The System for Efficient Indexing and Search in the Large Archives of Scanned Historical Documents

Martin Bulín(✉), Jan Švec, and Pavel Ircing

Department of Cybernetics, University of West Bohemia, Pilsen, Czech Republic
{bulinm,honzas,ircing}@kky.zcu.cz

Abstract. The paper introduces software capable of indexing and searching large archives of scanned historical documents. The system capabilities are demonstrated on the collection containing documents from the archives of the post-Soviet security services. The backend of the system was designed with a focus on flexibility (it is actually already being used for other related tasks) and scalability to larger volumes of data. The graphical user interface design has been consulted with historians interested in using the archived documents and was developed in several iterations, gradually including the changes induced both by the user's requests and by our improving knowledge about the nature of the processed data.

Keywords: Indexing · GUI design · Scanned documents

1 Introduction

Many institutions aiming to preserve historical documents (primarily written and graphical ones, but recently also audiovisual) have successfully adopted modern means of digitizing and storing those materials. However, making these documents accessible for scholars and even the general public still remains a serious challenge. The issues that have to be addressed are actually multifold: (a) rather sophisticated machine learning methods have to be employed to extract searchable metadata (or, at least, a machine-readable text) from the digitized material, (b) an efficient backend solution needs to be implemented for indexing and storing both the data and the metadata and (c) there is a need to design a comprehensible Graphical User Interface (GUI) to make the search feasible for both experts and general users. The paper presents our solutions to (b) and (c) as implemented in the joint research project with the Institute for Study of the Totalitarian Regimes (ISTR), Czech Republic [4]. This institution has collected and scanned over 0.5 million pages of materials from the Ukrainian and other

The work described herein has been supported by the Ministry of Education, Youth and Sports of the Czech Republic project LINDAT/CLARIAH-CZ and by the grant of the University of West Bohemia, project No. SGS-2022-017.

J. Kamps et al. (Eds.): ECIR 2023, LNCS 13982, pp. 206–210, 2023.
https://doi.org/10.1007/978-3-031-28241-6_15

post-Soviet archives to document the persecution of people by the Soviet regime. The nature of the archive presents specific challenges for the machine learning techniques mentioned above – but those are out of the scope of this paper and are extensively described elsewhere [2,3]. On the other hand, the backend solution was implemented to be very universal and has already proven its efficiency in other related tasks [1,7]. The same is true for the GUI, although naturally, to a smaller extent—effectively presenting scanned documents, of course, has different requirements than presenting a video recording. The presented demo should be interesting for all archivists and historians struggling with efficient indexation and presentation of their large, diverse (and potentially multi-medial) digitized archive materials.

2 Backend Server Implementation

The backend server provides an HTTP interface for the graphical user interface described in Sect. 3. Along with this interface, it controls and executes the data processing pipeline and interacts with the database. We designed the backend server using the experience of building our prototype system for searching audiovisual archives [5]. Unlike in this prototype, we focused on the ability to automatically process the input documents in the end-to-end pipeline and provide the web browser user interface accessible over the Internet. Two kinds of components control the overall processing of the document collection:

- *Workers* – A worker represents an atomic processing step. The workers have a consistent interface and can be executed as stand-alone processes or as part of the pipeline. An example of a stand-alone worker is an importing worker, which loads the source documents with their metadata and stores them in the document database. Other workers for converting media formats, performing OCR, or indexing OCR results are executed from the pipeline.[1]
- *Pipelines* – The pipeline is a basic principle which allows reusing different workers in different scenarios. It interconnects the workers and creates dependencies between the outputs of one worker and the inputs of another.

In other words, the workers define atomic units of work and pipelines represent the workflows. The backend stores its state entirely in the shared document database; therefore, the processing can be executed in parallel on multiple machines. Such deployment is simplified using modern containerized technologies such as Docker and Kubernetes.

The backend uses the MongoDB document database in the current implementation, storing both the processed and presented documents and the interim processing results. The backend server is implemented in Python and uses the Tornado web server for implementing the HTTP interface.

[1] The core worker in the presented demo is an OCR worker who internally uses the Tesseract OCR engine [6]. The OCR engine is wrapped with additional functionalities improving the performance of the engine in the domain of scanned documents – see the details in [2,3].

The search functionality in the pre-indexed results is implemented as part of the indexation worker. In other words, the indexation worker indexes the results (of, for example, OCR) in a specific format of an index, and at the same time, it allows to search through the index using textual queries.

The architecture of the backend server is very flexible. We use it not only for indexing the scanned documents but also for multilingual audiovisual archives [7,8] and interactive archives [1].

3 Graphical User Interface

The topmost part of the processing pipeline is the graphical user interface based on the latest web technologies, including React and Typescript. The interface is hosted on a separate Next.js server, which is independent of the backend server and the mutual communication between these two is established over the HTTP protocol. This design allows us to exploit the principal features of the Next.js framework, especially working with the client-side and server-side rendering options and optimizing the interface load time. When a user accesses the website, their web browser, in the role of the client, starts communicating with the backend server and exchanges the dynamic data only. In contrast, the static data structures are processed server-side in parallel. The website is responsive and capable of detecting the type of device - a mobile version is also provided. The application is currently accessible at https://cechoslovacivgulagu.kky.zcu.cz/en.

The crucial function of the interface is to provide a user-friendly browser of the backend data and to allow the user to search in them. Therefore, the functional design provides two fundamental modes:

Data Browser. Data are loaded from the backend server in pre-defined structures gradually, depending on the user's requests. Rendering the title page requires only the metadata of available archives, while the list of documents and more detailed data are loaded after selecting the corresponding archive. As there are many documents in total, we use pagination and load the documents in chunks. Once loaded, all the data are saved in the cache memory of the user's web browser, making them instantly accessible when requested repeatedly. The viewer of scanned documents (Fig. 1, right side) is capable of image rotation, zooming, listing documents metadata, and entering the full-screen mode.

Browser of Search Results. The application's core feature is searching for a textual phrase in a vast database of scanned documents, namely with no or a reasonable time delay. The GUI provides a search frame accepting a Unicode phrase with additional settings of the search scope with options: 1) the whole database, 2) the currently opened archive, 3) the currently viewed document. Additionally, a Russian virtual keyboard is available for entering special characters.

After sending the search query to the backend, the results are returned and listed in the lefthand side of the interface (see Fig. 1). Every single occurrence of

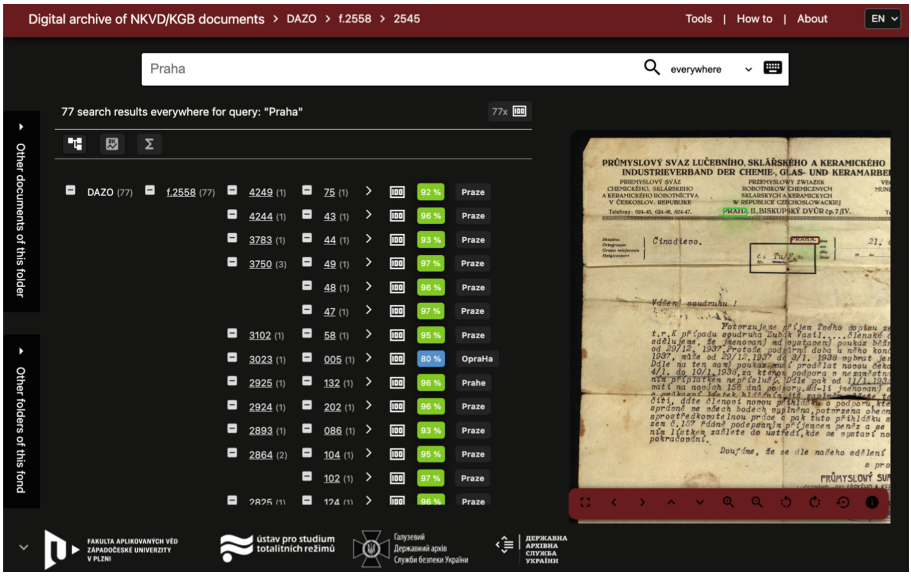

Fig. 1. GUI - showing results for search phrase "Praha" (**Lit.** *Prague*). The system found various Czech inflexion forms (*Praze, Prahy*) in many documents. Left: A tree structure of sorted results. Right: A multi-tool document viewer.

the requested phrase is returned with a confidence score. The documents then can be sorted by the highest score of contained occurrences, by the number of occurrences or hierarchically into a structured tree, which is the default option. Furthermore, results from unwanted archives can be filtered out when using the whole database scope. The list of occurrences on the lefthand side (Fig. 1) is connected to the bounding boxes in the document viewer; hence we can easily find the correct place in the image by clicking on the desired result in the list. Search results can also be downloaded into a well-arranged `.csv` file.

4 Conclusion

The presented demo system currently includes only a relatively small portion of the data collected by the cooperating historical institute ISTR. However, the backend solution has been proven to scale up in other tasks with much larger data sets. It is also important to stress that the current GUI design results from several rounds of graphical and functional design changes that were consulted with—and often initiated by—the actual target users, mainly the historians from the ISTR. Such changes include the addition of the Russian virtual keyboard or automated cropping of scanned documents.

References

1. Chýek, A., Šmídl, L., Švec, J.: Multimodal Dialog with the MALACH Audiovisual Archive. In: Proceedings of Interspeech 2019, pp. 3663–3664 (2019)
2. Gruber, I.: OCR improvements for images of multi-page historical documents. In: Karpov, A., Potapova, R. (eds.) SPECOM 2021. LNCS (LNAI), vol. 12997, pp. 226–237. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-87802-3_21
3. Gruber, I., et al.: An automated pipeline for robust image processing and optical character recognition of historical documents. In: Karpov, A., Potapova, R. (eds.) SPECOM 2020. LNCS (LNAI), vol. 12335, pp. 166–175. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-60276-5_17
4. Institute for Study of the Totalitarian Regimes (2022). https://www.ustrcr.cz/en/
5. Psutka, J., et al.: System for fast lexical and phonetic spoken term detection in a Czech cultural heritage archive. EURASIP J. Audio Speech Music Process. **2011**(1), 10 (2011). https://doi.org/10.1186/1687-4722-2011-10
6. Smith, R.: An overview of the tesseract ocr engine. In: Ninth International Conference on Document Analysis and Recognition (ICDAR 2007), vol. 2, pp. 629–633 (2007). https://doi.org/10.1109/ICDAR.2007.4376991
7. Stanislav, P., Švec, J., Ircing, P.: An engine for online video search in large archives of the holocaust testimonies. In: Proceedings of Interspeech 2016, pp. 2352–2353 (2016)
8. Zajíc, Z., et al.: Towards processing of the oral history interviews and related printed documents. In: Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018). European Language Resources Association (ELRA), Miyazaki (2018). https://aclanthology.org/L18-1331