University of Groningen

Design considerations for a large-scale image-based text search engine in historical manuscript collections

Schomaker, Lambertus

Published in:
Information Technology

DOI:
[10.1515/itit-2015-0049](https://doi.org/10.1515/itit-2015-0049)

IMPORTANT NOTE: You are advised to consult the publisher's version (publisher's PDF) if you wish to cite from it. Please check the document version below.

Document Version
Publisher's PDF, also known as Version of record

Publication date:
2016

[Link to publication in University of Groningen/UMCG research database](#)

Citation for published version (APA):

Schomaker, L. (2016). Design considerations for a large-scale image-based text search engine in historical manuscript collections. *Information Technology*, 58(2), 80-88. <https://doi.org/10.1515/itit-2015-0049>

Copyright

Other than for strictly personal use, it is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license (like Creative Commons).

The publication may also be distributed here under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license. More information can be found on the University of Groningen website: <https://www.rug.nl/library/open-access/self-archiving-pure/taverne-amendment>.

Take-down policy

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Downloaded from the University of Groningen/UMCG research database (Pure): <http://www.rug.nl/research/portal>. For technical reasons the number of authors shown on this cover page is limited to 10 maximum.

Special Issue

Lambert Schomaker*

Design considerations for a large-scale image-based text search engine in historical manuscript collections

DOI 10.1515/itit-2015-0049

Received November 30, 2015; accepted December 17, 2015

Abstract: This article gives an overview of design considerations for a handwriting search engine based on pattern recognition and high-performance computing, “Monk”. In order to satisfy multiple and often conflicting technological requirements, an architecture is used which heavily relies on high-performance computing, interactivity, and a Posix file-access model for the scientific programmers. The resulting system is able to handle billions of image files, in the order of petabytes of storage capacity, with a single mount point. Monk is operational since the year 2009.

Keywords: Search engines, handwriting image retrieval, continuous learning, crowd sourcing, high-performance computing.

ACM CCS: Applied computing → Document management and text processing → Document capture → Optical character recognition, Software and its engineering → Software organization and properties → Contextual software domains → Operating systems → File systems management, Computing methodologies → Machine learning → Learning paradigms → Multitask learning → Lifelong machine learning

1 Introduction

Current developments in “Big Data” computing allow for new approaches to traditional problems in pattern recognition. The massive amount of data is, on the one hand, an opportunity: The approximation of statistical models

is easier and more reliable. Also, in massive data collections the probability of actually finding a relevant result for a particular query goes up. Users have become accustomed to the principle of the “hit list” as an approximate answer to their information need. Their querying behavior is much more haphazard than the designers of traditional relational databases had in mind. For the machine, search has become a process of pattern association rather than a set-theoretic data selection process. Therefore, and on the other hand, these new developments in search and retrieval require many new technological solutions, even for the relatively simple case of traditional (encoded ASCII or Unicode) text search. New methodologies such as NoSQL [20, 23], key-value (K-V) stores [1] and the MapReduce [9] computing paradigms have changed the playing field. It may be appreciated that the necessity of new computational paradigms becomes even much stronger if one deals not with encoded text, but with raw scanned document images of handwritten text in massive amounts. We are currently building a European search engine for multi-language, multi-script handwritten document collections: the Monk system. In order to achieve this goal, a number of engineering decisions needed to be made, which will be described in this paper. Before getting at the technical aspects, it is important to briefly introduce the problem of handwriting recognition. Here, we define it as the process of generating text hypotheses in ASCII or Unicode for a given image that contains handwritten content.

A traditional route for allowing search engine access to historical handwritten collection would consist of converting the scanned images to encoded text using optical character recognition (OCR) [3], and subsequently applying an existing generic framework for text search. One could use software packages such as Lucene [11], or more simply, put the OCR output data on the Internet and leave text search to Google itself. The problem with this approach is that *character* recognition requires: a) that the characters in the text are identifiable and recognizable; b) that these characters, of many possible script styles and languages, are known and trained to the OCR system. Un-

*Corresponding author: Lambert Schomaker, Faculty of Mathematics and Natural Sciences, University of Groningen, Nijenborgh 9, NL-9747 AG Groningen, The Netherlands, e-mail: l.r.b.schomaker@rug.nl

Figure 1: Image examples of common, handwritten words with a questionable possibility of correct character segmentation for each of the letters. It is clear that traditional Optical Character Recognition (OCR) may not be the optimal approach. In the Monk system, the whole-word image is proposed as the target object for pattern recognition and machine learning.

fortunately, neither of these two prerequisites can be satisfied for the general case of search in large handwritten collections of diverse origins. In many styles, (Figure 1) the characters are hardly legible and slurred, suffering shape distortions that make an easy segmentation very difficult. Furthermore, only a tiny fraction of the world manuscript heritage is accompanied by labeled training sets representing the “alphabet” of each handwriting style. In the current practice of handwriting-recognition research, advanced modeling is applied, for instance using a combination of Hidden-Markov Models [13] and statistical linguistic models in order to handle the problems of ambiguity of written graphemes. Therefore, for one particular old manuscript, there needs to be sufficient human expertise on the content, i.e., yielding text labels for regions of interest in the images and sufficient human expertise in machine learning in order to obtain reasonable results. However, such an approach (“one PhD project per book”) is not scalable and cannot benefit from the recent technological advances in high-volume processing. In recent years we have developed a large-scale effort to approach this problem from a slightly different angle. By positioning handwriting recognition as an electronic transcription tool that basically process a scanned image once, the user will be disappointed: The recognition rates will be somewhere in the range of 60–95%. Scholars in the humanities are even dissatisfied by the OCR quality of machine-printed text, which may be far above 95%. This is frustrating to both users and system designers. Therefore, it is better to position the handwriting recognition function as a retrieval or search tool, without the promise of faithful transcrip-

tion, as yet. This what we have realized in the Monk system, taking the position that graphical objects with redundant visual information such as whole cursive words or Chinese characters are the targets for pattern recognition and machine learning. By allowing users to train the engine over Internet in a 24/7 machine-learning process, the list of known shapes will grow over time and the agglomeration of examples for a word class will lead to a continuous improvement in the machine-learning process. Below the hood of this e-Science service, a similar iterative improvement takes place as pattern-recognition researchers are improving the machine-learning methods over time. Especially for national and regional archives, such a functionality may be interesting. The perspective is based on the long term. Collections that were digitized in the previous decades currently are only accessible by human reading of the scanned image page, of which there may be thousands in a folder. Documents are only annotated at the level of the book, its title, origin etc. but there is no access to the internal text content. In the Monk system, the coverage of the text recognition will be dependent on user interest. Documents that attract attention of scholars and volunteers will be mapped with increasing accuracy over time, while other manuscripts are basically waiting to be labeled and then enter the retraining flux process. This continuous, iterated approach is very demanding on the machine and required a large-scale design and implementation effort, starting in the year 2009.

The pattern-recognition aspects have been and will be described elsewhere. In [26] we discovered that word retrieval can be surprisingly effective without requiring character models. This has lead to the current technical implementation of word matching in Monk. Overall usage considerations are given in [27], describing the causes for the apparent lack of general search engines for handwriting. In [15] we present a basic aspect of the Monk approach, namely the iterative improvement of system performance by an alternation between (Step I) word recognition, i.e., classifying an image X as being a member of a word class W by maximizing the probability $p(W|X)$ and, subsequently (Step II) by maximizing the retrieval quality (“how intuitive are ranked results?”), i.e., optimizing $p(X|W)$. The latter leads to well-formed hit lists of words such that correct instance are easy to label by the user. The word models are retrained and the process restarts with the recognition step (I). We call this the “Fahrkunst” principle, akin to the old two-beamed mine elevators in the German Harz region, where the miner could rise to the surface by jumping to the other footstep at the right moment. As the number of reliable labels increases, steps towards regular “left-to-right” recognition can be made,

deriving individual letter models from labeled words. At that moment, given enough training samples, the regular hidden-Markov [13] and the more recent BLSTM-CTC [5] character-based neural networks can be trained. It still remains an open question to what extent statistical linguistics can be used for improving the bottom-up recognition by these systems. The training corpora that are used for top-down word expectancy given a word n -gram context as is used, e.g., in the Google search engine and in machine translation are at least in the order of a billion words, embedded in sentences of contemporary text [17]. Such an amount of training data is exactly not applicable for improving bottom-up recognition results of a just-scanned historical manuscript image. The handwriting will mostly be in a language, content and style that is not well known yet: The whole purpose of digitization is usually to construct the first digital linguistic resource for this manuscript and its general class. Apart from the content of documents, i.e., the “What?” question, image processing techniques are also focused on “When?” or document dating, “Where?” for geographical provenance, and “Who?”, in order to find a known scribe from a large collection of manuscripts [2, 6–8, 14, 21, 25]. In [6], it was shown that cross-script writer identification can be realized for Chinese and English manuscripts by one hand, by using small structural details, ink junctures, in the scanned image. The amount of data collected in Monk could also be used to study some peculiarities of training hidden-Markov models (HMMs), notably the fact that a good recognition performance does not mean that an underlying Markov model was estimated properly [16].

In this paper, however, we will present the more technical design considerations applicable to a search engine of handwritten collections. A global overview of the architecture of the system will be given. Details on the recognition methodologies have been and will be published in other platforms. One of the design considerations implicates the convenient use of tools by researchers in an e-Science context, i.e., that they do not need to revert to highly particular, time and version bound solutions of complicated tool libraries with a high interdependence. The question was whether we could present developers a regular Posix [19]-based file system instead of a specific I/O framework for massive storage of hundreds of millions of variable-sized image objects. Quantitative information will be presented together with practical experiences.

2 Design considerations for a handwriting search engine

First and foremost, it is important to realize that handwriting recognition is not a solved problem, similar to the situation in speech recognition and computer vision in general. This means that the performance of such systems is not 100% correct. Consequently, there is a continuous scientific effort in improving such systems. Therefore, a search engine in these domains needs to be configurable and live together with a relatively fast development process of new fundamental processing tools, in many possible programming languages, written by different programmers, PhD students and open-source developers: There is no closed system design as in building a normal physical road bridge or a highway, where centuries of experience can be used. Furthermore, in science and engineering research, it is very costly to force the human workers into new and potentially time-variant software solutions. Such solutions may change too fast over the duration of a PhD project and imply a costly maintenance process known as “dependency hell” [4]. Therefore, it would be very attractive, if programmers and scientific users could use their own generic programming environment, including the standard tools and an general purpose file system. Given the scale of our problem, the envisaged storage, which is in the order of multiple petabytes with hundreds of millions of objects, should ideally be accessible via a regular Posix [19] file I/O interface. Let us call this design issue the “e-Science requirement”. In the Posix standard [10], all input and output is done using files, regardless of the device (disk, tape, screen, peripheral) and accessed using the same function calls. A file is an ordered sequence of bytes, all meaning is provided by the application that reads from or writes to it. A special file, the directory file, contains references to other files usually in a hierarchical structure. Typical functions are open, close, read, write, mkdir, readdir, rename, etc. A file can be found on the basis of an identifying string. The Posix file system allows for finding data on the basis of the identifier: $Y = f(Id)$ corresponding to a read operation from storage f and $f(Id) = Y$ corresponding to a write. The general Posix file-system description makes no reference to a preferred file size. However, efficiency considerations for storage on rotating magnetical disks have resulted in solutions that implicitly assume big block sizes, even if smaller items are stored in them. In such an environment, the use of large files is preferred. Other storage technologies such as SSD, however, may have a different optimality as regards granularity. It is also notable that the com-

mon and often sensible software-engineering tendency towards abstraction has also created a large distance between the Posix layer and the application. The basic operations (open, read, write, and close) reappear at each intermediate abstraction layer. At the functional level, however, all the user want to do is, e.g., to obtain a data item by $Y = f(Id)$ and the machinery for this is already provided at the Posix level. An important assumption is that mutations in the file-name space are atomic, similar to database record updates.

A **second** requirement is interactivity, allowing for keyword search, browsing and reading of historical text images and recognition results. Additionally, such a system should facilitate the labeling of image regions with their correct text labels, either by human scholars or volunteers from the general public (“the crowd”). In the operation of such a search engine there will exist several groups of users: The system administrator; scientific programmers; collection curators; humanities scholars; machine-learning researchers; and users from the general public with an interest in historical documents, regional history, genealogy, etc. Because the automatic handwriting recognition is not solved, there is a continuous need for text labels and a continuous retraining of the pattern-recognition methods.

The **third** requirement is storage capacity itself. Multiple institutes will have several collections, each collection consisting of several books. There may be hundreds of institutes, with hundreds of collections per institutes. Individual documents may contain hundreds or thousands of pages.

The **fourth** requirement is computing capacity. Due to the scale of the problem and the continuous flux of the system: each new label disturbs the current state of the machine-learning process and calls for a retraining the proposed system requires an ample amount of computing resources. These resources vary from large-scale Linux compute clusters with thousands of nodes and hundreds of competing users from other disciplines submitting their batch jobs, to GRID-based computing, Cloud-based computing, dedicated compute servers that are available 24/7 for the search engine, upto massive parallel computing facilities in the form of GPU racks or mainframes such as provided by the IBM BlueGene series or similar hardware by SGI, Cray or HP. Here, again, a major limiting factor is man power for the realization of the connections between storage and computing. An “e-Science” facility such as Monk must be able to benefit from computational resources in its ecological niche in a flexible manner, based on quick solutions, depending on the current computational requirements. As a general rule, computing demands steeply rise

when there are singular events such as the ingest of a new large collection with several thousands of scans. Another cause of increased computational demand can be a new gulf of text labels for image regions in a collection that suddenly attracts user interest.

3 Solutions

In the year 2009 we teamed up with astronomers, because of their decades of experience with image processing and long-term archives. The perspectives of astronomers are in many ways comparable to those of archivists in the humanities. The time horizon is in terms of decades and there is a healthy distrust of solutions that are either localized in time (trendy) or tied too much to the fate of a particular (small) company. Also open source solutions are considered to have their limitations and risks from the viewpoint of these users, astronomers and archivists alike. A large-scale interdisciplinary project was granted, Target [24], with the goal of developing a 20 petabyte storage cluster and high-performance computing facility for astronomers, genomics researchers and machine-learning researchers. The requirements differ vastly over these disciplines. Whereas astronomy uses huge images of hundreds of megabytes and larger, the number of characteristics per object such as a star may be just in the hundreds (of columns). In genomics, there is a large memory requirement, with tens of thousands of columns (RNA expression intensities) for tens of thousands of patients.

Requirement 1: a familiar e-Science development environment

For our image classification problem however, word matching, Monk needs to process hundreds of millions of potential word candidates and the number of shape features (columns) may be in the thousands. The highly variable size of image material in the case of text images in Monk precluded the use of standard relational databases which also cannot handle the large numbers of columns per object. On the other hand, key-value (K-V) stores were interesting, but appeared to be less reliable than commercial database systems. Today, for secure transactions, the provably atomic database transactions are preferred over K-V stores which have a different quality-of-service orientation, trading reliability for speed. In the evaluation process, the question arose whether it would be possible to allow users normal Posix file access to hundreds of millions of files on a single mount-point file system? This question

was put forward to the company participating in the Target consortium, IBM¹ which provided the – at the time – still experimental hardware and software environment. Traditional considerations of RAID systems, optimized considering block sizes on rotating disks, etc., would have lead to an immediate dismissal of this proposal. However, it was also clear at the time (2009) that storage technologies would start to change quickly. Therefore the challenge was accepted. Using the IBM general parallel file system (gpfs), disk partitions were made, optimized for the different e-Science disciplines. The gpfs system uses a separate distributed file metadata system, presenting the file system itself as a single-mount Posix network file system. Adaptations for the Monk system involved a smaller block size and a massively increased capacity for a large number of files by a very large maximum count for the number of “inodes”. Since this was a research project, there have been several ups and downs, but for the Monk system, the results are quite attractive, as will be shown in the next section. Notably after the replacement of the gpfs metadata disk system by SSD disks, the performance of the storage facility at large became satisfying.

Requirement 2: interactivity

By using a standard Apache web server, all of the needed interactivity can be realized in HTML and Javascript. Currently, the techniques have advanced to a point where the selection of regions of interest with the mouse can be realized at the client side, using Javascript. This is the easy part. However, interactivity is frowned upon in most of the high-performance computing centers. The computing model is based on queued batch jobs. In a continuously learning system such as Monk, it would be highly disappointing for users if the beneficial effects of their labeling efforts will only become visible at a late and unpredictable moment in time, such as “maybe next week”. It did cost us quite some time to convince the system operators and policy makers of the high-performance computing (HPC) center that interactive HPC has the future. Of course this is still realized within a queued-job framework. The solution is to package tasks into a large number of manageable computing efforts that each can be finished in a matter of not more than two hours. If the system response is not immediate, this is acceptable to users, when they observe the effects of recomputation over a wide range of a document

by the new word classes and the improved hit lists for existing word classes as time proceeds.

Requirement 3: storage capacity

The original and highly ambitious goal was to realize a 20 petabyte storage facility. Ultimately, we realized a 10 petabyte disk-storage facility, including a large tape robot system. A design error in the initial stages of the project was that the different disciplines, with their highly specific optimalities were all given access to the same network mount point, with disk-parameter configuration which was a compromise over all requirements. The requirements of the Monk system, with its large number of inodes put a heavy load on the single metadata server for the gpfs file system, whereas the massive throughput of huge astronomical images clogged the network infrastructure. Therefore, in spite of the maintenance advantage of the “one size fits all” configuration, the storage facility was split in pools, one for the astronomers (huge files and traditional Oracle databases), one for the genomics users (intermediate file sizes, massive memory requirements and high security due to the presence of patient data), and, finally, one pool for the Monk system with its large number of variable-sized but relatively small files. Although the Monk configuration was frowned upon by the engineers, over the years it appeared quite stable. In order to prevent inefficient disk use, the gpfs software was adapted by IBM to store very small files in the metadata server itself. The user does not notice whether a file is a regular block in Posix or a database record from the metadata server. In case of disk problems, the effects were more detrimental in the configurations dealing with huge files that each have a long chain of inodes than in the Monk setup, where files often reside in single disk blocks. It should be noted that this storage facility can only be effective by using solid-state disks (SSD) at critical points. In the initial project stages, the metadata server operated with rotating magnetic disks that essentially could not handle the high access load. Finally, as regards storage, the users are understandably worried about the backup situation. It should be noted that both the actual data and the disk metadata need to be backed up. Since these system components are separate, this requires considerable maintenance efforts, especially at the petabyte scale. Not only the data, but also its index is huge. Although there were some hardware and system software problems, for instance a failure of the cooling system during a hot summer there is, to date, no serious data loss for the case of the Monk system.

¹ The author is in no way related to this company

Figure 2: A schematic overview of the architecture of Monk. The system generates a large surface of word-candidate images. If one is labeled and new, this initiates the first construction of a word-class models. If such a class already exists, the models are refined. A system of agents is loosely based on the metaphor of the monastery. The Abbot detects the necessity of computing tasks, which are delegated to Novices. The input of human labelers disturbs the continuous learning process and keeps it in a flux.

Requirement 4: Computing

Because the available and necessary computing resources are highly variable, it is undesirable from the software-engineering point of view that the Monk search engine relies on a particular queuing system, such as PBS [18] or SBATCH [22] for Linux clusters.

In order to prevent an excess of solutions, we designed a Monk-specific queueing system (Figure 2). This system detects the necessity for recomputations and autonomously feeds its own queue. The requests are then forwarded or used by different compute facilities. In this manner, for each of the available compute resources an interface needs to be constructed, but this changes infre-

Table 1: Statistics of the Monk collection status, May 2015.

Item	Count
Books	372
Pages	72 k
Lines	1.8 M
Labeled (human-confirmed) words	640 k
Lexical words (classes)	115 k
Word-zone images	467 M

quently. Apart from using a generic Linux cluster, the system is using permanently allocated “front-end servers” in order to maintain the 24/7 interactivity. Then, there have been experiments with GRID and Blue Gene, which were not fruitful for a variety of reasons. A new and successful compute facility is the use of IBM microservers (24-core/28 GB single board computers running at less than 100 Watts each) of IBM Research, Zurich. Due to the separation of function, the migration from a PBS (Portable Batch System [18]) to an SBATCH (Slurm [22]) cluster at the university’s HPC center could be realized within half a day by adapting a few scripts. In practice, the Monk system has been using 60–90 cores continuously since the start, with peak loads of hundreds of cores at occasions. However, it should be noted that this is not enough for an almost real-time experience for the labeling users. As the number of manuscript collections goes up, the compute facilities need to be scaled up correspondingly.

4 Some numbers and facts

Table 1 shows the numbers of books, pages, lines, labeled words, lexical words and gross number of word-candidate images. It should be noted that word segmentation is a “chicken and egg” problem. The correct segmentation can only be known if the word is recognized, but in order to be recognized, a word needs to be segmented correctly from its context. Due to the variable spacing of text and the large number of strokes and embellishments, a reliable bottom-up segmentation cannot be performed. Therefore the system needs to present the word classifier with a large number of (partially) overlapping word candidates, explaining its large number. As words are reliably mapped, a process of pruning may reduce the number of images (read: inodes). The human-confirmed word labels are the actual harvest of the Monk system, which can be viewed as an observatory for (image, label) tuples needed in machine-learning research.

Table 2: File system statistics, May 2015. The maximum storage is > 1 PB. Please note the high number of inodes (read “files”): such a quantity would be considered extraordinarily high in most file-system implementations. The users see a Posix file system which is realized through a distributed, SSD-based metadata server using regular database technology as part of the server infrastructure, i.e., out of view.

Filesystem	inodes	iUsed	iFree	iUse%
/dev/gpfs9	2.0 G	1.3 G	775 M	63%
	Storage	Used	Avail	Use%
	104 TB	93 TB	11 TB	90%

Figure 3: Histogram count of file occurrence as a function of file size, in log-log scale. The sample date is four months later than that of Table 2.

Table 2 shows a few current statistics of the Monk file system. It should be noted that the maximum storage size is not a fundamental limitation: it can be increased by a factor of ten within the given hardware configuration. The maximum number of files (inodes) is massive. It has been growing from 500 M inode to 2 G inode during the last year, in anticipation of new collections being ingested by the system. However, the ambition is a ten-fold scale up during the coming years.

Figure 3 shows the distribution of file occurrence as a function of file sizes, in log-log scale. As in many natural size distributions, there is an abundance of smaller objects. The dip represents the separation between text and data on the left and images on the right.

Figure 4 shows the distribution of word-recognition accuracy for 126 manuscripts in different languages and international scripts. A selection has been made of words with at least 50 labeled samples, randomly selecting 25 samples of training and 25 for testing, using nearest-neighbour match. The Monk recognition accuracy and the number of lexical words per manuscript increases over

Figure 4: Histogram of manuscript frequency per recognition-accuracy bin (width 3%), on train/test sets of 25/25 labeled samples.

time as labels are harvested: this is a snapshot of August 2015. The overall recognition rate was 94477 words out of 115940 (81.5%) over all scripts (medieval Western, Chinese, contemporary cursive, etc.). As the histogram shows, manuscripts differ in their degree of difficulty. For large manuscripts it is possible to improve the results using, e.g., neural-network classifiers. However, within the Monk philosophy this is only attractive if it can be realized fully autonomously, i.e., without human intervention by a machine-learning researcher who steers the training process. Current research is directed at the identification of features, classifiers and training schemes that work well for a wide range of label quantities, ranging from fully unsupervised to many thousands of examples per class in the training set.

Currently, the Monk system serves manuscripts and collections from a host of institutions: the Dutch National Archive, Czech National Archive, Dutch Royal Library, City Archives of Louvain, Belgium, Harvard University Yenchin (Chinese handwritten) collection, the Dead Sea Scrolls, in cooperation with the Israel Antiquity Authority, the Gilman collection of the Schlesinger institute, Harvard, Beowulf-era documents by Stanford University, the minutes of parliament of the Honorary Middle Temple society, London and many others.

5 Conclusion

In this article, an overview is given of a large search engine for handwritten document collections, Monk. A number of concrete requirements is introduced and the chosen solutions are presented. Although the end of the Posix file system model is predicted by some, we have shown

that this system can be lifted to the modern “Big Data” environment by using an encapsulated, metadata based file system, which present the user a regular file environment, which is implemented in a quite different manner, below the hood. Apart from the IBM gpfs file system, there are several other, similar developments, such as Luster [12] which is a GNU General Public Licence file system implementation. It is not the goal of the current paper to criticise regular database technology (it is used, “below the hood”), or newer K-V store approaches. Also, in principle, part of the computations would lend themselves quite well for the MapReduce setup. A critical implementation aspect is a reliable, scalable and fast (i.e., parallel metadata server system for mapping from name space to physical data locations. What is demonstrated by the Monk architecture is that costly software revision and adaptation could be avoided by rigorously adhering to a general, familiar working environment (with C, Matlab, Python, R, etc.) for e-Science researchers in machine learning and digital humanities. In that sense, this work may have implications beyond the particular problem of large-scale handwriting recognition, in other problem domains. For the Monk system, the challenge will be to scale up one order of magnitude in terms of the number of manuscript collections served, in the next five years.

Funding: Rijksuniversiteit Groningen, EU/SNN project Target and the Dutch Science Foundation NWO.

References

1. C. Basescu, C. Cachin, I. Eyal, R. Haas, A. Sorniotti, M. Vukolic, and I. Zachevskv. *Robust data sharing with key-value stores*. 42nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), 25–28 June 2012.
2. M. Bulacu, R. van Koert, R., L.R.B. Schomaker, and T. van der Zant. Layout analysis of handwritten historical documents for searching the archive of the Cabinet of the Dutch Queen. Proc. of 9th Int. Conf. on Document Analysis and Recognition (ICDAR 2007), IEEE Computer Society, pp. 357–361, vol. 1, 23–26 September, Curitiba, Brazil, 2007.
3. M. Cheriet, N. Kharm, C.L. Liu, and C.Y. Suen. *Character Recognition Systems: A Guide for Students and Practitioners*. Hoboken, NJ: Wiley, ISBN: 978-0-470-17652-8, 351 p., 2008.
4. https://en.wikipedia.org/wiki/Dependency_hell
5. A. Graves. *Supervised sequence labeling with recurrent neural networks*. Springer, Berlin, 2012.
6. S. He, M. Wiering, and L.R.B. Schomaker. Junction detection in handwritten documents and its application to writer identification. *Pattern Recognition*, 48, 4036–4048, 2015.
7. S. He, P. Samara, J. Burgers, and L.R.B. Schomaker. Towards style-based dating of historical documents. 14th International Conference on Frontiers in Handwritten Recognition, Crete, Greece, pp. 265–270, 2014.
8. S. He and L.R.B. Schomaker. *A Polar Stroke Descriptor for Classification of Historical Documents*. Proceedings of the IAPR 13th International Conference on Document Analysis and Recognition (ICDAR), Nancy, France, 2015.
9. R. Lämmel. Google’s MapReduce programming model Revisited. *Science of Computer Programming*, 70(1), 1–30, 2008.
10. D. Lewine. *POSIX programmers guide*. O’Reilly, Sebastopol, 611 p., 1994.
11. M. McCandless, E. Hatcher, and O. Gospodnetić. *Lucene in Action (2nd edition)*. 338 p., ISBN 978-1-933988-17-7, Manning, Greenwich, 2010.
12. [https://en.wikipedia.org/wiki/Lustre_\(file_system\)](https://en.wikipedia.org/wiki/Lustre_(file_system))
13. U.V. Marti and H. Bunke, Using a statistical language model to improve the performance of an HMM-based cursive handwriting recognition system. *Int. Journal of Pattern Recognition and Artificial Intelligence*, 15(1), 65–90, 2001.
14. M. Niitsuma, L.R.B. Schomaker, J.P. van Oosten, Y. Tomita, and D. Bell. Musicologist-driven writer identification in early music manuscripts. *Multimedia Tools & Applications*, 1–17, 2015.
15. J.-P. van Oosten and L.R.B. Schomaker. Separability versus Prototypicality in Handwritten Word-Image Retrieval. *Pattern Recognition*, 47(3), 1031–1038, 2014, DOI:10.1016/j.patcog.2013.09.006.
16. J.P. van Oosten and L.R.B. Schomaker. *A reevaluation and benchmark of hidden Markov models*. Proc. Int. Conference on Frontiers in Handwriting Recognition, September 2014, Crete, Greece, IEEE Computer Society, pp. 531–536, 2014, DOI:10.1109/ICFHR.2014.95.
17. R. Parker, D. Graff, J. Kong, K. Chen, and K. Maeda. *English Gigaword Fifth Edition*. LDC (Linguistic Data Consortium), 2011.
18. https://en.wikipedia.org/wiki/Portable_Batch_System
19. 1003.1-2008 Standard for Information Technology – Portable Operating System Interface (POSIX(R)), IEEE, <http://standards.ieee.org/develop/wg/POSIX.html>, 2008.
20. S. Sakr, A. Liu, D.M. Batista, and M. Alomari. A Survey of Large Scale Data Management Approaches in Cloud Environments, *IEEE Communications, Surveys and Tutorials*, 13(3), 311–336, 2011.
21. L.R.B. Schomaker, K. Franke, and M. Bulacu. Using codebooks of fragmented connected-component contours in forensic and historic writer identification. *Pattern Recognition Letters*, 28(6), 719–727, 2007.
22. <http://slurm.schedmd.com/>
23. M. Stonebraker. *SQL Databases v. NoSQL Databases*. *Communications of the ACM*, 53(4), 10–11, 2010.
24. <http://www.rug.nl/science-and-society/target/>
25. F. Wahlberg, L. Mårtensson, and A. Brun. Large scale style based dating of medieval manuscripts. Proceedings of the 2015 Workshop on Historical Document Imaging and Processing. ACM, New York, 2015.
26. T. Van der Zant, L.R.B. Schomaker, and K. Haak. Handwritten-word spotting using biologically inspired features. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(11), 1945–1957, 2008.
27. T. Van der Zant, L.R.B. Schomaker, S. Zinger, and H. van Schie. Where are the Search Engines for Handwritten Documents? *Interdisciplinary Science Reviews*, 34(2–3), 224–235, 2009, DOI:10.1179/174327909X441126.

Bionotes

Prof. Dr. Lambert Schomaker

Faculty of Mathematics and Natural
Sciences, University of Groningen,
Nijenborgh 9, NL-9747 AG Groningen, The
Netherlands

l.r.b.schomaker@rug.nl

Lambert Schomaker (1957) is a full professor in Artificial Intelligence at the University of Groningen since 2001. His main interest is in pattern recognition and machine learning problems. He has contributed to over 140 peer-reviewed publications in journals and books ($h = 15$ /ISI, $h = 35$ /Google Citations). His work is cited in 23 patents. In recent years his focus is on continuous learning systems and bootstrapping problems, where learning starts with very few examples. Professor Schomaker is a senior member of IEEE, member of the IAPR and is a member of a number of Dutch research programme committees in e-Science (NWO), Computational Humanities (Koninklijke Nederlandse Academie van Wetenschappen), Computational science and energy (Shell/NWO/FOM). He is the chairman of the international Unipen foundation for benchmarking of pattern-recognition systems and director of DeepLearn24 BV, a company that has as its ambition to translate current insights from computational intelligence to real-world applications.