Journal of
open research software

Damerow, J et al 2017 The Giles Ecosystem – Storage, Text Extraction, and OCR of Documents. *Journal of Open Research Software,* 5: 26, DOI: https://doi.org/10.5334/jors.164

SOFTWARE METAPAPER

The Giles Ecosystem – Storage, Text Extraction, and OCR of Documents

Julia Damerow, B. R. Erick Peirson and Manfred D. Laubichler

Arizona State University, US

Corresponding author: Julia Damerow, Scientific Software Engineer (jdamerow@asu.edu)

In the digital humanities, there is a constant need to turn images and PDF files into plain text to apply analyses such as topic modelling, named entity recognition, and other techniques. However, although there exist different solutions to extract text embedded in PDF files or run OCR on images, they typically require additional training (for example, scholars have to learn how to use the command line) or are difficult to automate without programming skills. The Giles Ecosystem is a distributed system based on Apache Kafka that allows users to upload documents for text and image extraction. The system components are implemented using Java and the Spring Framework and are available under an Open Source license on GitHub (https://github.com/diging/).

Keywords: Text extraction; OCR; Document storage; Apache Kafka; Java; Spring Framework
Funding statement: Funding was provided by grants from NSF SES 1656284, ASU Presidential Strategic Initiative Fund and the Smart Family Foundation.

(1) Overview

Introduction

In the (Digital) Humanities data usually consists of text, image, audio, or video files. Often researchers travel to archives in order to access documents, conduct their research, and make electronic copies if permitted. Combined with a growing number of archives and libraries providing access to documents in electronic form, in addition to services such as Google Books or JSTOR, researchers in the digital humanities are increasingly interested in using digitized documents to run computational analysis algorithms and provide novel ways of exploring and interacting with the documents. For textual documents, however, this first requires the documents to be in the form of text files rather than images or PDFs [7]. Hence, an often asked question by digital humanities scholars is how to transform large sets of PDF and image files into text files they can feed to their text analysis software.

While there exist several tools to extract text from PDFs (such as Adobe Reader [2] or pdftotext [6]) or run OCR on images (such as ABBYY FineReader [1] or Tesseract OCR [13]), this is not a straightforward process. It often requires a scholar to know how to use a command line or programming language, or to buy commercial software. There are several tutorials describing the workflow for humanities scholars (see for example [10] or [12]); however, especially when working with corpora comprised of several hundred or thousand documents, and when facing errors in the conversion process, many scholars are not equipped with the skills and knowledge to complete the task. In addition, after extracting text from a document, all related files such as the document itself and the extracted text often need to be stored and made available to other services. Those other services include tools for computational text analysis (such as topic modelling algorithms) or metadata repositories that enable documents to be searchable and available online.

Some libraries developed solutions for this problem by customizing or extending existing repository solutions to incorporate an OCR workflow (see for example [8] or [11]). However, those solutions are usually specific to the developing institution, reflecting the institution's metadata standards and workflow specifics. Adapting them to new projects typically requires software developers or frontend designers, which smaller projects often lack.

The development of the Giles Ecosystem was based on the need for a system that would allow users to upload documents for storage, text and image extraction, and OCR that could easily be integrated with existing applications. Specifically, the Giles Ecosystem was based on the following requirements:

- The system should allow users to upload (several) documents. For uploaded images, OCR routines should be run. For uploaded PDF files, embedded text should be extracted, for each page an image should be created, and OCR routines should be run on each page.

- The system should provide a REST API to allow other applications to upload and retrieve documents.
- The system should be agnostic in regards to a document's metadata. Given that different projects prefer different metadata standards and many already have a system in place to capture those, the Giles Ecosystem should not provide metadata management features. Instead it should provide a simple REST API to retrieve documents. Document metadata would be stored in an external system.
- The system should be able to handle varying workloads. Especially in the beginning phase of a project, often many documents have to be processed (several hundreds to thousands) followed by only few documents uploaded sporadically. Hence, the system should be easily scalable as the need arises.
- The system should provide single sign-on authorization using OAuth or OpenId Connect.
- The system should provide access to images through the image viewer software Digilib (http://digilib.sourceforge.net/) but add authorization checks.

At the time of writing this paper, two projects are using the Giles Ecosystem. The Digital Innovation Group at Arizona State University developed an application for metadata management called *Amphora* (https://github.com/diging/amphora). It integrates with the Giles Ecosystem and allows users to upload documents, collect metadata, and use the extracted text from the documents sent to the Giles Ecosystem for computational text analysis. Another project is located at the Max Planck Institute for the History of Science in Berlin. This project developed a document manager that allows images to be uploaded and annotated. The images can be displayed along with the text extracted by the Giles Ecosystem (through OCR).

Implementation and architecture

The Giles Ecosystem has been developed using Java and the Spring Framework. It consists of five components (Giles, Nepomuk, Cepheus, Andromeda, and Cassiopeia) that communicate through an Apache Kafka [3] instance, which relies on Apache Zookeeper [4]. The Giles Ecosystem utilizes the image viewer software Digilib [5] to serve uploaded or extracted images. A relational database such as MySQL [9] is used to store information about uploaded files. Moreover, Giles provides an API for external metadata management apps to integrate with the Giles Ecosystem. **Figure 1** shows the architecture of the Giles Ecosystem. The following sections will describe each component in more detail.

Giles

The Giles app allows users or applications to upload files for processing and to retrieve the processed results. Its main role is the coordination of the extraction process. When a user uploads a file, Giles sends a storage request to Apache Kafka, which will route the request to Nepomuk for processing. Once a file has been stored, if the file is a PDF file, Giles will send a text extraction request and then an image extraction request to Kafka. Image extraction requests are handled by Cepheus. Text extraction requests are handled by Andromeda. For each image file that has either been uploaded to Giles or extracted from a PDF, Giles will submit an OCR request that is processed by Cassiopeia.

Besides coordinating the extraction process of files, Giles is the user-facing component of the Giles Ecosystem. This means that Giles provides all the required webpages for uploading and browsing files (see for example **Figure 2**), an API for the integration of external applications, user management functionality, and permission checks on files. When an image file is requested from Giles, it

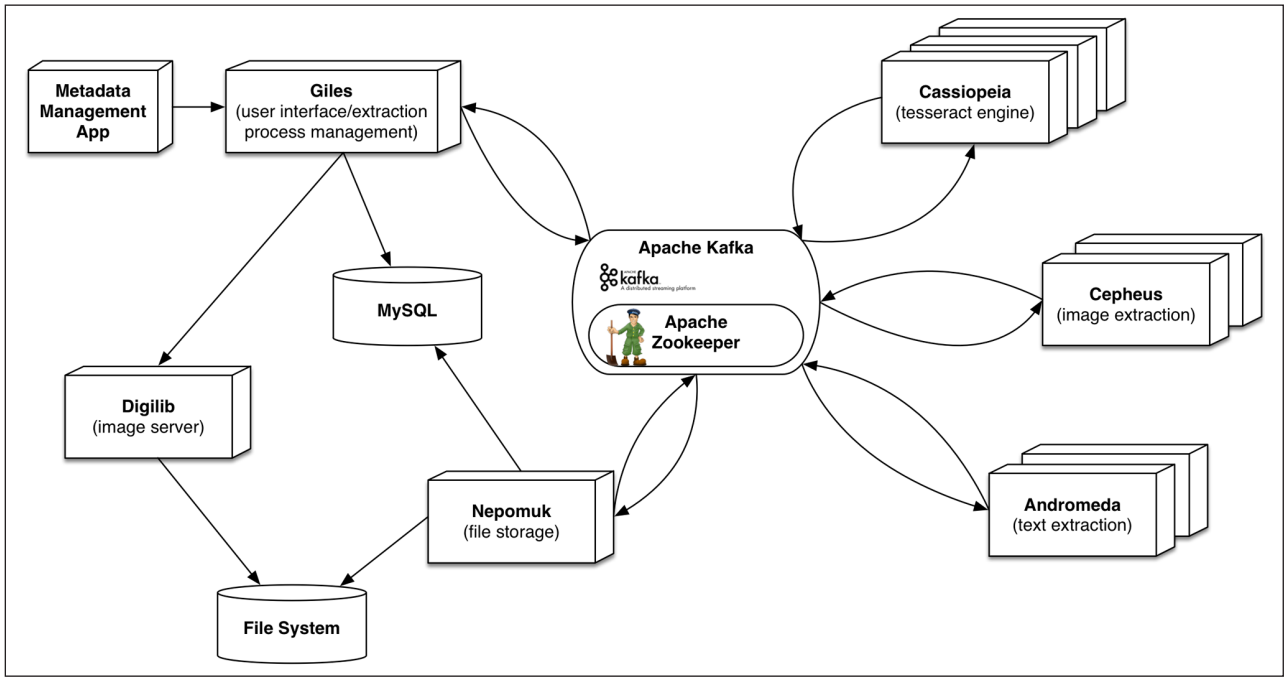

Figure 1: Giles Ecosystem Architecture.

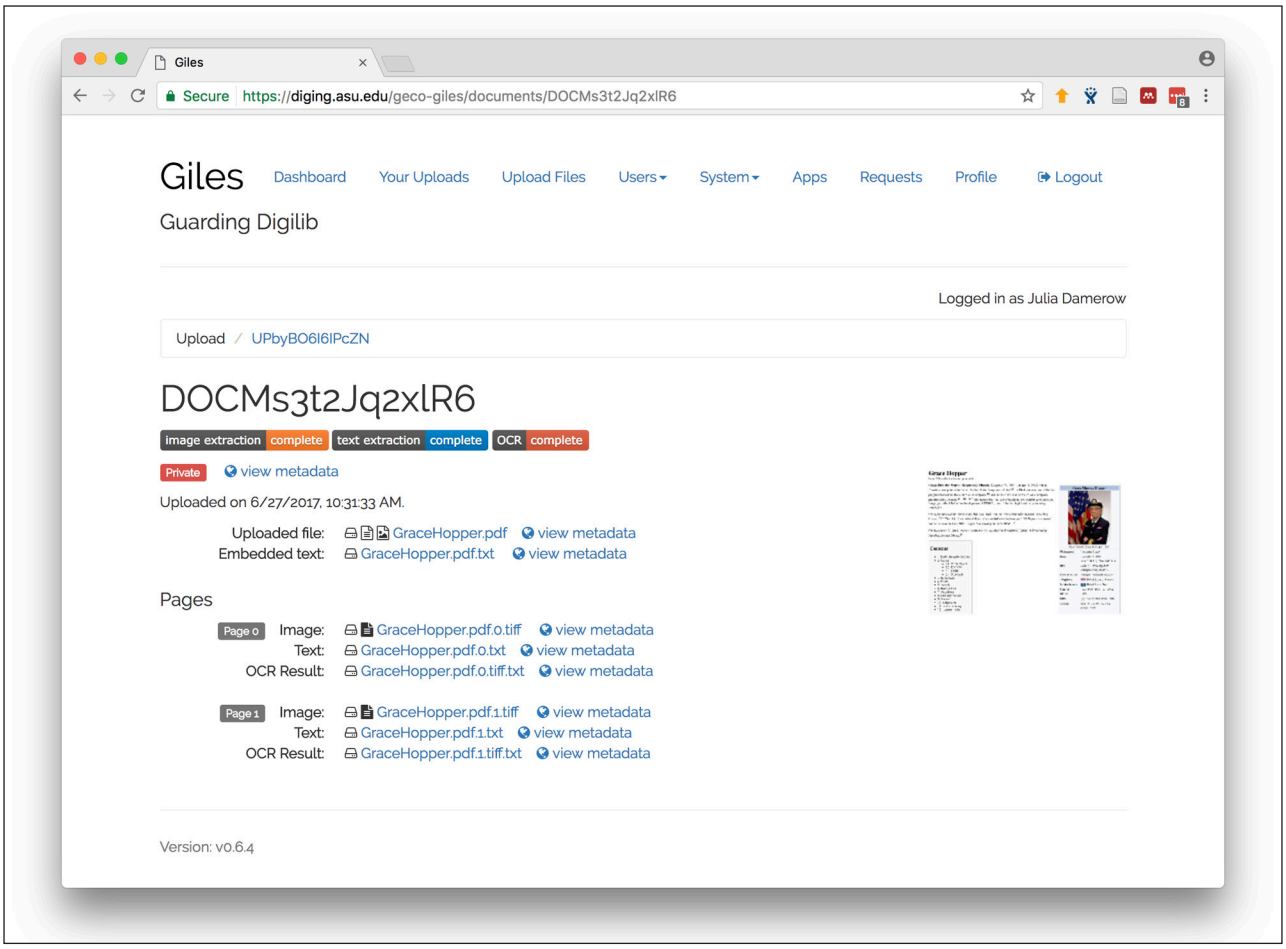

Figure 2: Webpage for an uploaded and processed pdf document.

runs permission checks on the file and if the checks pass forwards the request to Digilib, which prepares the images according to the request (e.g. zooms in and clips the image).

Giles is implemented using the Spring Framework. It uses Apache Tiles (https://tiles.apache.org/) for page layout and rendering. Spring Security (https://projects.spring.io/spring-security/) is used for access control, and Spring Social (http://projects.spring.io/spring-social/) is used to allow single sign-on with GitHub, Google, or the MITREid Connect server (https://github.com/mitreid-connect/).

Nepomuk

Nepomuk is the storage component of the Giles Ecosystem. It listens to Apache Kafka for storage requests. When receiving a storage request, Nepomuk downloads the file to be stored from the URL provided in the request and stores it in the file system. Digilib has access to the folders in which Nepomuk stores uploaded files and can therefore serve stored image files. Nepomuk then sends a storage request completion message to Kafka, which is received and handled by Giles.

Nepomuk provides an API to download stored files, which is used by Giles when serving files to users and applications. It also has a simple user interface that allows administrators to configure Nepomuk. Like Giles, Nepomuk is built using the Spring Framework. It uses Apache Tiles for webpage layout and rendering and Spring Security for access control.

Cepheus

Cepheus extracts images from PDF documents. It handles image extraction requests sent through Kafka. When receiving a request, Cepheus downloads the file from the URL provided in the request and then uses Apache PDFBox (https://pdfbox.apache.org/) to turn each page of the PDF into an image. When a request is completed, Cepheus sends a completion message back to Kafka, which is received by Giles. Cepheus is developed using the Spring Framework and Apache Tiles.

Andromeda

Andromeda extracts embedded text from PDF documents. It handles text extraction requests sent through Kafka. When receiving a request, Andromeda downloads the file from the URL provided in the request and then uses Apache PDFBox (https://pdfbox.apache.org/) to extract embedded text. When a request is completed, Andromeda sends a completion message back to Kafka, which is received by Giles. Andromeda is developed using the Spring Framework and Apache Tiles.

Cassiopeia

Cassiopeia is the OCR component of the Giles Ecosystem. It handles OCR requests sent through Kafka by running the OCR software Tesseract (https://github.com/tesseract-ocr) on images. It uses Apache Tika (https://tika.apache.org/) as a bridge to Tesseract and is built on the Spring Framework and Apache Tiles. When an OCR has been completed, Cassiopeia sends a completion message to Kafka, which is received and handled by Giles.

Scaling Up

Cepheus, Andromeda, and Cassiopeia are the most computationally and resource intensive components of the Giles Ecosystem as they process PDFs and perform OCR. A Kafka-based architecture lends itself nicely for scaling up as the need arises. If a higher number of uploads is expected, Cepheus, Andromeda, and Cassiopeia can be scaled horizontally by adding more instances. Requests will then be distributed across several instances rather than being processed one-by-one.

Quality control

The code has been tested using unit tests. In addition, a limited amount of load testing has been done by making upload requests to the Giles API. The load tests created 200 upload requests over a period of 3 minutes. Uploading files using the API requires polling for upload results. This resulted in about 420,000 requests distributed over approximately one hour (around 110 requests per second). The average response time for the initial upload requests was 230 ms. Polling requests were answered with an average response time of 5 ms.

Load tests were executed on a development machine with a 2.9 GHz processor and 16 GB of memory. All components of the Giles Ecosystem were run on the same machine. In a production environment, each component would ideally have its own machine, conceivably resulting in higher throughput and shorter response times.

A Docker Compose file has been created to easily set up the Giles Ecosystem for testing and evaluation purposes. It can be found at https://github.com/diging/giles-eco-docker. Depending on network speed and hardware, building the docker images might take some time (up to several hours). On a machine with a 2.9 GHz processor and 16 GB of memory, the build process takes about 40 minutes. Once set up (instructions are provided in the readme file), the system can be tested by uploading a PDF file through the user interface. An example PDF is provided in the GitHub repository. When processing of a file has successfully finished, there should be at least one image and one text file (created through OCR) for each page available on the document page for the uploaded file. If an uploaded PDF has embedded text, there should be one additional text file for the whole document, and one text file for each page.

(2) Availability

Operating system

When using Docker, the Giles Ecosystem should run on every system capable of running Docker. If running directly on a machine, the Giles Ecosystem should run on every system with Java 8, Apache Tomcat, and Tesseract installed. The Giles Ecosystem has been tested on Mac OS X El Capitan, macOS Sierra, Red Hat Enterprise Linux, and Ubuntu.

Programming language

Java 8

Additional system requirements

In the current setup of the Giles Ecosystem, Giles is the only component that has to handle high numbers of concurrent requests. Hence, number of cores, CPU speed, and RAM size can greatly affect Giles' performance. The exact configuration has to be determined for each system individually depending on expected traffic. Nepomuk requires sufficient disk space to store uploaded files (recommend are a minimum of 500 GB for production). Please consult the Apache Kafka and Apache Zookeeper documentations for requirements and recommendations for Kafka and Zookeeper.

When running the Giles Ecosystem using Docker, 16 GB of memory and a 2.2 GHz or higher processor are recommended. The environment is likely to build with less memory and on slower processors, however, build times increase and processes can slow down considerably. In production environments, it is highly recommended to run the Giles Ecosystem directly on several machines rather than using Docker.

Dependencies

The Giles Ecosystem requires either Docker/Docker Compose or Apache Tomcat 8 (for each component), Apache Zookeeper, Apache Kafka, MySQL (for Giles and Nepomuk), and Tesseract (installed on the machine running Cassiopeia).

List of contributors

Student workers of the Digital Innovation Group, Arizona State University (software development).

Software location

Code repository

Name: GitHub
Identifier:https://github.com/diging/giles-eco-giles-web
Licence: Mozilla Public License Version 2.0
Date published: Latest release is 01/06/17

The Giles Ecosystem consists of several GitHub repositories. All repositories are linked from the above repository.

Language

English

(3) Reuse potential

The Giles Ecosystem can be used in any project that requires plain text files extracted from images or PDFs. The system can be integrated with other applications that, for example, manage metadata of uploaded documents or use extracted text files for analysis and computation. Due to the Apache Kafka-based architecture it is also relatively easy to add other applications to

the processing pipeline. New listener groups can be registered with Kafka that, for example, listen to text extraction completion messages or storage completion messages without interfering with existing applications. For example, one could allow another application to listen to text extraction completion messages in order to automatically run named entity recognition algorithms on extracted texts or add extracted texts to an indexer like Solr.

The Giles Ecosystem is being developed and maintained by the Digital Innovation Group. To report problems or ask questions, users can create issues in GitHub (https://github.com/diging/giles-eco-giles-web/issues) or contact the author JD directly. Issues directly related to the Docker Compose environment can be reported in the corresponding GitHub repository (https://github.com/diging/giles-eco-docker/issues).

Acknowledgements

Thanks to Dirk Wintergrün (Max Planck Institute for the History of Science, Berlin) for installing and using the Giles Ecosystem for his projects. His input is important for development goals and improvement of the software. We also thank Bryan Daniels (Arizona State University) for reviewing the manuscript and providing helpful comments.

Competing Interests

The authors have no competing interests to declare.

References

1. **ABBYY** 2017 ABBYY FineReader [computer software]. https://www.abbyy.com/en-us/finereader/.
2. **Adobe Systems** 2017 Adobe Acrobat Reader [computer software]. https://get.adobe.com/reader/.
3. **Apache Software Foundation** 2017 Apache Kafka [computer software]. https://kafka.apache.org/.
4. **Apache Software Foundation** 2017 Apache Zookeeper [computer software]. https://zookeeper.apache.org/.
5. **Casties, R** and **Raspe, M** 2017 digilib [computer software]. http://digilib.sourceforge.net/.
6. **Glyph & Cog, LLC** 2014 pdf to text; distributed as part of Xpdf [computer software]. http://www.foolabs.com/xpdf/home.html.
7. **Hockey, S M** 2000 Electronic texts in the Humanities. Oxford University Press. Chapter 2, Creating and Acquiring Electronic Texts, 11–23. DOI: https://doi.org/10.1093/acprof:oso/9780198711940.003.0002
8. **Huculak, J M** and **Justice, B** Report for the University of Victoria Libraries on Fedora Commons-Based DAMS: Building Collaborative Scholarship Environments, A Test Case. Available from: http://hdl.handle.net/1828/7212 [Accessed 16th December 2016].
9. **Oracle Corporation** 2017 MySQL [computer software]. https://www.mysql.com/.
10. **Peirson, E B** 2015 Tutorial: Text Extraction and OCR with Tesseract and ImageMagick. Available from: https://diging.atlassian.net/wiki/display/DCH/Tutorial%3A+Text+Extraction+and+OCR+with+Tesseract+and+ImageMagick [Accessed 15th December 2016].
11. **Princeton University Library** Plum: A Hydra head to support digitization workflows. Available from: https://github.com/pulibrary/plum [Accessed 16th December 2016].
12. **Schmidt, B** Tutorial: Command-line OCR on a Mac. Available from: http://benschmidt.org/dighist13/?page_id=129 [Accessed 15th December 2016].
13. **Smith, R,** et al. 2017 Tesseract OCR [computer software]. https://github.com/tesseract-ocr/tesseract.

How to cite this article: Damerow, J, Peirson, B R E and Laubichler, M D 2017 The Giles Ecosystem – Storage, Text Extraction, and OCR of Documents. *Journal of Open Research Software,* 5: 26, DOI: https://doi.org/10.5334/jors.164

Submitted: 13 February 2017 **Accepted:** 15 September 2017 **Published:** 28 September 2017

]u[*Journal of Open Research Software* is a peer-reviewed open access journal published by Ubiquity Press

OPEN ACCESS