IJDAR
DOI 10.1007/s10032-010-0146-0

ORIGINAL PAPER

Text retrieval from early printed books

Simone Marinai

Received: 8 February 2010 / Revised: 27 November 2010 / Accepted: 10 December 2010

Abstract Retrieving text from early printed books is particularly difficult because in these documents, the words are very close one to the other and, similarly to medieval manuscripts, there is a large use of ligatures and abbreviations. To address these problems, we propose a word indexing and retrieval technique that does not require word segmentation and is tolerant to errors in character segmentation. Two main principles characterize the approach. First, characters are identified in the pages and clustered with self-organizing map (SOM). During the retrieval, the similarity of characters is estimated considering the proximity of cluster centroids in the SOM space, rather than directly comparing the character images. Second, query words are matched with the indexed sequence of characters by means of a dynamic time warping (DTW)-based approach. The proposed technique integrates the SOM similarity and the information about the width of characters in the string matching process. The best path in the DTW array is identified considering the widths of matching words with respect to the query so as to deal with broken or touching symbols. The proposed method is tested on four copies of the Gutenberg Bibles.

Keywords Early printed books · Dynamic Time Warping · Self-Organizing Map

1 Introduction

In the infancy of the printing technology, books were designed to emulate medieval manuscripts, and therefore, it is not surprising that early printed books are among the most difficult printed documents to be recognized by machine reading systems. For this reason, incunabula (books printed before the year 1501) share with medieval manuscripts the practice of using hand-painted initial capital letters, ligatures, and abbreviations. The latter were employed in manuscripts to justify the words in text-lines and have been slowly abandoned with the subsequent improvements in the printing technology. Ligatures appear in later works, such as the Trévoux dictionary of the seventeenth century [1], were frequent in nineteenth century books and are still occasionally used, for instance in the 'fi' ligature that is a different symbol with respect to two juxtaposed characters 'f' and 'i'.

The Gutenberg Bible, printed in the 1450s, is the first complete book extant in the West and the earliest printed from movable type [2]. Gutenberg probably printed some other books in the same years, even if there is no evidence of this. The original print run of the Bible is unknown, but with about 40 copies still in existence, this work is regarded as the first book in the West with a relatively large circulation.[1] Unlike modern printed books, the copies of the Gutenberg Bible are not exact reproductions of each other. Illuminated initial capital letters were painted by hand after printing the main text, and therefore, these are different in each copy. There are also differences in the arrangement of the words in the lines as we will discuss in Sect. 5.2. Even the number of lines in each page can change: most pages are printed on two columns with 42 lines, but the first pages of some copies have 40 lines.

Since the Gutenberg Bible is a milestone in the printing technology, some copies held by major Libraries around the world have been digitized and made available on the Internet [3]. In particular, in our experiments, we used four copies that can be freely downloaded from three Digital

S. Marinai (✉)
Dipartimento di Sistemi e Informatica,
Università di Firenze, Firenze, Italy
e-mail: simone.marinai@unifi.it

[1] Movable type in China and Korea predates these books by several centuries, but printing became mechanized in Europe in the mid-1400.

Published online: 04 January 2011

Libraries (additional details are reported in Sect. 5). The Gutenberg Bibles are among the most studied Renaissance documents,and therefore, an automatic transcription of their textual contents is probably of little interest for scholars in the Humanities. On the other hand, we believe that these volumes can be regarded as a benchmark to be used to compare different algorithms dealing with historical documents. This is one of the main motivations for using these data in our experiments. The proposed techniques can be used also to search for word similarities in incunabula so as to discover spelling variations in these documents. Related applications are addressed in [4] and in [5] where methods to discover variations in typefaces in early printed books are described.

One way to perform word image retrieval (in Sect. 3 we report an analysis of other techniques) is based on the recognition of the whole textual content of the indexed documents using supervised classifiers with techniques related to OCR systems. The extensive use of ligatures and abbreviations restricts the applicability of recognition-based approaches because a large (and possibly partially unknown) set of symbols should be considered by the classifier. Furthermore, a suitable dictionary listing all the possible forms for a given lemma cannot be easily built. In the proposed text retrieval tool, we do not attempt to recognize the whole text, but we look for occurrences of query words with a query by example approach that is partially tolerant to ligatures and abbreviations. Spelling variations in the Gutenberg Bibles have been already studied by scholar in the Humanities (Sect. 2). However, since we aim at processing other early printed books (where different ligatures and abbreviations might be used), we cannot adopt recognition-based techniques.

This paper is organized as follows. The main features of the incunabula and of the Gutenberg Bibles are summarized in Sect. 2. In Sect. 3, we discuss some recent work related to the proposed technique that is described in details in Sect. 4. The data set and the experiments that we performed are described in Sect. 5. Some conclusions are presented in Sect. 6.

2 Early printed books

The incunabula are some of the most rare printed works due to their very limited print run. The Digital Library of the French National Library [6] lists more than 1,800 books printed between 1,450 and 1,500 that are a mere 0.8% of the whole collection. These works are in several cases printed with fonts difficult to recognize by OCR packages that are tuned on modern typefaces. In addition to the use of old fonts, incunabula show a significant variance of spelling for many words. Words' variants have been used for several centuries in many countries [7] bringing to a progressive standardization of

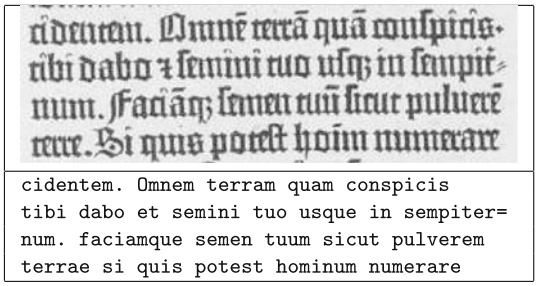


Fig. 1 Fragment of one Genesis page [40] with the corresponding text transcription at the bottom

spelling. Some examples are reported in [4] for the works by Montaigne in the sixteenth century.

The text recognition in the Gutenberg Bible is difficult for three main reasons. First, the images available on the Internet are stored as JPG files with a size of 965×1390 pixels[2] (the average character size is 10×15 pixels). Second, there are typical problems caused by the book aging such as support deterioration, bleed through, and see through. Third, the cost of the printing supports (in particular vellum) was very high, and every possible strategy was adopted to compress the text. To this purpose, many abbreviations and ligatures are used, and there is a very limited space between words that is frequently comparable to the space between characters in the same word. The latter features influence the text reading by humans as well, and therefore, it is really difficult to read the text without the help of a side-by-side transcription.

Some copies of the Gutenberg Bibles have been digitized in recent years and made available on the Internet for free browsing and download. To illustrate the difficulties of the automatic recognition of the Gutenberg Bible, we report in Fig. 1 one text fragment with the corresponding transcription. In the image, some ligatures and abbreviations can be noticed. For instance, "Omnem terram quam" is printed as "Omnē terrā quā" in the first line; "hominum" is printed as "hoīm" in the last line. The Gutenberg Bible contains more than 75 types of ligatures, and 15 of them have two or more meanings [9]. For instance, (Fig. 2) ē can correspond either to "em", "en", or "est". The letters in the Gutenberg Bibles are in *black letter* (or Gothic script) that is based on the manuscript styles of the time. When digitized, many characters look very similar one to each other (Fig. 3).

The layout, including the use of hand-painted illuminated letters, is influenced by manuscripts as well. The text-lines form a straight edge on both sides of the columns. Only some signs such as periods and hyphens (probably hand drawn after printing) extend beyond the right border, whereas only few painted capitals go outside the left border. Nowadays, text-lines are justified by carefully adding white spaces between

[2] These values are related to the Göttingen copy [8], the other collections that we used in our experiments have very similar features.

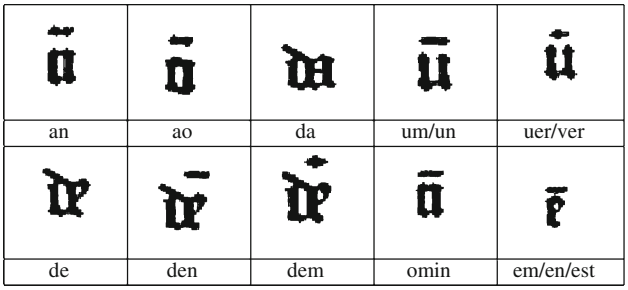


Fig. 2 Examples of ligatures and abbreviations used in the Gutenberg Bible. Below each image, we report the corresponding meaning (in some cases more than one)

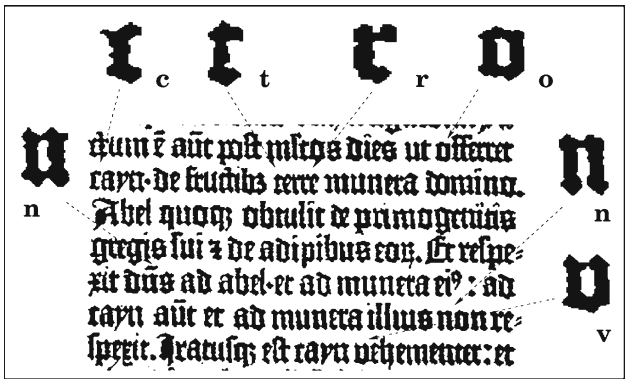


Fig. 3 Examples of similar characters corresponding to different alphabet classes [8]

words in the line until the rightmost character is aligned with the rest of the paragraph. In early printed works, the space between words was not increased. On the opposite, the justification was achieved by reducing the width of some words in text-lines. To this purpose, some isolated symbols had multiple designs (with different widths) and ligatures and abbreviations were widely used [10]. Taking into account both ligatures and abbreviations, there are 290 different "characters" in the Gutenberg Bible. By using variable combinations of these symbols, it was possible to perfectly justify the text without considering variable size spacing. Another interesting feature is that, similarly to manuscripts, the distance between lines is low, and ascenders and descenders are very close to neighboring lines. From the automatic recognition point of view, the most critical characteristic is the very limited space between contiguous words in the same line (Fig. 4). To deal with this problem, the method described in Sect. 4 performs the word indexing and retrieval without segmenting the words.

Fig. 4 Two occurrences of the word "abram". The space separating neighboring words is very small

The Gutenberg Bible has been already used as a test bed for preprocessing algorithms designed to work on historical documents. For instance, [11] describes an approach for pixel classification used for document image enhancement. The binarization of historical documents is described also in [12]. Other papers dealing with early printed books addressed the retrieval of ornamental initials [13,14].

3 Previous work

Text retrieval methods dealing with either printed or handwritten documents have been widely studied in the last years. The most intuitive way to approach the text retrieval is based on the text search in a transcription of the indexed works that can be obtained either with a manual annotation or with an automatic recognition. When applied to printed documents, this recognition-based approach can rely on a suitable use of optical character recognition (OCR) tools. In most cases, state-of-the-art techniques for the recognition of handwritten documents can reach significant results in applications that either involve a small lexicon or deal with single writer documents.

At the beginning of our research, we attempted to tackle the text retrieval on the Gutenberg Bible by adapting open source OCR packages to this task. We trained the Tesseract [15] and the Gamera [16] classifiers with the characters in these collections. When the characters that form the words can be easily identified, the Tesseract classifier performs well. However, it is nearly impossible to achieve a satisfactory character segmentation of whole pages, because this software is designed to work with contemporary documents. As a consequence, the overall recognition rate is very low. The segmentation is more accurate with Gamera that is designed to work on historical documents. However, the unavailability of a suitable historical dictionary listing all the spelling variations did not allow us to take advantage of any linguistic information to improve the recognition. Among commercial tools, the *Abbyy FineReader OCR XIX* is the only one addressing the recognition of Gothic typefaces. We attempted to recognize our documents with this software, but since it has been designed to work with texts from the period between 1800 and 1938, it was able to recognize only few characters in the collection. A method for the automatic transcription of Latin manuscripts is presented in [17]. The approach uses a statistical model based on a generalized HMM that is trained with one instance of each letter. Even if these documents are not printed, their overall quality is better than the Gutenberg Bible and the words can be easily segmented. Most importantly, ligatures and abbreviations are not extensively used, and therefore, it is possible to take into account only 22 letter classes.

The methods that do not attempt to recognize the whole text in the indexed documents can be grouped in various

ways. In the following, we organize the approaches on the basis of the granularity adopted in the text representation. We begin with holistic approaches, that represent individual words with word level features, and then move on methods that represent the words with character codes. The last category contains methods that take into account features computed from a window moved across the text-line.

One common feature of most image-based word retrieval methods (including those dealing with handwritten documents) is the assumption that individual words can be reliably extracted from the text [18]. This assumption is true in modern printed books, but is in general not realistic in handwriting and in early printed documents. Holistic approaches deal with whole word images as basic objects to be identified. In this framework, some methods use word shape coding (e.g. [19]) where the documents are indexed by associating a code to each word image. To minimize errors due to wrong character segmentations, the codes are computed from a set of topological shape features such as ascenders/descenders, character holes, and character water reservoirs [19]. In [20], an approach inspired from cross-lingual retrieval is proposed. In this case, relevance models are used for searching handwritten word images. In particular, a joint probability distribution between features computed from word images and the corresponding transcriptions is considered. Word images are described in this case with holistic features based on shape features and on Fourier coefficients computed from word profiles. Techniques for clustering word images have been proposed to perform word spotting from handwritten documents. For instance, in [21] and [22], word clusters are computed by calculating pairwise word distance with dynamic time warping (DTW)-based techniques. Dynamic time warping is an algorithm for measuring the similarity between two sequences that has been initially applied in speech recognition systems. In the case of document images, the words are represented with a sequence of features computed in a single-column window slided across the word image. Word clustering is considered to speed-up searching in large document collections also in [23]. Similarly to the previous method, words are clustered considering the DTW between pairs of words that are represented with word profiles and structural features. The peculiarity of this approach is the adaptation of the DTW distance to deal with most common morphological word variants. Clusters are then annotated by their root word and this information is used to search the indexed documents. The speedup of the search in large collections is approached by Kumar et al. [24] where they discuss an indexing method that uses locality sensitive hashing (LSH) working with holistic word image features such as ascenders, descenders, and projection profiles.

One alternative word representation is based on character objects (CO). One CO is a part of a word image that might correspond to a character (in most cases, the CO is composed by one connected component). In methods based on character shape codes [25], each CO is replaced with codes that capture its main features. For instance, character shape codes can describe whether or not a given symbol has ascenders or descenders with respect to the text baseline. In this case, *g, p, q, and y* are all represented with the same code because they have descenders but not ascenders. With respect to OCR, this representation is faster to obtain and more robust to noise in the input images, but it is sensitive to touching characters. Character coding has been used also in [26] where text similarity between documents is computed. Each character is in this case represented with two feature vectors that are based on the horizontal and vertical traverse density. The characters extracted from a document image are clustered, and each cluster is represented with its centroid. At the end of the indexing, each character in the collection can be mapped to a unified class that corresponds to the cluster. In so doing, it is possible to compute document similarity without recognizing the text with OCR tools. The method described in [26] does not perform very well in the presence of touching characters that cannot be clustered in suitable classes. To partially avoid this problem, in [27], a word is represented by discrete entities that broadly correspond to strokes in characters. The features extracted from each word are discretized and represented with definite attributes so as to be able to employ an inexact feature matching based on DTW. With a related approach in [28], we proposed a word indexing and retrieval system that works with 18-19-th century books. The indexing is based on character codes, and a suitable word representation is proposed to deal with touching and broken characters. Also in this case, however, the indexing requires a word segmentation and cannot handle touching words.

The methods summarized so far rely on word segmentation as a first step in the indexing and retrieval of the textual content. To bypass the word segmentation, Cao et al. [29] proposed a method for handwritten keyword retrieval that models the word segmentation probabilities and integrates these probabilities in the word spotting model. When dealing with documents, such as those addressed in this paper, where the word segmentation is difficult or even impossible, techniques that search a query word in whole text-lines are required. In [30], the text-lines of historical handwritten documents are represented as ordered sequences of pixel columns. Each column is described with three features based on upper and lower profiles and on the number of black–white transitions. In principle, each pixel column is examined as a potential starting point of a candidate word. However, to speed up the search, a set of heuristics are used to discard positions that are unlikely to correspond to valid matches. To further reduce the use of an expensive DTW-based algorithm applied to image columns, a simple distance among upper and lower profiles is computed. The main weakness of this approach is

the high computational cost. Leydier et al. [31] further relax the segmentation requirements and propose a method based on the matching of strokes described with features computed from the gradient of the gray level image. The computational burden is in this case very high prohibiting any application to large datasets. In [32], N-gram language models are used in combination with HMMs for unconstrained handwritten word recognition. A fixed width window is moved column by column across each text-line, and a feature vector is computed at each position. In analogy with audio processing, this sequence of vectors is modeled with continuity density Hidden Markov Models. There is no word segmentation, and word boundaries are estimated during the decoding taking into account N-gram language models. In addition to this linguistic information, the system is trained by building 26 character models for the main classes. It is not possible to apply this approach to our data for two main reasons: first, due to ligatures and abbreviations, a significantly higher number of character classes should be modeled; second, the existing transcriptions of the Gutenberg Bible map different abbreviations in a unique word class. Therefore, it is not possible to use the transcription information (that has low relationship with the actual word appearance) for the HMM training.

Although in most cases required for handwritten text, the use of DTW with a granularity corresponding to pixel columns appears unfeasible for large data collections. If the words cannot be easily segmented, representing the text by character objects is probably the best compromise between word and pixel indexing. The use of DTW-based techniques at the character level has been described in some papers. One modified DTW that takes into account under- and over-segmentations of characters is described in [33] for Arabic documents transcription. To deal with duplicate document detection, Lopresti and Zhou modified the approximate string matching algorithm considering *split* and *merge* edit operations in addition to the standard *deletion*, *insertion*, and *substitution* [34]. The edit distance is considered also in [35] where confused characters in erroneous words are located and edit operations are applied to create a collection of erroneous error-grams that are used to perform query expansion. More recently, the edit distance has been used also for table identification [36].

The approach proposed in this paper represents text-lines on the basis of character objects that are matched with query words with a DTW-based method. The details of this approach are analyzed in the next Section.

4 The text retrieval method

One peculiarity of our approach is that character objects are clustered by means of a self-organizing map (SOM) that is trained with the COs extracted from a subset of the indexed pages. By using the SOM clustering, it is possible to label each CO with the class corresponding to the nearest cluster. In this way, we can speed-up the matching process with respect to a pure template matching approach working at the image level and we can use approximate string matching techniques. To improve the retrieval in the presence of touching or broken characters (e.g. see Fig. 5), we modified the approximate string matching algorithm to combine in a unique framework the SOM clustering and the information about the position and width of COs in the text-lines. The overall approach is described in the rest of this Section and is graphically synthesized in the system architecture diagram reported in Fig. 6.

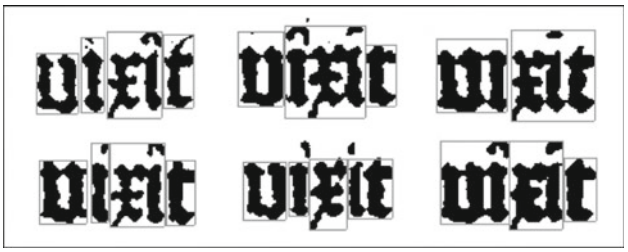


Fig. 5 Some instances of the word "vixit" that are segmented in different ways

4.1 Preprocessing and layout analysis

In the preprocessing and layout analysis step (*Text-line extraction*, Fig. 6), we extract the columns and the text-lines from each page. The techniques used to identify the text-lines have been partially tuned on few pages of one copy of the Gutenberg Bible. Concerning this task, it is important to notice that our aim is not to propose a general segmentation approach (alternatives techniques could be adopted to identify the text-lines) since the focus of this work is on retrieval techniques.

Since the page structure in the Gutenberg Bibles is rather regular, we implemented projection profile-based algorithms to identify columns, rows, and characters from each page. The text columns are extracted by computing the vertical projection profile. We obtain a perfect segmentation with the exception of few pages where pieces of illuminated letters are included in the columns. This problem had a low effect on the retrieval performance, because it did not affect the text-line identification. Since our method does not require word segmentation, the presence of additional symbols did not affect too much the overall performance.

Text-lines are subsequently identified analyzing the "gradient" of the horizontal projection profile (H_j) with a mobile window of 7 pixels:

$$G_j = \frac{\sum_{k=1}^{3}(H_{j+k} - H_{j-k})}{7}. \quad (1)$$

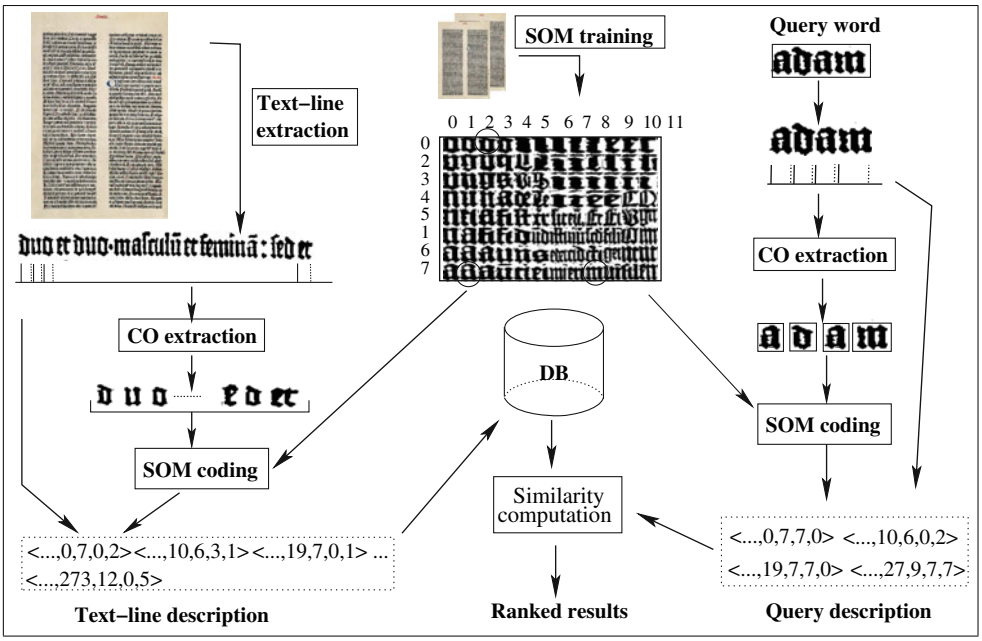


Fig. 6 System organization of the proposed approach

The text-lines are identified by thresholding the G profile since highest values correspond to text-background transitions. We subsequently identify pixels of the connected components that extend above or below the text-line bounding box and include all these pixels in an expanded text-line. The last segmentation step is the character extraction that is based on the identification of connected components in expanded text-lines (*CO extraction*, Fig. 6). Potential touching characters are searched among the widest connected components. In this case, we look for minima in the vertical projection profile of the component to be split.

Even if we carefully designed the segmentation algorithms, it is not possible to obtain a perfect segmentation and under- or over-segmentations can occur. Some examples are in Fig. 5 where we show different segmentations that are obtained from a few instances of the word 'vixit'. As a consequence of this problem, the matching algorithm should be tolerant to differences in the number of objects composing the words.

4.2 Character clustering

Each CO is represented by an 80-th dimensional feature vector that is obtained by linearly scaling the character to fit an 8×10 grid. The vectors corresponding to the COs extracted from a subset of the indexed pages are clustered with a self-organizing map, SOM (*SOM training* in Fig. 6).

The SOM [37] is an artificial neural network that performs clustering by means of unsupervised competitive learning. The SOM neurons, corresponding to clusters, are usually arranged in a two-dimensional lattice (the feature map), but higher-dimensional lattices are possible as well. Let us consider training samples represented by real vectors $x(p) \in R^n$, where p is the index of the sample ($p \in [1, N_P]$), and N_P is the number of training patterns. The SOM learning computes a data clustering, and each SOM neuron contains a model vector $m_i \in R^n$ that is the average of the patterns in the cluster. The number of neurons and their spatial organization are set by the user defining the SOM width and height. The initial values for cluster centroids can be selected with a random sampling of the data or with more complex initialization functions. Similarly, with k-means, the training is performed with several iterations where in turn cluster centroids are moved and patterns are assigned to the closest centroid. Unlike k-means, it is possible that some nodes have no patterns (and therefore no cluster) associated. During the training, the centroids are moved taking into account the values of neighboring nodes, and therefore, at the end of the training, the SOM neurons are arranged in the lattice in a way that preserves as much as possible the distance and proximity relationships of the original data. Additional details on the SOM training and its use for CO labeling in the case of modern printed documents can be found in [28,38].

In our system, the vectors obtained by CO encoding are used to train an SOM with a size of $X \times Y$ neurons. The SOM is trained with a subset of the characters to be indexed that represent the distribution of COs in the whole collection. In most cases, few random pages contain enough characters for this purpose. The trained SOM is then used to label each CO with a pair of integers $S_x(CO) \in [0, \ldots, X-1]$

Fig. 7 Example of a 12×8 SOM map. Each neuron is represented by the closest pattern belonging to the corresponding cluster

and $S_y(CO) \in [0, \ldots, Y-1]$ that identify the SOM neuron closest to the CO. An example of a SOM trained with the Gutenberg data is shown in Fig. 7. In the figure, each neuron is represented by the closest pattern in the training set belonging to the corresponding cluster.

Taking advantage of the spatial organization of the SOM, we can use the distance between prototypes in the map as a measure of similarity among patterns in the clusters. To this purpose, after training an SOM with the pages in the training set, we store all the pages by indexing each CO_i with the following n-uple:

$$CO_i = \langle Pag(CO_i), Col(CO_i), Row(CO_i), BB_l(CO_i), BB_r(CO_i), S_x(CO_i), S_y(CO_i)\rangle \quad (2)$$

where Pag, Col, and Row are the Page, Column and Row containing the CO, respectively. BB_l and BB_r are the left and right positions of the CO bounding box in the text-line. S_x and S_y are the coordinates of the SOM neuron that is the closest to CO_i. For example, in the left part of Fig. 6, we show the BB_l and BB_r positions of four COs in an indexed text-line. By checking the first n-uple ($\langle \ldots, 0, 7, 0, 2\rangle$), we can notice that the corresponding CO ('d') begins at the pixel 0 in the text-line, has a width of 7 pixels, and belongs to the cluster (0,2) in the SOM. This information is summarized for each text-line by the *Text-line description* shown in the bottom-left part of Fig. 6 and then stored in the database (*DB*).

In the retrieval, we use the distance among SOM prototypes to compute the similarity between characters associated with each cluster and we combine this information to identify words similar to the query as described in the following.

4.3 Similarity computation

The proposed text retrieval method is based on a query by example approach; the user selects from the document collection one word image that is used as a query. The image is subsequently split in its COs and represented similarly to indexed text-lines. One example of the query description for the word 'Adam' is shown in the right part of Fig. 6. The search for indexed words matching the query is performed with an adaptation of the dynamic programming solution to the approximate string matching that is described in this section (*Similarity Computation*, Fig. 6). In Sect. 4.3.1, we briefly summarize the standard method for text searching based on the edit distance, and in Sects. 4.3.2 and 4.3.3, we describe the modifications to the method proposed in this paper.

4.3.1 Text searching

The edit distance algorithm has been used in several contexts to compute a distance between two strings that takes into account three types of errors: *substitution*, *insertion*, and *deletion*. Let us first summarize the standard edit distance algorithm that is used to compare a query string Q (with $|Q|$ symbols) and a string T (having $|T|$ symbols). The basic data structure is a matrix $M_{0\ldots|Q|,0\ldots|T|}$ whose elements $M_{i,j}$ represent the minimum number of edit operations needed to match $Q_{1\ldots i}$ with $T_{1\ldots j}$. In other words, $M_{i,j} = Ed(Q_{1\ldots i}, T_{1\ldots j})$ is the edit distance of the two substrings $Q_{1\ldots i}$ and $T_{1\ldots j}$. The elements of the matrix are computed according to the following equation:

$$M_{0,0} = 0 M_{i,j} = min(M_{i-1,j-1} + \sigma(Q_i, T_j), M_{i-1,j} + 1,\ M_{i,j-1} + 1) \quad (3)$$

$\sigma(Q_i, T_j)$ is the cost of the substitution of Q_i with T_j, and the insertion and deletion costs are fixed to 1. In the basic formulation, $\sigma(Q_i, T_j) = 0$ if $Q_i = T_j$ and 1 otherwise. The value $M_{i,j}$ in Eq. (3) is computed using the values in the previous column or just above the current value. The matrix M is therefore computed starting from $(i, j) = (0, 0)$ evaluating one column after the other. The values of $M_{*,0}$ and $M_{0,*}$ are fixed to suitable values, usually ∞, or with increasing values ($M_{i,0} = i$, $M_{0,j} = j$). When the whole matrix is computed, the last position provides the edit distance between the two strings: $M_{|Q|,|T|} = Ed(Q, T)$.

Instead of using string matching, we use text searching if words are not extracted from text-lines. In text searching, we look for the pattern Q in the text T allowing an occurrence of Q to begin at any position in T. This is reflected in the matrix M by setting $M_{0,j} = 0$ for each j. After computing the whole matrix, lowest values in the last row indicate the final character of occurrences of Q in T. An example is

		a	d	e	m	a	d
	0	0	0	0	0	0	0
a	1	0	1	1	1	1	0
d	2	1	0	1	2	2	1
a	3	2	1	1	2	2	2
m	4	3	2	2	**1**	2	3

Fig. 8 Matrix M for searching the word "adam" in the text "ademad"

shown in Fig. 8 where the pattern "adam" has one occurrence with one error (a substitution) in the text "ademad".

The previous formulation is appropriate for a symbolic domain, where objects are represented by tokens belonging to a finite set of classes. Common examples are textual documents where the tokens correspond to the alphabet. In statistical pattern recognition, we frequently deal with numerical feature vectors rather than with strings. Image text retrieval is an intermediate case since the indexed objects are inherently symbolic, but are usually represented with numerical features.

In the following, we describe the proposed modifications to the text searching algorithm that includes in it the SOM-based similarity and the consistency check of the matching word width.

4.3.2 Weighting differences between COs

The simplest way to perform text searching on the basis of clustered COs considers that two characters match (and therefore have a zero edit distance) if and only if they belong to the same cluster. In our approach, we use the SOM to cluster the character objects and we can therefore consider the topological organization of the map when comparing characters. By inspecting the SOM in Fig. 7, we can observe that in many cases, closer clusters in the map correspond to similar characters. For instance, in the bottom-left part of the map, we can notice some 'a's whereas in the top-left part an 'o' smoothly changes to 'd' in the horizontal direction. By analyzing the first SOM column, we can also observe that the previous 'o' changes to 'v' and then to 'n'. The contiguity of SOM neurons therefore reflects the similarity of characters analyzed in Fig. 3.

The edit distance computation needs to be modified in order to take into account the SOM features. First, we consider Q and T to be sequences of COs instead of sequences of symbols. Second, we weight the similarity between symbols according to the distance in the SOM map between the corresponding clusters:

$$\sigma_S(Q_i, T_j) = \frac{\sqrt{(S_x(Q_i) - S_x(T_j))^2 + (S_y(Q_i) - S_y(T_j))^2}}{MaxSomDist} \tag{4}$$

where $MaxSomDist$ is the maximum distance between pairs of neurons in the SOM; S_x and S_y describe the neuron position in the SOM as defined in Eq. (2).

Taking into account this value of $\sigma_S(Q_i, T_j)$, it is now possible to recompute the matrix M with Eq. (3). The elements of M are in this case real values, but the overall interpretation is the same; lowest values in the last row of M correspond to best matching occurrences.

4.3.3 Considering the sub-word width

One important limitation of the DTW framework is that it implicitly considers that all the COs have the same size. This is not a problem if all the characters are accurately segmented, but poor results are possible when character segmentation errors occur. For instance, if one character is split into two sub-components, then the matching cost with a perfectly segmented word will be at least 2 (or more generally the cost of one deletion and one substitution). One solution to this problem relies on the introduction of *split* and *merge* edit operations (in addition to insertion, deletion, and substitution) in order to model typical OCR errors [34].

In our approach, we consider the width of the query characters in addition to the identifiers of the clusters the COs belong to. The width of the query image can be computed by $W(Q) = BB_r(Q_{|Q|}) - BB_l(Q_1)$ that is the difference between the rightmost point of the last character and the leftmost point of the first character in Q. Similarly, the width of a generic sub-query (composed by the first i characters in Q) can be computed by:

$$W(Q_i) = BB_r(Q_i) - BB_l(Q_1). \tag{5}$$

The computation of the width of a sequence of characters in the text to be searched T is more complex. When dealing with the character T_j in matrix M, the rightmost point of the sub-string in T is $BB_r(T_j)$. However, the leftmost point of the sub-word in T depends on the path followed in the computation of M. To trace the beginning of potential matching words, we associate with each element $M_{i,j}$ a value that corresponds to the leftmost point of the sub-word $T_{1\ldots j}$. These values are stored in a matrix $L_{0\ldots|Q|,0\ldots|T|}$ that is read and updated in parallel with M.

The costs for substitution, deletion, and insertion are computed taking into account three elements:

1. the accumulated cost in the M matrix up to the previous symbol;
2. the SOM-based character similarity measured by $\sigma_S(Q_i, T_j)$ that is computed with Eq. (4);
3. the compatibility of the matching sub-word lengths computed on the basis of the information stored in L.

The latter element has a different value if we consider a substitution ($\sigma_{WS}(Q_i, T_j)$), a deletion ($\sigma_{WD}(Q_i, T_j)$), or an insertion ($\sigma_{WI}(Q_i, T_j)$) since the resulting sub-word width will change according to the edit operation. In particular:

$$\sigma_{WS}(Q_i, T_j) = \frac{|W(Q_i) - (BB_r(T_j) - L_{i-1,j-1})|}{AvgW}$$
$$\sigma_{WD}(Q_i, T_j) = \frac{|W(Q_i) - (BB_r(T_j) - L_{i-1,j})|}{AvgW}$$
$$\sigma_{WI}(Q_i, T_j) = \frac{|W(Q_i) - (BB_r(T_j) - L_{i,j-1})|}{AvgW} \quad (6)$$

where $W(Q_i)$ is the width of the sub-query composed by the first i characters in Q (Eq. (5)) and the remaining part of the numerator is the estimated width of the matching text in T. For instance, $(BB_r(T_j) - L_{i-1,j-1})$ is the estimated width of the matching sub-word in T in the case of a symbol substitution. $AvgW$ is the average width of characters in the collection.

The two values $\sigma_S(Q_i, T_j)$ and $\sigma_{W.}(Q_i, T_j)$ are combined by means of two parameters (α and β) to compute the costs for substitution ($Cost_S$), deletion ($Cost_D$), and insertion ($Cost_I$):

$$Cost_S = \alpha \cdot \sigma_S(Q_i, T_j) + \beta \cdot \sigma_{WS}(Q_i, T_j) + M_{i-1,j-1}$$
$$Cost_D = \alpha \cdot \sigma_S(Q_i, T_j) + \beta \cdot \sigma_{WD}(Q_i, T_j) + M_{i-1,j} \quad (7)$$
$$Cost_I = \alpha \cdot \sigma_S(Q_i, T_j) + \beta \cdot \sigma_{WI}(Q_i, T_j) + M_{i,j-1} \quad (8)$$

After computing the three costs, we update $M_{i,j}$ and $L_{i,j}$ according to the minimum cost operation as follows:

- If $Cost_S = min(Cost_S, Cost_D, Cost_I)$ we have a substitution and we set $M_{i,j} = Cost_S$ and $L_{i,j} = L_{i-1,j-1}$.
- If $Cost_D = min(Cost_S, Cost_D, Cost_I)$ we have a deletion and we set $M_{i,j} = Cost_D$ and $L_{i,j} = L_{i-1,j}$.
- If $Cost_I = min(Cost_S, Cost_D, Cost_I)$ we have a deletion and we set $M_{i,j} = Cost_I$ and $L_{i,j} = L_{i,j-1}$.

To initialize the algorithm, we set suitable values for $L_{*,0}$ and $L_{0,*}$: $L_{i,0} = BB_l(T_1)$ for $(i = 1 \ldots |Q|), and L_{0,j} = BB_l(T_j)$ for $(j = 1 \ldots |T|)$. The latter values define the leftmost point of a potential matching word in the input text T.

When the whole text T is processed, the values in the last row of the matrix M correspond to errors of possible occurrences of Q in T. The lowest values identify potential matches whose starting coordinate is indicated by the corresponding value in L.

To summarize, in the proposed method, two factors contribute to the weight computation at each step: the distance between cluster centers in the SOM map and a comparison of the matching word lengths. By sorting the values computed for all the text-lines, it is possible to rank potential occurrences of the query word in the indexed documents (*Similarity computation*, Fig. 6) To give some insights into the method and into the results that can be achieved, we show one query, some positive answers, and the related features for the word *deus* in Fig. 9. Each CO is annotated with indexing information (we omit the page, column, and row information). For instance, the first CO has the following values: 96 is the leftmost point of CO ($BB_l(CO)$); 9 is the CO width; 2,1 denotes the SOM neuron ($S_x(CO) = 2$, $S_y(CO) = 1$). By comparing the SOM neurons of corresponding characters in different words, we can notice that similar characters are in many cases clustered in neighboring neurons. It is important to notice also that the query word is over-segmented since it is composed by five COs instead of the expected four characters. However, the proposed algorithm is able to identify also words with four COs and also one word that is under-segmented (the last one has 3 COs).

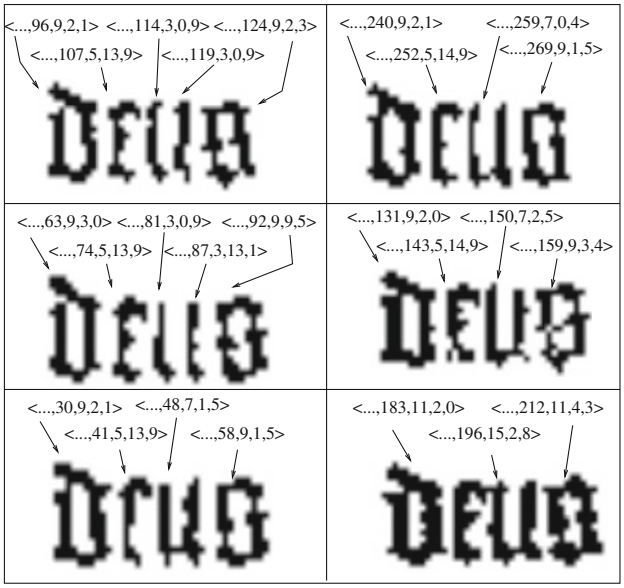
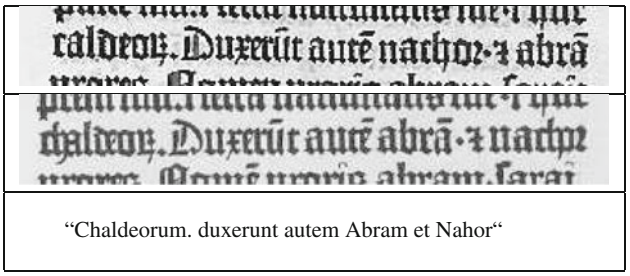


Fig. 9 Query word for "deus" (*top-left*) and five positive answers. Each character is annotated on top with its position, size, and cluster neurons

In the next section, we discuss the experiments that we performed to evaluate the proposed method.

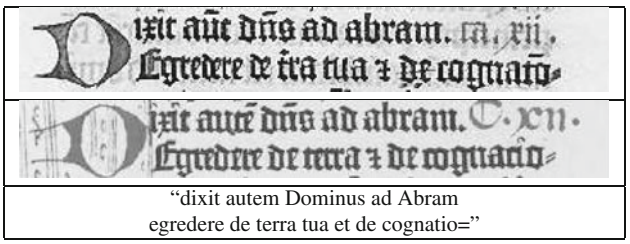
5 Experimental framework

The numerical validation of text retrieval in the Gutenberg Bible is a difficult task since an accurate transcription of the text is not freely available in electronic form. The content is well known because Gutenberg used the so-called *Biblia Vulgata* as source. Unfortunately, the text in the *Vulgata* versions that are available (e.g. [39]) is not a character by character transcription of the Gutenberg's work since it does not take into consideration ligatures and abbreviations. In the current system, the text is searched with a query by example paradigm and it is not possible to find occurrences of a query word that are printed with a very different ligature,



"Chaldeorum. duxerunt autem Abram et Nahor"

Fig. 10 Two text-lines that are organized in a different way in the Göttingen (*top*) and in the Munich (*bottom*) copies



"dixit autem Dominus ad Abram
egredere de terra tua et de cognatio="

Fig. 11 Two fragments of text in the Göttingen (*top*) and in the Munich (*bottom*) copies. The gray text and the capitals are painted in the originals. In the second line, different abbreviations are used

although minor differences can be compensated by the proposed distance. To measure the performance of the system, we manually annotated all the visual appearances of some query words taking into account the actual ligatures used. In so doing, it was possible to compute the *Precision* and *Recall* of various configurations of the system and to compare it with a baseline method on this subset of keywords.

5.1 The data set

We made our experiments on the "book" of the Genesis that is printed on 49 pages at the beginning of the first of the three volumes composing the Gutenberg Bible. We used the images available from three Digital Libraries: The Göttingen University Library [8], the Bayerische Staatsbibliothek in Munich [40], and the British Library [41] that holds two digitized copies (one printed on paper and one printed on vellum). In general, the pages can be downloaded by selecting by hand each page, with the exception of [40] where it is possible to download each volume as a single large PDF file. Other copies are available with lower resolution, such as the one from Keio in Japan [42]. However, we used in our experiments only the copies with a higher resolution.

The image sizes of the four copies are similar, ranging from 800 × 1129 of the two British Library copies to 965 × 1390 of the Göttingen copy. The texts in the above three copies appear aligned whereas the Munich copy has a different organization of the text in the pages (examples are shown in Figs. 10, 11). It is interesting to notice that in most cases, the pages and the text columns begin and end with the same sentence (frequently with the same word). Another important

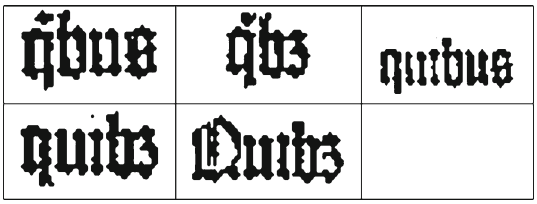


Fig. 12 Different forms of the word "quibus" in the Göttingen copy

feature of the Munich copy is that the first two pages contain 40 lines rather than the usual 42. In order to keep aligned the pages in the rest of the volume, the text in the first two pages of the Munich copy is therefore significantly "compressed".

5.2 Ground-truth

Although several copies of the Gutenberg Bible are available for download, ground-truth information cannot be easily accessed. Scholars in the Humanities agree that the Gutenberg Bibles have been printed starting from the so-called *Biblia Vulgata* which is a Latin version of the Bible that was common in the 15-th century. The identification of the closest text with respect to the printed books is out of the scope of this paper; however, by comparing the *Vulgata* text that is available on the Internet [39] with the digitized pages, we can notice that there is a general correspondence of most words. In Figs. 10 and 11, two examples of image fragments with the corresponding *Vulgata* text are shown. We can notice abbreviations (autem → aūt; dominus → dn̄s) and differences between the two copies, but the text roughly corresponds. Other differences are due to various spellings of the words (e.g. "Sarra" is wrote as "sara"; "terrae" as "terre") or to variations of the same word. In most cases, the query words used in our experiments have more than one form in the Genesis pages for each term in the ground-truth. For instance, the first occurrences of *quibus* in the Göttingen copy are printed as follows (in parentheses we write the number of occurrences of the corresponding form): quibus (1) quibs (8) q̄bs (3) q̄bus (1) Quibs (5). Some examples are shown in Fig. 12.

Even with the above-mentioned limits, the use of the Vulgata text [39] is, in our view, a good approximation of ground-truth. A perfect transcription of the text of each copy would be excessively expensive requiring a significant amount of skilled human effort. To take into account the variable organizations of text in each line, we built our ground-truth with a text-line resolution. We therefore edited the Vulgata ASCII file by adding appropriate line, column, and page breaks visually aligning the text with the Göttingen copy at the line level. Subsequently, we accurately identified the query words in the text-lines with the help of the ground-truth. Eventually, we visually checked different forms and

Table 1 Results obtained on the validation set with several values of (α, β)

Top	(α, β)	(0.0,1.0)	(0.25,0.75)	(0.5,0.5)	(0.75,0.25)	(1.0,0.0)
10	Precision	0.11	0.11	0.12	**0.13**	0.12
	Recall	0.10	0.11	0.12	**0.14**	0.12
	F1	0.10	0.11	0.12	**0.13**	0.12
20	Precision	0.06	0.06	0.07	**0.08**	0.07
	Recall	0.11	0.13	0.14	**0.15**	0.15
	F1	0.08	0.08	0.09	**0.10**	0.10
50	Precision	0.03	0.03	0.04	**0.04**	0.04
	Recall	0.14	0.16	0.18	0.18	**0.19**
	F1	0.05	0.05	0.06	**0.06**	0.06

Bold values denote the best results

locations of the query words in the four copies and annotated this information in the final ground-truth file that we used to compare the various retrieval methods, as detailed in the following.

5.3 Experimental results

We first separately indexed the Genesis pages of each copy considered in our experiments. The SOM training has been performed on a random subset of the indexed pages of each collection, and the trained map has been subsequently used for representing all the pages.

To identify the best combination of the two contributions to the overall similarity score (Eq. (8)), we considered one separated validation set composed by 92 words that we use as queries on the Göttingen dataset. Taking into account the query words in the validation set, we computed the Precision, Recall, and F1 values for some combinations of α and β values in Eq. (8). The three values are computed for the top-10, top-20, and top-50 most similar words. From the experiment summarized in Table 1, we identified the optimal values ($\alpha = 0.75$ and $\beta = 0.25$) that we used in the subsequent tests.

In the main experiment, we considered 22 words in the Genesis and used 134 occurrences of these words as queries. This set of queries does not include stop words and contains most person names (such as Abel, Abram, Cain, and Sara) that occur more than once with a given spelling. In correspondence with these 22 word classes in the transcription, there are 30 different spellings in the digitized books. To compute the Recall level, we manually annotated all the spelling variations of the queries that correspond to a total of 1363 words. From a morphological point of view, these 134 words include several types of situations. For instance, *dominus* and *quod* are words having a significant abbreviation and the latter word is composed by a single CO.

To analyze the methods, we computed the Precision and Recall in the top-10, top-20, and top-50 positions in the answer set comparing various approaches. To better asses the results obtained with the proposed method, we considered also a baseline method that can be used as a reference. To this purpose, we implemented a search based on the application of DTW on feature vectors extracted from a single-column slice that is moved across the indexed text and the query images. The four features that we considered are inspired by those used in [21]: the projection profile, the upper and lower word profiles, and the number of background/ink transitions.

In the main experiment, we compared the proposed method using $\alpha = 0.75$ and $\beta = 0.25$ (denoted by $M1$ in Table 2) with alternative approaches. In $M2$, we set $\sigma_S(Q_i, T_j) = 1$ if and only if Q_i and T_j belong to the same SOM cluster. In so doing, the SOM structure is not considered. We computed also the standard string edit distance (Ed). The last comparison method is based on the column-wise DTW (the baseline method previously presented). The experiments have been replicated for each of the four data sets. From Table 2, we can notice that in all the cases, the proposed $M1$ method has better values for Precision, Recall, and F1 with respect to the other methods. The approach based on the standard edit distance (Ed) is a reference that allows us to figure out the improvement that can be obtained with the proposed method because Ed does not consider both the SOM-based CO similarity and the integration of word widths in the DTW algorithm. Since $M2$ does not take into account the SOM contribution, the difference between $M2$ and Ed somehow measures the impact of the integration of word widths in the DTW algorithm to the overall performance. Likewise, the difference between $M2$ and $M1$ reflects the contribution of the SOM weighting to the retrieval performance. By inspecting the table, we could also conclude that the baseline method (based on a column-wise application of DTW) performs significantly worst than the proposed one. To have a fair comparison, we should consider that the features adopted have been originally proposed for cursive manuscripts, and therefore, it is possible that these are not particularly appropriate for the documents used in our experiments. A more accurate tuning of the features and of the baseline algorithm could provide improved results. However, from the analysis described in the next section, it is clear that the computational cost of this category of methods is significantly higher than the proposed one.

5.4 Complexity issues

The comparison with the column-wise DTW-based approach (the baseline) allows us to have a look at the computational cost of the various methods.

Without considering implementation specific optimizations, the cost of the word search for text-lines represented as columns of feature vectors is $O(N \cdot r \cdot n)$ where N is the number of indexed text-lines, n is the average number of

Table 2 Comparison of the results obtained on the test sets by the four compared methods

		Göttingen				BLP				BLV				Munich			
Top		$M1$	$M2$	Ed	DTW	$M1$	$M2$	Ed	DTW	$M1$	$M2$	Ed	DTW	$M1$	$M2$	Ed	DTW
10	Pr	**0.27**	0.22	0.17	0.13	**0.23**	0.19	0.15	0.15	**0.22**	0.18	0.14	0.13	**0.25**	0.20	0.19	0.11
	Re	**0.14**	0.11	0.09	0.03	**0.12**	0.10	0.10	0.09	**0.13**	0.12	0.11	0.10	**0.13**	0.11	0.11	0.08
	F1	**0.18**	0.15	0.12	0.05	**0.15**	0.13	0.11	0.12	**0.16**	0.14	0.12	0.12	**0.17**	0.14	0.14	0.09
20	Pr	**0.17**	0.14	0.11	0.09	**0.16**	0.13	0.09	0.10	**0.15**	0.11	0.08	0.09	**0.17**	0.13	0.12	0.07
	Re	**0.17**	0.13	0.11	0.04	**0.14**	0.12	0.09	0.11	**0.16**	0.13	0.11	0.12	**0.15**	0.13	0.12	0.10
	F1	**0.17**	0.14	0.11	0.06	**0.15**	0.12	0.09	0.11	**0.15**	0.12	0.10	0.10	**0.16**	0.13	0.12	0.08
50	Pr	**0.10**	0.08	0.07	0.06	**0.09**	0.08	0.05	0.06	**0.09**	0.06	0.05	0.05	**0.10**	0.07	0.06	0.04
	Re	**0.21**	**0.17**	0.15	0.07	**0.19**	0.16	0.12	0.14	**0.20**	0.16	0.14	0.16	**0.20**	0.16	0.15	0.13
	F1	**0.14**	0.11	0.09	0.08	**0.13**	0.10	0.07	0.08	**0.12**	0.09	0.07	0.08	**0.13**	0.10	0.09	0.07

Bold values denote the best results
Göttingen is the copy of the Göttingen University Library. BLP and BLV refer to the paper and vellum copies of the British Library, respectively. Munich is the copy of the Bayerische Staatsbibliothek

columns in the query images, and r is the average text-line width in pixels. On the other hand, the cost of approaches based on character object representations (such as the proposed method) is $O(N \cdot r' \cdot n')$ where N is again the number of indexed text-lines, n' is the average number of character objects in the query images, and r' is the average number of character objects in the indexed text-lines. In the documents used in our experiments, $N = 4030$, $n' = 4.32 - 5.38$, $r' = 25.13 - 31.80$, $n = 69$, and $r = 427$. For n' and r', we report the minimum and maximum values in the four collections considered in the experiments.

The theoretical gain from the computation point of view has been verified also during the experimental analysis addressed in this section. For instance, when comparing the performance of the various methods on the Göttingen data set, the word retrieval required an average CPU time[3] of 103 s in the case of the baseline approach that is significantly higher than the 3.61 s required by the $M1$ method. The time required to index the Genesis pages by the two approaches is similar, requiring 521 s for the baseline and 468 s for the proposed method. In the latter case, it is interesting to detail the time required by the sub-steps; the preliminary image analysis that performs the CO extraction required 182 s, the SOM training 44 s, and the final indexing 242 s.

6 Conclusions

In this paper, we propose a text retrieval method designed to deal with early printed books. These documents are represented by four copies of the Gutenberg Bible that we used as a test bed in our experiments. Two main ideas are considered in the proposed method: the SOM-based clustering of indexed characters and the integration of the word width information in a modified dynamic time warping matching algorithm. These techniques allow us to retrieve words without performing word segmentation and without assuming a perfect character segmentation with a computational cost that is reduced with respect to a column-wise application of the DTW algorithm.

Future work is related to the use of the system on larger collections, so as to address scalability issues, and to test its performance on other volumes of the same period.

[3] These times have been measured on a machine running the Ubuntu operating system and equipped with 1 Gbyte of memory and a single core CPU running at 1.86 Ghz.

References

1. Belaïd, A., Turcan, I., Pierrel, J.-M., Belaïd, Y., Rangoni, Y., Hadjamar, H.: Automatic indexing and reformulation of ancient dictionaries. In: International Workshop on Document Image Analysis for Libraries, pp. 342–354 (2004)
2. Gutenberg bible: In: Encyclopaedia Britannica, Chicago: Encyclopaedia Britannica (2010)
3. Gutenberg bible census: http://clausenbooks.com/gutenbergcensus.htm. Clausen Books
4. Le Bourgeois, F., Trinh, E., Allier, B., Eglin, V., Emptoz, H.: Document images analysis solutions for digital libraries. In: Proceedings of First International Workshop on Document Image Analysis for Libraries, pp. 2–24 (2004)
5. Agüera y Arcas, B., Fairhall, A.: Archaeology of type. Nature **411**, 997 (2001)
6. Digital library: http://gallica.bnf.fr/?lang=en. Bibliotheque nationale de France
7. Gotscharek, A., Neumann, A., Reffle, U., Ringlstetter, C., Schulz, K.U.: Enabling information retrieval on historical document collections: the role of matching procedures and special lexica. In: AND '09: Proceedings of the Third Workshop on Analytics for Noisy Unstructured Text Data, New York, NY, USA, pp. 69–76, ACM (2009)
8. Gutenberg digital: http://www.gutenbergdigital.de. Göttingen University Library

9. Coulmans, F.: Johannes Gutenberg. In: The Blackwell Encyclopedia of Writing Systems. Blackwell (1999)
10. Wild, A.: La typographie de la bible de Gutenberg. Cahiers GUT enberg, **22** (1995)
11. Smigiel, E., Belaïd, A., Hamza, H.: Self-organizing maps and ancient documents. In: Document Analysis Systems, pp. 125–134 (2004)
12. Gupta, M.R., Jacobson, N.P., Garcia, E.K.: OCR binarization and image pre-processing for searching historical documents. Pattern Recognit **40**(2), 389–397 (2007)
13. Delalandre, M., Ogier, J.-M., Lladós, J.: A fast CBIR system of old ornamental letter. In: International Workshop on Graphics Recognition, pp. 135–144 (2007)
14. Karray, A., Ogier, J.-M., Kanoun, S., Alimi, M.A.: An ancient graphic documents indexing method based on spatial similarity. In: Int'l Workshop on Graphics Recognition, pp. 126–134 (2007)
15. Tesseract OCR: http://code.google.com/p/tesseract-ocr/
16. Gamera: a framework for building document analysis applications: http://gamera.informatik.hsnr.de/index.html
17. Edwards, J., Teh, Y.W., Forsyth, D.A., Bock, R., Maire, M., Vesom, G.: Making latin manuscripts searchable using gHMMs. In: NIPS (2004)
18. Konidaris, T., Gatos, B., Ntzios, K., Pratikakis, I., Theodoridis, S., Perantonis, S.J.: Keyword-guided word spotting in historical printed documents using synthetic data and user feedback. Int. J. Doc. Anal. Recognit. **9**(2–4), 167–177 (2007)
19. Lu, S., Li, L., Tan, C.L.: Document image retrieval through word shape coding. IEEE Trans. Pattern Anal. Mach. Intell. **30**(11), 1913–1918 (2008)
20. Rath, T.M., Manmatha, R., Lavrenko, V.: A search engine for historical manuscript images. In: ACM SIGIR 04, pp. 369–376 (2004)
21. Rath, T.M., Manmatha, R.: Word image matching using dynamic time warping. In: CVPR (2), pp. 521–527 (2003)
22. Rath, T.M., Manmatha, R.: Word spotting for historical documents. Int. J. Doc. Anal. Recognit. **9**(2–4), 139–152 (2007)
23. Balasubramanian, A., Meshesha, M., Jawahar, C.V.: Retrieval from document image collections. In: Document Analysis Systems, vol. 3872 of Lecture Notes in Computer Science, pp. 1–12, Springer (2006)
24. Kumar, A., Jawahar, C.V., Manmatha, R.: Efficient search in document image collections. In: Computer Vision—ACCV 2007, 8th Asian Conference on Computer Vision, Tokyo, Japan, November 18–22, 2007, Proceedings, Part I, vol. 4843 of Lecture Notes in Computer Science, pp. 586–595, Springer (2007)
25. Smeaton, A.F., Spitz, A.L.: Using character shape coding for information retrieval. In: International Conference on Document Analysis and Recognition, pp. 974–978 (1997)
26. Tan, C.L., Huang, W., Yu, Z., Xu, Y.: Imaged document text retrieval without OCR. IEEE Trans. Pattern Anal. Mach. Intell. **24**(6), 838–844 (2002)
27. Lu, Y., Tan, C.: Information retrieval in document image databases. IEEE Trans. Knowl. Data Discov. **16**(11), 1398–1410 (2004)
28. Marinai, S., Marino, E., Soda, G.: Font adaptive word indexing of modern printed documents. IEEE Trans. Pattern Anal. Mach. Intell. **28**(8), 1187–1199 (2006)
29. Cao, H., Bhardwaj, A., Govindaraju, V.: A probabilistic method for keyword retrieval in handwritten document images. Pattern Recognit. **42**(12), 3374–3382 (2009)
30. Kolcz, A., Alspector, J., Augusteijn, M., Carlson, R., Viorel Popescu, G.: A line-oriented approach to word spotting in handwritten documents. Pattern Anal. Appl. **3**(2), 153–168 (2000). doi:10.1007/s100440070020
31. Leydier, Y., Le Bourgeois, F., Emptoz, H.: Omnilingual segmentation-free word spotting for ancient manuscripts indexation. **1**, 533–537 (2005)
32. Vinciarelli, A., Bengio, S., Bunke, H.: Offline recognition of unconstrained handwritten texts using HMMs and statistical language models. IEEE Trans. Pattern Anal. Mach. Intell. **26**(6), 709–720 (2004)
33. Lorigo, L.M., Govindaraju, V.: Transcript mapping for handwritten Arabic documents. In: Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series, vol. 6500 (2007)
34. Lopresti, D.P.: String techniques for detecting duplicates in document databases. Int. J. Doc. Anal. Recognit. **2**(4), 186–199 (2000)
35. Fataicha, Y., Cheriet, M., Nie, J.Y., Suen, C.Y.: Retrieving poorly degraded OCR documents. Int. J. Doc. Anal. Recognit. **8**(1) (2006)
36. Lopresti, D.P.: Optical character recognition errors and their effects on natural language processing. In: Workshop on Analytics for Noisy Unstructured Text Data, pp. 9–16 (2008)
37. Kohonen, T.: Self-organizing maps. Springer Series in Information Sciences, (2001)
38. Marinai, S., Marino, E., Soda, G.: Self-organizing maps for clustering in document image analysis. In: Machine Learning in Document Analysis and Recognition, pp. 193–219 Springer (2008)
39. Biblia vulgata: http://www.thelatinlibrary.com/bible.html. The Latin Library
40. Gutenberg bible vol.1: http://daten.digitale-sammlungen.de/~db/bsb00004647/images/. Bayerische Staatsbibliothek
41. Treasures in full: Gutenberg bible: http://www.bl.uk/treasures/gutenberg/homepage.html. British Library
42. Takamiya, T.: How to make good use of digital contents: The Gutenberg bible and the HUMI project. In: Kyoto International Conference on Digital Libraries, pp. 110–112, (2000)