko, T. Rath, R. Manmatha, Holistic word recognition for handwritten historical documents, in: International Workshop on Document Image Analysis for Libraries, 2004, pp. 278–287.
[23] J. Chan, C. Ziftci, D. Forsyth, Searching off-line arabic documents, in: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2, 2006, pp. 1455–1462.
[24] F. Perronnin, J.A. Rodríguez-Serrano, Fisher kernels for handwritten word-spotting, in: International Conference on Document Analysis and Recognition, 2009, pp. 106–110.
[25] L. Rothacker, M. Rusiñol, G.a. Fink, Bag-of-features HMMs for segmentation-free word spotting in handwritten documents, in: International Conference on Document Analysis and Recognition, 2013, pp. 1305–1309.
[26] 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 (4) (2014) 1003–1015.
[27] V. Frinken, A. Fischer, M. Baumgartner, H. Bunke, Keyword spotting for self–training of BLSTM NN based handwriting recognition systems, Pattern Recognit. 47 (3) (2014) 1073–1082.
[28] J. Almazán, A. Gordo, A. Fornés, E. Valveny, Word spotting and recognition with embedded attributes, IEEE Trans. Pattern Anal. Mach.Intell. 36 (12) (2014) 2552–2566.
[29] J. Almazán, A. Gordo, A. Fornés, E. Valveny, Segmentation-free word spotting with exemplar SVMs, Pattern Recognit. 47 (12) (2014) 3967–3978.
[30] M. Khayyat, L. Lam, C.Y. Suen, Learning-based word spotting system for Arabic handwritten documents, Pattern Recognit. 47 (3) (2014) 1021–1030.
[31] M. Rusiñol, D. Aldavert, R. Toledo, J. Lladós, Browsing heterogeneous document collections by a segmentation-free word spotting method, in: International Conference on Document Analysis and Recognition, 2011, pp. 63–67.
[32] D. Aldavert, M. Rusiñol, R. Toledo, J. Lladós, Integrating visual and textual cues for query-by-string word spotting, in: International Conference on Document Analysis and Recognition, 2013, pp. 511–515.
[33] M. Rusiñol, D. Aldavert, R. Toledo, J. Lladós, Efficient segmentation-free keyword spotting in historical document collections, Pattern Recognit. 48 (2) (2015) 545–555.
[34] A. Sharma, Pramod Sankar K., Adapting off-the-shelf CNNs for word spotting & recognition, in: International Conference on Document Analysis and Recognition, 2015, pp. 986–990.
[35] S. Sudholt, G.A. Fink, PHOCNet: adeep convolutional neural network for word spotting in handwritten documents, in: International Conference on Frontiers in Handwriting Recognition, 2016, pp. 277–282.
[36] 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.
[37] S. Sudholt, G.A. Fink, Evaluating word string embeddings and loss functions for CNN-based word spotting, in: International Conference on Document Analysis and Recognition, 2017, pp. 493–498.

[38] G. Lluis, M. Rusiñol, D. Karatzas, LSDE : Levenshtein space deep embedding for query-by-string word spotting, in: International Conference on Document Analysis and Recognition, 2017, pp. 499–504.
[39] K. Riesen, Structural pattern recognition with graph edit distance, Adv. Comput.Vis. Pattern Recognit. (2015).
[40] M. Stauffer, T. Tschachtli, A. Fischer, K. Riesen, A survey on applications of bipartite graph edit distance, in: International Workshop on Graph-Based Representations in Pattern Recognition, 2017, pp. 242–252.
[41] 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.
[42] 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.
[43] P. Wang, V. Eglin, C. Garcia, C. Largeron, J. Lladós, A. Fornés, A novel learning-free word spotting approach based on graph representation, in: International Workshop on Document Analysis Systems, 2014, pp. 207–211.
[44] 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.
[45] P. Riba, J. Lladós, A. Fornés, Handwritten word spotting by inexact matching of grapheme graphs, in: International Conference on Document Analysis and Recognition, 2015, pp. 781–785.
[46] K. Riesen, H. Bunke, Approximate graph edit distance computation by means of bipartite graph matching, Image Vis. Comput. 27 (7) (2009) 950–959.
[47] 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: International Conference on Pattern Recognition, 2014, pp. 3074–3079.
[48] L.I. Kuncheva, Combining Pattern Classifiers, 2004.
[49] 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.
[50] 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.
[51] 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.
[52] I. Pratikakis, K. Zagoris, B. Gatos, J. Puigcerver, A.H. Toselli, E. Vidal, ICFHR2016 handwritten keyword spotting competition (H-KWS 2016), in: International Conference on Frontiers in Handwriting Recognition, 2016, pp. 613–618.
[53] 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.
[54] Z. Guo, R.W. Hall, Parallel thinning with two-subiteration algorithms, Commun. ACM 32 (3) (1989) 359–373.
[55] J.B. Kruskal, On the shortest spanning subtree of a graph and the traveling salesman problem, Am. Math. Soc. 7 (1) (1956). 48–48
[56] H. Bunke, G. Allermann, Inexact graph matching for structural pattern recognition, Pattern Recognit. Lett. 1 (4) (1983) 245–253.
[57] A. Dumay, R. van der Geest, J. Gerbrands, E. Jansen, J. Reiber, Consistent inexact graph matching applied to labelling coronary segments in arteriograms, in: International Conference on Pattern Recognition, 1992, pp. 439–442.
[58] L. Gregory, J. Kittler, Using graph search techniques for contextual colour retrieval, in: International Workshop on Structural, Syntactic, and Statistical Pattern Recognition, 2002, pp. 186–194.
[59] 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 (10) (2001) 1089–1105.
[60] S. Fankhauser, K. Riesen, H. Bunke, Speeding up graph edit distance computation through fast bipartite matching, in: International Workshop on Graph-Based Representations in Pattern Recognition, 2011, pp. 102–111.
[61] T.C. Koopmans, M. Beckmann, Assignment problems and the location of economic activities, Econometrica 25 (1) (1957) 53.
[62] R. Burkard, M. Dell'Amico, S. Martello, Assignment Problems, 2009.
[63] K. Riesen, A. Fischer, H. Bunke, Estimating graph edit distance using lower and upper bounds of bipartite approximations, Int. J. Pattern Recognit.Artif. Intell. 29 (02) (2015) 1550011.
[64] H. Sakoe, S. Chiba, Dynamic programming algorithm optimization for spoken word recognition, IEEE Trans. Acoust. Speech, Signal Process. 26 (1) (1978) 43–49.
[65] M.R. Ameri, M. Stauffer, K. Riesen, T.D. Bui, A. Fischer, Keyword spotting in historical documents based on handwriting graphs and hausdorff edit distance, in: International Graphonomics Society Conference, 2017.
[66] M. Stauffer, A. Fischer, K. Riesen, Speeding-up graph-based keyword spotting in historical handwritten documents, in: International Workshop on Graph-Based Representations in Pattern Recognition, 2017, pp. 83–93.
[67] M. Stauffer, A. Fischer, K. Riesen, Speeding-up graph-based keyword spotting by quadtree segmentations, in: International Conference on Computer Analysis of Images and Patterns, 2017.

Michael Stauffer has received a BSc and MSc in Business Information Systems from the University of Applied Sciences and Arts Northwestern Switzerland in 2012 and 2015, respectively. He is currently a PhD student at the same institution in cooperation with the University of Pretoria. His research interests are pattern recognition with a special focus on graph matching and graph representations.

Andreas Fischer received his MSc and PhD degrees in Computer Science from the University of Bern in 2008 and 2012, respectively. He then conducted postdoctoral research at CENPARMI and the Ecole Polytechnique de Montreal. Since 2016, Andreas Fischer is an Assistant Professor at the University of Applied Sciences Western Switzerland and a lecturer at the University of Fribourg. He has more than 60 publications related to historical document analysis, handwriting recognition, keyword spotting, signature verification, hidden Markov models, recurrent neural networks, and graph matching for structural pattern recognition.

Kaspar Riesen received his MSc and PhD degrees in Computer Science from the University of Bern in 2006 and 2009, respectively. His PhD thesis received the Alumni price for an outstanding work at the Institute of Computer Science and Applied Mathematics of the University of Bern. His research interests cover the fields of artificial intelligence, pattern recognition, machine learning and data mining. In particular, he is working on the development of novel algorithms for solving graph matching problems in various domains of intelligent information processing.