Geosoftware II, WiSe 2018/19
Enhancing discovery of geospatial datasets in data repositories

Daniel Nüst, Edzer Pebesma

November 21, 2018

1 Introduction

Geosoftware II provides a challenge to groups of geoinformatics students to apply their technological and conceptual skills. Students cooperate to solve a realistic and relevant spatial issue with computer science methods. This semester, the challenge lies in extending existing software projects with geospatial capabilities and quickly break in to the areas of geospatial catalogues and repository platforms.

The teachers take the role of the *customer*, depending on the chosen base platform either a data repository operator or a geospatial catalogue operator. The customer publishes a problem statement and an invitation to bid[1] for solving the problem. The student groups are the *contractor*, i.e. the architects, designers, and developers of a software system to advance the state-of-the-art in scholarly discovery. Each group bids with their own approach to the project, implements it, and presents it to the customer.

Veranstaltungsnummer im Vorlesungsverzeichnis: 144929

[1] https://en.wikipedia.org/wiki/Invitation_for_bid

2 Invitation to bid

2.1 Problem statement

A large value, or potentially a trap for the consumer, in online shopping lies in recommendation systems. These systems suggest similar products based on all or specific customer's shopping behaviour. In a similar vein, discovering related works and similar research is a challenge for scientists. The challenge applies to their own work (who is working in my area of expertise or region of interest?) and finding relevant literature (what publications exist on topic X?).

With the increased requirements by funding agencies to publish data in open repositories, this project should tap into the wealth of published research data and provide new ways to connect and discover datasets based on their geospatial properties. By extending our catalogue platform and data repository, we except to gain a significant advantage over competitors.

2.2 State-of-the-art

The discovery of scientific publications today is largely based on full text & keyword search, editorial decisions (i.e. scholarly journals), social networks (Twitter, word of mouth), and citations. If a researcher is lucky, a literature survey for the topic of interest exists, which may have utilised all of the above methods. Academic search engines, such as Google Scholar or ScienceOpen, allow researchers to harness full text search capabilities across millions of articles. However, full text search is often limited to titles, authors, and abstracts, as the copyright of articles often does not allow free access. The available data often stems from citation databases (e.g. CrossRef, DOAJ). They have structured metadata on scholarly publications, largely due to the need to properly reference other works and the need to track impact of an article. Google recently announced[2] a new service for Dataset Discovery (https://toolbox.google.com/datasetsearch) but it is a closed system[3] In general, these databases do not include or expose spatial and spatio-temporal metadata or specific metadata for parts of a research project, e.g. licenses for related data or code. Kmoch et al.[4] demonstrate extraction of spatial information from full texts of research articles by using geocoding, but such processes are not common.

It is also widely acknowledged that geospatial data poses specific challenges for archival and preservation, see e.g. Janée et al. (2008)[5] and Clark (2016)[6] The quality of existing publication metadata is usually good, if an article is published in a journal or as a preprint, due to the editorial process of journals. During this process all metadata is provided by author and editorial staff in a manual fashion. Because the practices of Open Science become increasingly

[2] Cf. https://www.blog.google/products/search/making-it-easier-discover-datasets/

[3] See Kraker P. (2018). *Google is capitalizing on a movement that they have contributed nothing to.* Elephant in the Lab. https://doi.org/10.5281/zenodo.1434695

[4] https://meetingorganizer.copernicus.org/EGU2018/EGU2018-14542.pdf

[5] https://dl.acm.org/citation.cfm?id=1378912, PDF download

[6] https://doi.org/10.1080/15420353.2016.1185497

widespread, data and code are often published alongside research articles, either as supplemental material or in the form of a research compendium. But these metadata rarely include spatio-temporal information.

Public and free data repositories provide safe storage and identification of data and files using DOIs, for example Zenodo, OSF, or b2share. These data repositories are mainly file dumps, i.e. they provide appropriate access (e.g. syntax highlighting, previews) and management (e.g. version control), but do not process contents of the files to leverage contained information.

On the geospatial side, geospatial catalogues provided standardised access to structured metadata based on standardised formats (i.e. OGC CSW). GDAL is the core library for reading and writing many different data formats. It is used by almost all geospatial software projects and can be integrated in any programming language.

Document databases (e.g. Elasticsearch) index spatial data and allow sorting of search results by distance.

None of the mentioned platforms or websites utilise geospatial (meta)data, or expose similar works on the same platform using text, data, or metadata in their UI or API.

2.3 Project goals

This project will close the gap between geospatial data formats and repositories respectively geospatial metadata catalogues and similarity measurements.

Project groups will extend an existing Free and Open Source Software (FOSS) project with the functionality to retrieve and view similar records. This comprises both the API and UI, namely providing an HTTP endpoint to retrieve an ordered list of records based on a provided record and displaying/linking similar records in a detail view of a record respectively.

Two base software projects are possible:

- a repository platform
- a geospatial catalogue

We make the following assumptions:

1. the number of scholarly analysis leveraging geospatial data increases
2. geospatial aspects of data have a unique potential to integrate and connect works and people across disciplines (transdisciplinarity)
3. developers and operators of data repository solutions do not have the necessary expertise or resources to evaluate the potential of geospatial (meta)data for discovery
4. the customer operates both a repository platform and a geospatial catalogue, but the contractor is free to suggest a base platform to extend within the boundary conditions, see 2.4.4.

The new system must take into consideration different target groups:

- scientists searching for similar or related work (during research, writing manuscripts, or evaluating papers)
- preservationists improving access to research outputs (data, articles)
- operators of data repositories or catalogue services who want to easily integrate similarity measures in their deployment

Further requirements and the process to fulfil them are detailed in the remainder of this document.

2.4 Requirements

2.4.1 Functional requirements

The main functionality of the web-based platform is browsing and searching records with geospatial properties.

Extraction

FE001
Supported geospatial (meta)data formats: GeoPackage, NetCDF, GeoJSON, Shapefile, CSV on the Web, ISO 191xx, GeoTIFF

FE002
A CLI tool to extract geospatial extent at different levels of detail (bounding box or a single feature, i.e. polygon, line, point) from a single file

FE003
A CLI tool to extract the temporal extent from a single file

FE004
A CLI tool to extract geospatial and temporal extent from a directory of files

FE005
Metadata extraction for a specific record can be triggered via an API call by all logged-in users, which immediately updates the record's metadata

FE006
Metadata extraction is automatically triggered for new uploaded records to the base software

FE007
Metadata extraction during creation of a new record runs as an independent process (i.e. it does not need to complete for record creation to complete)

API

FA001
All user-facing functionality is available via RESTful HTTP API endpoints

FA002
API endpoints return valid JSON in responses, including errors

FA003
API endpoints use appropriate HTTP status codes

FA004
Geospatial data in the API is encoded using GeoJSON (RFC 7946)

FA005

Enhanced metadata, i.e. including the temporal and geospatial information extracted from files, are included in the regular metadata for records (no special endpoint)

FA006

With the parameter `similar=n` added to a request to read a record, the response is enhanced with ids and similarity scores for `n` many similar records

Similarity calculation

FS001

API endpoint providing the similarity score of two records based on the bounding box of all data in the record; records are provided as their repository-specific ID

FS002

API endpoint providing a sorted list of similar records for a repository-specific record ID; the length of the returned list can be defined by the user, a maximum length can be configured server-side

FS004

The similarity value is normalised in the interval $[0, 1[$

FS005

The input record is never included in the list of similar records

FS006

The similarity value takes the data type into consideration for the types `vector`, `raster`, or `timeseries`, i.e. a similarity value for two records with same extent is higher if data types match as well

UI

FU001

A configurable number of similar records is displayed on a page for viewing a single record (must not be integrated with existing UIs for the base software)

Configuration

FC001

All configuration of additional functionality is possible via plain text files, e.g. YAML format, and ideally integrated with configuration mechanisms of the base software

FC002

The configuration is at least active after restarting the service

Bonus feature One bonus feature from the following list or own ideas for features must be completed.

- extend existing UI to show similar datasets in a search result display
- include source code or text in similarity calculation
- use > 1000 real world records in a demo deployment
- similarity supports fuzzy matches and advanced geometries (beyond bounding box)
- Create a static yet reproducible infographic of spatial and temporal properties of all records, on-demand via an API
- geospatial search and map-based browsing of records
- The similarity score is configurable per request, e.g. weights can be assigned to aspects such as data type, extent, and the type of the score is configurable (at least two different algorithms)

Bonus feature decisions must be submitted one month before final submission and must be formally accepted by the customer. The chosen feature must be mentioned clearly in the project documentation and underlies the same non-functional requirements as regular features.

Only one contractor may implement each feature (first come first serve).

2.4.2 Test data

An appropriate number of test datasets must be listed in the bid, but there must be at least one record per supported file format. Test data must be published under an open licenses and its provenance must be documented clearly. Test and demonstration data should be taken from existing online repositories to ensure practical relevance, realism, and geographical spread.

Examples for dataset searches:

- "Open" "Nc" on Zenodo
- "geotiff" on https://pangaea.de
- "shapefile" on B2SHARE
- "geopackage" on Figshare

2.4.3 Non-functional requirements

Maintainability The developed software must be published as open source under an OSI-approved license according to the license requirements and taking into account the licenses of other used or extended software. The contractor must ensure license compatibility.

An established build and configuration system as well as dependency management must be applied for the used Programming language, e.g. Gradle for Java, or npm for JavaScript. According documentation on building, configuring and installing the system must be provided in Markdown-formatted documentation files using the appropriate markup for lists, links, etc.

User friendliness The system must support intuitive use to the extend that targeted groups can use its main (non-bonus) features without further documentation. Common practices for web-based user interfaces and interaction paradigms should be applied.

Performance The response time of an API call to a repository may increase no more than a factor of 2 when related projects are listed, as compared to not showing related projects.

In general, a next page must be displayed within 1 second to allow users to stay focused on their current train of thought; complex page contents may be loaded asynchronously; interactive visualisations must react within 0.1 seconds to give a user the impression of direct manipulation.[7]

The contractor provides a test script to evaluate the performance using at least 10 different URLs or views with an appropriate number of repetitions.

2.4.4 Deployment

Docker must be used to ensure easy deployment. Used images must be based on Dockerfiles and hosted on Docker Hub (or comparable). If base projects have existing Docker images or Dockerfiles, they should be extended.

Boundary conditions The language of the user interface is English.

One of the following projects must be used as a basis. Other base projects must be formally accepted by the customer.

- Invenio
- DSpace
- GeoNetwork (CSW)
- pycsw (CSW)
- sat-api (STAC)

[7] Source: `http://www.nngroup.com/articles/powers-of-10-time-scales-in-ux/`

2.5 Project management

The contractor shall apply an agile development process, e.g. oriented at Scrum. The customer must be given access to the contractor's online task management to observe the progress. The contractor may include the customer in regular meetings for updates and feedback on the progress.

2.6 Delivery contents

The **bid** (Pflichtenheft[8]) must be submitted to the customer via WWU's Learnweb no later than **October 31st, 14:00 CET**. It must comprise an implementation plan for all required features from the invitation to bid (Lastenheft) as well as a schedule for the implementation as a single PDF document. The bid document may be English or German. Its content should follow common standards for content and structure of bids.

The **final delivery** must be submitted one day in advance of the final presentation (to be scheduled). It consists of the following items:

1. project report (using screen-shots, referencing requirements from this document, including the references to all following items) as a single PDF document, submitted via Learnweb
2. commented source code on zivgitlab, GitLab.com, or GitHub.com in one repository; delivery in more than one repository must be formally accepted by the customer
3. ready-to-use Docker images (if applicable with docker-compose configuration)
4. installation documentation
5. integrated user documentation (i.e. within the regular UI)
6. API test suite (e.g. as documented `curl` requests or a Postman/SoapUI project)
7. operational installation on a server (provided by the customer)

2.7 Acceptance criteria

The acceptance criteria encompass the fulfilment of all functional and non-functional requirements as described in the concept and schedule, and the complete and on-time delivery of all items listed in section 2.6.

[8] https://de.wikipedia.org/wiki/Pflichtenheft, https://wiki.induux.de/Pflichtenheft, and http://www.infrasoft.at/downloads/Anleitung_zum_Pflichtenheft.pdf

3 Seminar organisation

3.1 Learning objectives

1. make project management experiences, including group work, task management, coordination
2. learn to work for a customer, in a process similar to typical business procedures (invitation to bid, communication, submission)
3. extend established software projects and re-use libraries where appropriate
4. handle large open source projects (install, maintain, develop further, even in new programming languages)
5. develop specific features as small independent open source projects
6. deepen knowledge and skills in software development (e.g. collaboration platform GitLab)

3.2 Timetable

The seminar consists of five phases spread over the whole winter semester and the prior semester break.

Phase	Content	Dates
Initial training	get to know a relevant topic and prepare it for the fellow seminar members in a handout and presentation (10 mins presentation + 5 mins discussion lead by presenters); customers present *invitation to bid* at first meeting; questions on bid and groups announced at second meeting	first two meetings
students are split up into teams; deadline for project plan and final presentation date are announced		
planning	project groups create concept and schedule for the implementation of the tender	two weeks/meetings
implementation	project groups implement their solution, revising documentation and concepts along the way; special phases: project peer review (around Christmas/New Year), pre-release (three weeks before final presentation)	about 11 weeks/meetings as needed
presentation	project groups present their solution to the customers, peers, and guests	final course meeting (end of January)
report	each student creates short report on contributions and lessons learned	mid February

3.3 Efforts and grading

9 ECTS correspond to 270 working hours. Assuming 24 hours for preparing and attending initial & final presentation and 10 hours for the individual report and the peer review, that leaves 236 hours over 10 weeks (not counting Christmas holidays) for the implementation. Assuming 16 hours preparation during the semester break this gives an average workload of 16.9 **hours per week during the semester** for each student.

What	Work load (hrs)	Grade contribution
initial training, presentations & report (talk preparation, attendance, final presentation, personal report)	36	20%
planning (Pfilchtenheft, Zeitplan)	32	15%
implementation (incl. technical documentation, peer evaluation, demonstration, regular meetings)	200	65%

Grading is based on code contributions on the software development platform. Therefore, we require each student to contribute under their **own account**.

A fork & pull development model is highly encouraged because it allows to list the relevant pull requests in a final **personal report** presenting *learning achievements*, *role in the team*, *personal contributions* in a single PDF document (max. 3 pages).

We *strongly* encourage vertical splitting of tasks (a student works on UI, backend, and installation) instead of horizontal (one team member does maintenance, another backend, a third frontend) even at the cost of effectiveness.

3.4 Initial training

See **https://github.com/Geosoft2/geosoft2-2018** for topics, submission process, and requirements for the initial training sessions.

1. scholarly publishing: history and future
2. academic search engines
3. data publishing requirements in science
4. publication metadata
5. research data lifecycle & best practices
6. metadata extraction
7. FOSS repositories & preservation
8. geodata catalogues & geospatial metadata
9. geospatial data formats & libraries
10. time series data formats & libraries
11. spatial similarity calculation
12. agile collaborative software development