Universität zu Köln

Historisch-Kulturwissenschaftliche

Informationsverarbeitung

Spatial (Meta-)Data Extraction and Visualisation for Data Discovery and Exploration in an Archeology Archival Application

Bachelor Thesis

Submitted by: Hendrik Schmeer

Matriculation number: 5423988

Advisor: Dr. Øyvind Eide

Submitted: 2017-01-10

Contents

1 Abstract

The software developed for this bachelor thesis extracts, collects and visualizes spatial data and metadata from archaeological file collections of various media types. It aims to help exploring or querying data collections with respect to their locational information. In the text, not only the applied extraction strategies and their limitations are discussed, but also the bridges between archeology, information retrieval and information extraction and their impact on spatially enhancing research within the humanities.

2 Introduction

The considerations concerning this bachelor thesis started out within the context of a working group called IANUS[1], which creates long term archive infrastructure and a research data center of archeology and ancient studies. It is organized under the German Archaeological Institute[2] as umbrella organisation.

Archaeologists often work with maps, because the discipline has strong ties to the spatial dimension, besides the temporal. Arranging findings and research results along their spatial structure would lead to the production of new maps or geographical information systems (GIS). They also utilize a great variety of data formats, from textual representations over images to tables, databases etc.. Some of these media types hold genuine geographical information.

The most important question being asked in this thesis is, how research questions within the archeology and the humanities at all can be extended towards spatiality. If it is possible to provide a means to arrange or query a data collection according to their distribution throughout geographical space, new questions emerge.

As archaeological research utilizes a great variety of neighboring disciplines from the humanities and sciences, the final amount of data types is tremendous. When it comes to archiving the different media formats, it is a great challenge to extract spatial data and metadata from them, in order to derive the collections spatial structure from them.

An example for a media type holding spatial metadata are geo-tagged images. In archeology, also vector files intended for use in a GIS can often be found in the collections. Many media types do not hold spatial information, that can as easily be retrieved and evaluated for spatial indexing: the neighboring disciplines of the humanities have a well established culture of textuality; and textuality at all plays an important role in almost every scientific discourse. Articles in most cases are the preferred way of exchange and

[1] IANUS - Forschungsdatenzentrum Archäologie & Altertumswissenschaften, http://www.ianus-fdz.de. It is not an acronym, the double-faced divinitie has been chosen as the project's patron.

[2] Deutsches Archäologisches Institut (DAI), http://www.dainst.org

communication. But how can an archive be scanned for a specific geo-spatial region of interest, without spending too much work on the manual creation of spatial indices? Because of that, strategies exist to extract information from weakly structured data, such as text documents.

With regard to archiving, most data providers do not make any provisions to facilitate the generation of indices by providing rich metadata, nor can they be expected to be aware of the problem.

While sometimes provided files hold the spatial information of a collection, it is still a challenge for an automated process to make their contents available. This is especially the case, when the extraction tool is not a genuine implementation of the application that produced the data. So it actually remains to be the work of the archivists to extract and facilitate the obtained spatial resources.
Regarding the technical implications, which are addressed by this bachelor thesis, the following questions emerge:

1. How to retrieve spatial data from different levels: the file name, the file metadata, the file content body?

2. How can spatial references, either numerical coordinates or textual place name representations, be identified?

3. How can extraction be done automatically, so that large collections of several gigabytes can be processed?

4. How can an interface be designed, that allows the exploration of a data collection geographically, among the extracted data?

For this bachelor thesis, a software toolset named *GeoCrunch* has been developed. It addresses the challenges listed above and joins the various existing approaches related to the different media types occurring throughout archeology. The developed software should be considered as a pilot project. Implemented strategies and supported file formats are rather restricted, to meet the scope limitations of a bachelor thesis. For now, only English and German Language is supported. A demo package with some preprocessed sample data is also provided to show the software's capabilities (Appendix A, demo.zip).

Before outlining the software development, the further text will describe the discipline preliminaries, conditions and context in section 3. The archival context and the assumptions made about archaeological data collections are explained in section 3.1. Digitization and the spatial turn in the humanities have created — and were created by — technical developments, that enabled the spatial analysis of literature and textual documents in general. Section 3.2 reflects on this process and the new type of questions, that can now be asked using the extended methodology. The development of *GeoCrunch* and its potential

benefits for humanistic research is also discussed there. Section 3.3 reviews pre-existing approaches, which are evaluated to see whether the project's software development can be built upon them or not.

Section 4 provides a functional sketch of the analysis software, covering both the Extractor tool (4.1) and the Viewer of the *GeoCrunch* software package. The subsection on extraction stages and strategies (4.1.1) gives an overview of the variety of necessary solutions which are depicted more closely in the following subsections. A documentation of the implemented classes in section 4.1.7 ant the configuration section 4.1.8 finalize the description of the *GeoCrunch* Extractor tool.

The visualization discussion in section 4.2 explains the three different GUI elements of the *GeoCrunch* Viewer.

The evaluation of the software (section 5) does not focus on the classical quantitative ratings known from the information retrieval context (IR), precision and recall, as those are determined by the built in third-party components. Instead, an emphasis is put on how the enhanced query possibilities, as they are provided by the visualized result of analyzed sample data in the Viewer, can improve humanistic research or exploration of the analyzed data collections. A final conclusion (section 6) discusses the possible use cases, not only restricted to the envisioned application in an archaeological archive setting.

3 Context and Surrounding Conditions

The software development is targeted towards a wider surrounding of discipline preliminaries, concerning archeology and the neighboring humanities disciplines. While the archaeological archive IANUS gave the inspiration for this bachelor thesis as applied research, the software aims to provide a tool that is able to spatially enhance humanistic research questions in general. Thus besides the technical implications concerning data encoding standards and existing methodology, the scholarly backgrounds have to be explored, that have lead to the technical achievements within the humanities. Those are being used throughout the development of *GeoCrunch*, but it is important to understand how humanistic questions differ from the inherited techno-logical analysis, to interpret the results of technical tools correctly in a humanistic setting.

3.1 Archeology Archival Context

Archived digital objects are exposed to several risks like bit-rot or format obligation, which can occur after archival terms of just a few years. Usually, archival periods about 50 years are envisioned, which would be longer than most digital objects have ever been existing since information technologies have emerged.

Creating such a data center requires all kind of metadata to be created and utilized in

order to retrieve the relevant information from the archive. Among many other metadata attributes, the position of a given archival object in space and time has a high significance to archaeologists. While the chronological position can be described using taxonomies or ratio scales using a single dimension, spatial relations require at least two dimensions. Search interfaces for spatially indexed information systems usually employ a map view or provide other means to restrict the result set to a bounding box. This pattern is typically the most characteristic feature of a geographical information system (GIS).

In an archival context, the spatial indexing has to happen by the time of ingest. There are different requirements to a single object or file, compared to indexing a project collection as a whole. IANUS currently requests polygon data of the examined area by the data provider. However, this data is only collected for the entire project collection. In case the researched area stretches all over a large area, i.e. a continent, chances are, that just specific parts of that collection might be of interest for a later researcher's query on the archive's data center. So, the need for enhanced spatial metadata extraction becomes apparent.

The developed software tool is oriented towards the patterns found in the currently prototypical IANUS data center website, anticipating a possible later integration. Therefore, some framing structures have been borrowed to get the new software project started. This becomes most apparent when looking at the directory tree browser on the left, which has been improved and extended for the *GeoCrunch* Spatial Data Viewer (figures 1 and 3).

Figure 1: IANUS filebrowser

Another important, but at first glance invisible relationship between IANUS and *GeoCrunch* is the inherited data model, which has earlier been created for the IANUS filebrowser. *GeoCrunch* builds upon this data model. The Viewer has been developed by extending and improving the former IANUS filebrowser. All objects, either directories or files, are being described as collection objects, which hold the attributes found during analysis and also their descendants. This way, the data structure behind the directory tree and the map view is a tree itself.

A further implications from the archaeological context is the use of specific gazetteers for resolving (geocoding) found or suspected place names during the analysis process. The

DAI maintains such a gazetteer of ancient placenames. The *iDAI.gazetteer*[3] holds places that are of interest for archaeological research activities.

Some more orientation regarding the required functionality of the *GeoCrunch* prototype for this bachelor thesis has happened towards the data already available at the IANUS data portal. For now it is designed to find the spatial data and metadata from the media types of the existing IANUS collections. Based on these achievements, more testing can be applied to other data sets and additional extraction strategies for further media types can to be established at a later stage apart from this bachelor thesis. A quick review of the data collections that can already be found on the IANUS data center[4] gives an impression of the kind of data. During development, there were only two of them (since December 2016 there is a third collection, which will not be discussed).

The first collection, *Holozängeschichte der Tierwelt Europas* (Benecke, Driesch, and Heinrich 2016) contains no images, but many of PDF text files and tabular data such as CSV and Excel spreadsheets. While the tabular data can be considered to represent the collections spatial metadata, the PDF files lack any spatial metadata reference. Hence in both cases the spatial references are hidden in the file content bodies, these files have to be opened and parsed by the software for extraction of the geographical references. A remark must be made on the offered tabular data: the CSV files[5] just are a duplicate of the Excel spreadsheets beside them. Due to the long term preservation character of the IANUS archive, they try to avoid format obligation. CSV has a slightly better prognosis on this, though it has never been experienced on spreadsheets. A second consideration might be applicable for later dissemination, as this format can easily be imported into a multitude of applications. Conversion to CSV on ingest is not a standard procedure, neither in IANUS nor for long term preservation in general. Recommendations exist[6], though. To sidestep doubling the number of identical place references, only the CSV data become evaluated by the software.

IANUS' second collection *Felsbilder im Hohen Brandberg (Namibia)* (Lenssen-Erz and Pager 2016) in contrary contains only imagery of rockpaintings and just two descriptive PDF files. The images were created before the era of GPS enhanced photography and therefore have no geo tags holding their coordinates. Instead, the images' metadata has been extended in previous curational work to contain the place name.

Apart from the collections, an emphasis has to be put on the huge variety of media

[3] https://gazetteer.dainst.org

[4] http://www.ianus-fdz.de/datenportal

[5] The term *comma separated value list* (CSV) is not used in a strict sense, as the format is defined in RFC 4180 (https://www.ietf.org/rfc/rfc4180.txt). Instead, also all similar delimited value list formats are included.

[6] IANUS IT-recommendations: http://www.ianus-fdz.de/it-empfehlungen/tabellen, IANUS format migration table: http://www.ianus-fdz.de/it-empfehlungen/sites/default/files/DateiformateDatenmigration_IT-Empfehlungen_2016-07-07.pdf

types and diversity of neighboring disciplines that are intertwined with archeology. To put it simple, archaeological research makes use of almost all kind of data representation technique, that IT ever provided. As it will be clarified later, different extraction strategies have to be developed for different data types.

GIS vector data or three-dimensional CAD-structures are not unknown in archeology. They also are no longer foreign to humanistic disciplines of research, although sciences are their domain of origin. In the next section, we will see that times have changed, and technical efforts of two different academic areas, the sciences and the arts, will finally be combined.

3.2 Digitization and Spatial Turn in the Humanities

3.2.1 Historical Reflection

With regard to the humanities, which is the related top-level discipline of archeology, spatiality is a fairly new phenomenon. Paradoxically, it started with a geo-critcal bias in geography:

> *"The spatial turn began within the discipline of geography itself. By the early 1970s, geographer Edward Soja observes, many people in the field 'sought alternative paths to rigorous geographical analysis that were not reducible to pure geometries'. In this new critical geography, 'rather than being seen only as a physical backdrop, container, or stage to human life, space is more insightfully viewed as a complex social formation, part of a dynamic process.'" (Ayers 2010, p. 1)*

The term *spatial humanities* only emerged in literature in recent years, as in publications like D. Bodenhamer, Corrigan, and Harris (2015), D. J. Bodenhamer (2010) or Tally (2013). Spatial humanities do not keep at the traditional humanistic methodology. Instead new methods have been developed that are driven by technology and new approaches, such as information technology and spatial research. So GIS experienced increasing appreciation within the arts, while geo-criticism, as illustrated by Soja (2001) above, pushed this particular technology away from the humanistic geography subdisciplines. One can imagine, how the use of this new technology crept only slowly into the humanistic methodology, as of these adversing streams. Also unfamiliarity with IT among the humanist scholars might play a role in slowing down this process, as observed here:

> *"It is likely that most humanities scholars have had no dealings with e-science or associated grid technologies. In many ways the term e-science is sufficient to convince humanists that this area of methodological innovation is not of any interest or relevance to them." (Ell 2010, p. 145)*

As archeology belongs to the humanities, we will also have to talk about the relation between them — and their relation to space. Despite the relation between archeology and the humanities, archaeologists work on and across the borders between humanities and

the sciences. Strong bonds e.g. exist to history, cultural anthropology or history of art as members of the humanities, all of which are heavily involved with the study of textual sources. Towards sciences, a whole lot of geo-prefixed or geo-related disciplines need to be mentioned: geophysics, geodesy, cartography, climatology and many others, which play an important role in supporting the analysis of findings. With regard to the tight relation with the more space affine sciences, it is not a surprise to see spatially aware research and spatial technologies applied in the field of archeology first, before it emerged in other humanistic disciplines. Archaeologists belong to the early adopters of GIS technologies, as those fit their needs perfectly (cp. D. Bodenhamer 2010, p. 21).

Humanities have a long tradition on text based discourse, and research is centered on text. For a long time information technologies (IT) did not play a major role within humanistic research, as Gregory et al. characterize it:

> *"Once upon a time not so very long ago, it was all very simple - information technology was concerned with storing and analyzing databases of numbers. The discipline of statistics - which predated computing by centuries - provided suitable techniques for taking the large amount of numbers held in a database and summarizing them and the relationships between them, using a much small number of summary statistics and graphics. Thus the use of IT involved quantitative data and social science approaches, and, conversely, if you did not use quantitative sources or were suspicious of social science approaches you would not use it." (Gregory et al. 2015, p. 150)*

They further argue that in the following time during the ending twentieth century, IT increasingly came up with corpus linguistics for analysis of large corpuses using linguistic tools. Other linguistic related concepts, like "distant reading" (Moretti 2016), summarizes all techniques that extract the fundamental facts (e.g. what, when, where) from a text without requiring a human being to actually read the document. This period can be regarded as the digitization of the humanities. Linguists belong to the early adopters of IT, as their analyses primarily rely on statistics and formal logics, development of corresponding software tools and theories was then obvious. Information retrieval (IR) provides the theoretical background for systems that address the problem of retrieving just the relevant documents for a query. The actual search engines finally deploy a variety of tools originating in this development of linguistics for queries along text-based corpora.

The software *GeoCrunch* in turn relies heavily on techniques backed by linguistic processing techniques like named entity recognition (NER) to extract spatial references. Though *GeoCrunch* should not be considered to be a search engine that finds relevant documents, some IR concepts as relevance, precision and recall (see figure 2) can be applied for the extraction of spatial references, that *GeoCrunch* provides.

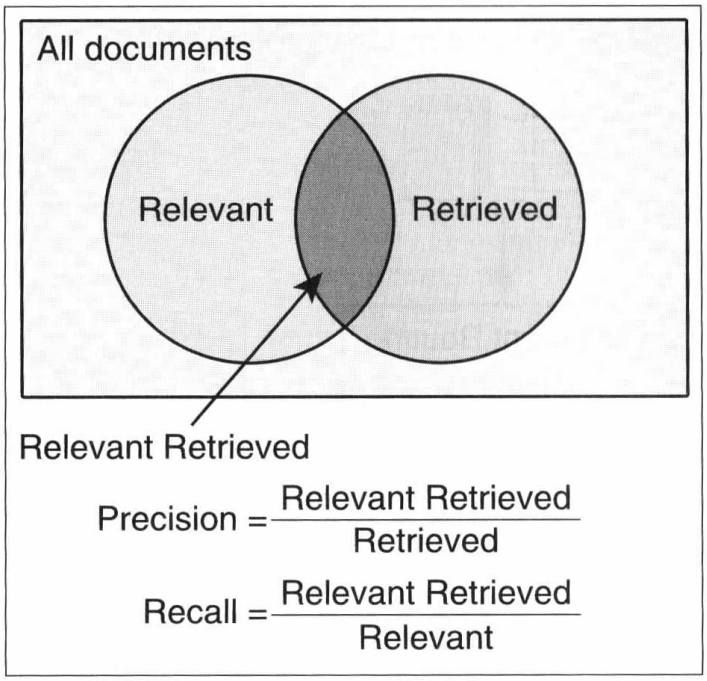

Figure 2: Precision & Recall in Information Retrieval (Grossman and Frieder 2004, p. 4)

As of IT's influence on text analysis, in parallel information technology also infiltrated the spatial sciences, bringing up geographical information systems (GIS) in the 1960s (cp. D. Bodenhamer 2010, p. 17). As Kemp states, this technology is rather connected to the sciences, than to the humanities:

> *"Geographic Information Science (GISci) is the science behind the technologies of Geographic Information Systems (GIS). As a science, GISci evolved in a context of precision, quantitative measurement, and notions of accuracy. As such, it might seem that its technology has little application in the humanities where imprecision, qualitative information and individual, sometimes conflicting, interpretations of 'facts' are the norm." (Kemp 2010, p. 31)*

The neighboring geo-sciences of archeology could easily provide information within this framework, to support archaeological research, though. Within the humanities in general, the new GIS query capabilities of course allow new questions to be asked.

3.2.2 Spatial IR for Humanities Research

Nevertheless within humanistic research, the new GIS technologies based on exact numbers have to be questioned themselves. The humanistic setting demands for fuzzy queries, which is an applicable approach for visual, interactive GIS, that do not rely on programmatical queries, but allow the user to explore the data visually. What is being attempted to be achieved with *GeoCrunch*, is to combine the methods of digitization of text-based research and geometry-based spatialization into a spatial information retrieval approach, that suits the needs of humanities research. Due to archeology's huge media type di-

versity, *GeoCrunch's* extraction strategies are diverse and provide results with diverging accuracy. Thus, some context information out of the file content body has to be displayed to the user, to allow assessment of the displayed results.

The use of a classical text-based IR system is considerably different to operating a GIS. Both expect being fed with a query by the user, but typical graphical user interface (GUI) implementations and also query types differ, as the following examples illustrate:

IR query:
Which are the relevant documents for the query?

GIS queries:
What is at/near a given coordinate?
What can be found in a given box?
What is the coordinate of an address? (geocoding)

One quickly comes to the conclusion, that the single IR query type somehow is similar or related to the first two GIS types.

Spatial IR query:
What is relevant for a given place or region?

Creating such a spatial IR system for the humanities has a clear impact on the design of the GUI. Typical IR GUI implementations just provide a text box (or some more of them), where one is expected to submit a textual query. A GIS can be queried in a similar way, by filling text boxes with numerical coordinates or attribute values. But most of us have already been using popular GIS like GoogleMaps in a different, interactive manner: by zooming, panning and scrolling the map to retrieve the information of interest. This operational mode is much more suitable for the problems and questions brought up in the humanities, if they touch the spatial dimension. The considerations of Gregory et al. have also been taken into account while designing the Viewer GUI of *GeoCrunch*.

> *"Clicking other place-names in the text would, in turn, highlight these on the map. This approach allows the reader to ask the question What has been said about this location? and then read all of the responses in the corpus. (sic!) It is thus well suited to close reading but makes use of the hyper-textuality within digital texts such that rather than reading in a linear manner from beginning to middle to end, the reader can switch from place to place within and between texts. In this case, geographic location provides the hyper-textual structure through which the reader can approach the analysis (or reading) of text. [...] While map-based querying provides an approach that fits well with the humanities traditions, spatial analysis comes*

> *firmly from the social and Earth sciences. While this means that the approaches offered must be used sensitively, it does not mean that they should be rejected, as they provide a highly effective way of summarizing large and complex geographical patterns." (Gregory et al. 2015, p. 163)*

Humanist researchers explore cultural space, which often is deeply interwoven with the physical space. The map being displayed in interactive, visual GIS can provide rich information about the context of the problem being explored by the researcher. The context of physical space has reflections on the applied concepts of nearness. The human operator in front of an interactive map has the query in mind and can extend or change it quickly according to her deep knowledge on the subject and the provided information. For example, a mountain range might have a different impact on the dispersion of cultural achievements than a river, thus a settlement of boat people far away along a river course may be considered much closer than the village just on the other side of a mountain ridge, even if bee line distances as the exact concept of nearness might tell a different story. If the people researched had no affinity to water, on the other hand, then all features in physical space would have to be rated in a different manner. This cultural context information can hardly be processed by any automated IR process, either text based or spatial — and it will hardly collect all relevant information.

The *GeoCrunch* Viewer GUI has been designed with regard to the challenges outlined in this chapter, featuring a rich and interactive interface, supporting the fuzzy humanistic research modus to explore the data collections analyzed by *GeoCrunch*. A "humanities GIS" is envisioned by Ell, which is not centered on map display of results, but on research infrastructure:

> *"Rather, GIS will be used in a conceptually far simpler way as a vital piece of e-research infrastructure. All humanities research sources contain key elements that can be used by a GIS. First, information has a spatial location whether expressed precisely or generally. Second, all humanities data contains a temporal marker, even though its granularity may vary from minutes to a decade, century, or more. Thus, in most instances in the humanities, research information is placed within a chronological framework. Third, all humanities data can be classified by subject. In fact it is only this last facet of humanities research data that has received much attention to date thanks to the work of library and information management professionals. In the analog world libraries organize monographs and manuscripts by subject area." (Ell 2010, p. 145)*

With regard to the outlined humanities GIS, *GeoCrunch* can contribute to its creation, as far it is able to extract the spatial references. The still prototypical archaeological IANUS archive project facilitates metadata across all other attributes mentioned by Ell above. Both in in combination would remarkably well match the envisioned humanities GIS, as IANUS can be considered to be an infrastructure project.

3.3 Existing Approaches on Spatial Information Retrieval

As already mentioned, the challenge in archaeological archive settings is to query different types of media. Only some of them are text-based, others contain genuine geographic information on various encoding levels. Regarding the problems of spatial IR for the humanities, investigation on existing software during the bachelor thesis creation showed that next to nothing has been developed before to tackle this diverse problem at once. On the one hand, there are many approaches towards information extraction from text. The choice of implemented libraries among corpus linguistics and text analysis will be discussed in section 4.1.2. On the other hand, implementations are available which focus on non-textual media types. None of them is simultaneously designed to provide spatial IR capabilities, the previously required information extraction and to match the requirements of humanistic research.

A project targeting media types that contain genuine geometry data, is the GIS software gvSIG[7]. This open source GIS project was formerly developed by the regional government of Valencia, Spain. Now the development is mainly driven by a programmer who is also seated at the DAI. A dedicated archaeological version of gvSIG has also been released.[8] The software is capable of importing vector data, imagery (raster data) and databases containing well-known geographic information, so far not unusual for a GIS. Unfortunately the available documentation of gvSIG is very poor. In addition, it has only partly been translated from Spanish into English, yet. Díaz et al. (2007) have built an extension on top of this software for semi-automatic extraction and export of spatial metadata. Presumably, the information has to be well defined for a successful primary import into gvSIG. In case crucial information is missing, e.g. the CRS or the column names to import coordinates from, further user input is required, as indicated in a tutorial for importing[9]. Only if the provided information complies with common metadata standards, the process might be easier and could be automated. Hence the approach by Díaz et al. remains operating semi-automatically. Since the publication of their extension, development of gvSIG has been driven much further, so that the extension would not work any more, according to information from the developer at the DAI. So further investigation on gvSIG has been canceled for creation of the bachelor thesis software.

Finally, gvSIG only imports the data into a GIS. As the extension developed by Díaz et al. cannot be deployed any more, it is not clear, if the retrieved spatial information can be exported into any other format for further processing in an archive system. For integration into an automated software tool, a commandline interface (CLI) would also be required. gvSIG has such a CLI, but the documentation is very poor again. No hint

[7] http://www.gvsig.com/en/products/gvsig-desktop

[8] http://oadigital.net/software/gvsigoade

[9] http://downloads.gvsig.org/download/documents/books/gvSIG-Kurzanleitung_fur_die_Erstellung_von_Fun.pdf#page=38

could be found, that it provides the capability to export actual data into something like a CSV. As Apache Tika, the framework being used to access the file contents and metadata (discussed in section 4.1.2), provides file metadata export of a much wider range of media types, gvSIG and its extensions have not been investigated any further. Instead, Apache Tika has been favored, even though Tika does not provide export of genuine spatial data like vector data. For the development of the *GeoCrunch* pilot within the bachelor thesis project, gvSIG has been abandoned for the above reasons.

Software libraries such as GDAL[10] could be integrated much easier to handle the task of spatial data extraction from vector data formats, as they are designed to be imported into software projects. However, *GeoCrunch* is currently limited to the display of point coordinates, thus it would be required to extend *GeoCrunch's* capabilities towards more complex geometries, e.g. polygons. Otherwise it would be neccessary to make use of GDAL's abstraction abilities, e.g. to calculate a polygon's centroid to supply a point coordinate representing the polygon.

3.4 Data Encoding Standards

For the retrieval of spatial references, metadata and data encoding standards are of importance. Even though *GeoCrunch* is constructed to take raw data as input, which usually will not be backed by edited metadata, standards exist that deal with the consistent encoding of data. Whichever level a standard actually is designed for, they have to be considered for determination of relevant metadata fields or data columns during extraction. This is of increasing importance, as a standard's field names of interest are possibly spelling way apart from the most obvious names for places or geographical references.

Regarding the literal place names and references, a quite simple *bag of words* approach has been chosen for *GeoCrunch*, which includes the most obvious field names with place references in English and German language, like *place, spot, city, country, ort, fundort*, but also the relevant labels from a variety of encoding standards discussed in this chapter. This approach is suitable to extract literal place references as in listing 1.


```
"tika_metadata" : {
    "Artist" : "Photo",
    "subject" : "Rock art - paintings",
    "dc:title" : "Felsbild (Menschen)",
    "Headline" : "Felsbild (Menschen)",
    "resourceName" : "AAArC-HP0000095-1-PB-77-A10-F-9_6.jpg",
    "City" : "Amis",
    "title" : "Felsbild (Menschen)",
    "Last-Save-Date" : "2014-08-27T02:00:00",
    "File Size" : "1638835 bytes",
    "Credit" : "Pager, Harald",
```


[10] http://www.gdal.org/

```
    "Country/Primary Location Name" : "Namibia",
    "File Name" : "AAArC-HP0000095-1-PB-77-A10-F-9_6.jpg",
    "Content-Type" : "image/jpeg",
    "X-Parsed-By" : "org.apache.tika.parser.DefaultParser",
    "Province/State" : "Brandberg",
    [...]
}
```

Listing 1: Metadata sample with place indications

```
"tika_metadata" : {
  "GPS Measure Mode" : "3-dimensional measurement",
  "GPS Map Datum" : "WGS-84",
  "Image Description" : "County Road 32E",
  "tiff:Make" : "RICOH",
  "GPS Latitude Ref" : "N",
  "GPS Altitude Ref" : "Sea level",
  "geo:long" : "-98.412892",
  "GPS Latitude" : "45° 31' 53,11\"",
  "File Size" : "114275 bytes",
  "GPS Satellites" : "15,18,27,21,14,06,22,26,,,,",
  "GPS Longitude" : "-98° 24' 46,41\"",
  "resourceName" : "geotagging flooding.JPG",
  "GPS Longitude Ref" : "W",
  "geo:lat" : "45.531419",
  "GPS Altitude" : "390 metres",
  "File Name" : "geotagging flooding.JPG",
  "Content-Type" : "image/jpeg",
  "X-Parsed-By" : "org.apache.tika.parser.DefaultParser",
  [...]
}
```

Listing 2: Geocoded metadata sample

When it comes to the detection of coordinate pairs, corresponding field names have to be selected. Therefore, a general matching approach would fail, if one tried to select the fields containing *lon* and *lat* using a regular expression. Metadata retrieved from a geotagged test image (see listing 2) reveals the problem: in a random order, the fields *GPS Longitude*, *geo:lat*, *geo:lon* and *GPS Latitude* appear. Whereas the *GPS_* field names meet the EXIF standard, the other *geo:*-prefixed names belong to the Resource Description Framework (RDF). These resemble two corresponding field pairs. In addition, they make use of different number formats. While one of them (RDF) serves proper floating point decimal degree values. The other, Exchangeable Image File format (EXIF), makes use of sexagesimal fractions with degree, arcminute and arcsecond notation. To avoid problems during retrieval of corresponding value pairs, exact matching is a requirement for field names. For this reason, a closer look at the frequently implemented standards is

necessary.

EXIF [11]:

Exchangeable Image File format (EXIF). The fields *GPSLatitudeRef* and *GPSLongitudeRef* hold the cardinal indicating the coordinate orientation. Common Cartesian systems would be oriented towards North for the Y-axis, East for the X-axis. For example, 90°S latitude would point to the South-Pole. The alternative according notation is -90°N.

GPSLatitudeRef
GPSLatitude
GPSLongitudeRef
GPSLongitude
GPSMapDatum

RDF - *Spatial Thing* [12]:

Resource Description Framework (RDF). The use of the RDF vocabulary *geo:* namespace implies the WGS84 coordinate reference system.

geo:lon
geo:lat

IPTC-IIM [13]:

Information Interchange Model (IIM) by Comité International des Télécommunications de Presse (IPTC).

Content Location Name
City
Sub-location
Province/State
Country/Primary Location Name

XMP [14]:

Extensible Metadata Platform (XMP).

photoshop:City
photoshop:Country
photoshop:State

[11] Camera & Imaging Products Association (CIPA) and Japan Electronics and Information Technology Industries Association (JEITA) 2010
[12] W3C Semantic Web Interest Group 2009
[13] Comité International des Télécommunications de Presse (IPTC) 2014
[14] Adobe Systems Incorporated 2012

ADeX [15]:
Archäologischer Daten-Export (ADeX). In acheology, the ADeX is a standard for data export and exchange, which is not primarily intended for metadata encoding. Its module 3.1.2. about georeferences provides the relevant column names.

X_KOORD, Y_KOORD
KOORD_REFSYS

The CRS can be found throughout the mentioned (meta-)data standards, either implicit due to a standard's specification, or by explicit declaration. Sometimes the EXIF *GPSMapDatum* field is missing. Among the example data processed for this thesis project, WGS84 was the only occurring system. Therefore, WGS84 will also be assumed for all cases, where information about the CRS could not be found or is not present. For all other systems, a reprojection of the identified coordinates would be required, as the software internally also uses WGS84. Due to the insignificance of other reference systems among the analyzed example data, CRS detection and reprojection to WGS84 has not been implemented within the limits of the bachelor thesis.

4 Software Development

The project is named *GeoCrunch*, as it has the character of a machine or mill, providing a condensed spatial view of the spatially analyzed data to the user. A commandline analysis tool, the Extractor, processes the provided input data, and returns the extracted geographical references as a metadata object. Usability is of great importance — because presumably most people do not appreciate condensed, abstract object representation, the outcome is hidden into the 'crunch' part, the Viewer. This is a presentation of the collection and its spatial metadata in a browser-based, interactive application. The collection Viewer is intended for easy integration into web based data centers. It presents a file-tree view next to a map to display the detected geographical references and this way enables the spatial exploration of the examined data collection.

A distinction has to be drawn between the developed tools and the class of GIS software. Even though the interactive panning and zooming features provided by the *GeoCrunch* Viewer appear familiar to GIS software, its architecture is fundamentally different. GIS is usually built upon a database, which enables various queries on the included features. The *GeoCrunch* Viewer instead makes use of a nested Javascript object notation (JSON) object, that technically cannot easily be queried in the same manner. Practically, panning and zooming can be regarded as querying the provided data to retrieve a result

[15] AG Modellierung der Kommission „Archäologie und Informationssysteme“ im Verband der Landesarchäologen der Bundesrepublik Deutschland 2011

set concerning the area of interest. But the technical process behind the curtain differs substantially.

As the relations towards IANUS have been discussed in chapter 3.1, the presentation part is designed as a web based application intended for integration in the IANUS data portal. Nevertheless integration into other websites is quite easy as well. Being oriented towards the IANUS website and use of its internal client side data model lead to another important design implication. For now, *GeoCrunch* remains a client application only and does not include a webservice or dedicated server side software. The inherited JSON data model is intended for transferring information to the web browser application for display. No queries against the internal fields of this model are targeted, neither for the IANUS data portal prototype, nor the *GeoCrunch* software. Thus, the JSON output of the *GeoCrunch* Extractor is not a database, but an enhanced representation of the original collection structure. Running queries against particular attributes would require storing the data into any database implementation, either object oriented or relational. Spatial indices, which are required to support spatial queries are provided among both types.

Being able to perform spatial selections as in a GIS has not been targeted during creation of this bachelor thesis. Hence, this may be regarded as the main characteristic, which differentiates *GeoCrunch* from a GIS. Not being able to perform programmatical selections distinguishes it from an IR sytem at all. *GeoCrunch* aims at spatial information extraction and facilitating data exploration instead.

To achieve the envisioned results, two different paths have to be followed simultaneously: firstly, an analysis tool has to be developed to extract the spatial data and metadata from the input data. This will be discussed in section 4.1. Secondly, the results have to be presented in a way, that persons working on the analyzed data can easily assess them, based on the spatial analysis results. The visualization achieved by the *GeoCrunch* Viewer is outlined in section 4.2.

4.1 Extraction

4.1.1 Extraction Stages and Strategies

Spatial references can be found on three levels, which have to be considered during the analysis process.

1. **File and Folder Names**

   Names of directories and files often include literal place references.

2. **File Metadata**

   The retrieved file metadata may include literal place name references and/or plain coordinate references (geotags) beside other (i.e. technical) information.

3. File Content

While analysis on this level is focused on text based formats (e.g. no image, audio or video content), two different cases, unstructured natural language text and semi structured data (tabular data), have to be handled separately.

Table 1 gives an overview of what has been achieved during creation of this bachelor thesis. Further development can continue implementation of additional detectors based on the created framework.

Level	**Content Type**	**Strategy**	**Implemented**
file & folder names		name splitting	✓
		geocoding	✓
file metadata		coordinate field detection	✓
		CRS detection	
		coordinate conversion	
		place field detection	✓
		hierarchic place evaluation	✓
		place geocoding	✓
content	text	NER based places detection	✓
		places geocoding	✓
		coordinate detection	
		coordinate conversion	
	CSV	coordinate column detection	✓
		coordinate conversion	
		place column detection	
		places geocoding	
	tabular	tabular data conversion to XML	
		XML based column parsing	

Table 1: Georeference extraction strategies on the levels *file name*, *metadata* and *content*

A remark has to be made about the implemented coordinate detection on the levels metadata and content (levels 2 and 3 on the list above): as all *GeoCrunch* components internally work with polar WGS84 coordinates, coordinate conversion or re-projection is not required, as long as only polar WGS84 coordinates are detected. For simplification within the scope of a bachelor thesis, a coordinate parser class has been developed that is intended to check solely, whether a given string is a valid polar coordinate. So far, this parser can process the separated ordinates of a coordinate pair and WGS84 has to be assumed as the CRS, as no other information is available. Even though UTM and Gauss-Krüger coordinates can be distinguished by their string format and numerical range, detection or validation logic for these has not been implemented within the limited scope of this

bachelor thesis. Thus the coordinate parser either returns the validated polar coordinate values, cardinal and direction (if applicable) or an empty object on error. Parsing of full coordinate pairs, e.g. from free text, has been skipped due to their numerous possible formats. Number formats differ between locales and separators between longitude and latitude can be confused with decimal delimiters. Finally, no uniform convention exists which axis is mentioned first for polar coordinates. If no cardinals are given, the order of axes can not even be guessed, as long as no value exceeds 90°. For these reasons, detection of other than polar WGS84 coordinates from other than well defined metadata fields or tables has been put aside.

On the file content level (level 3), characterizing natural language text as unstructured implies a simplified technical perspective, because it is nevertheless well structured with respect to the grammatical rules of real languages, where linguistic tools such as NER can be applied. Basically, this labeling is done to distinguish prosaic, literary forms of data from those organized in tables, rows and columns. Tabular data from file contents on the other hand, can be seen as unstructured from a linguistic point of view. As a single data cell contents hardly ever holds a complete sentence, grammatical rules do not apply. For this reason, the only grammar applicable here, is the relation between column headers, field names and the corresponding data cells. This is, technically spoken, a rather simple grammar. From a technical perspective, that does not know about natural language structures, tabular data can be seen as well structured as opposed to *unstructured* text. However, the proper column names have to be known or should at least be recognizable by the analyzing software. File contents often are not designed along a well-known metadata standard, thus we have to consider tabular data as *semi structured*. Detection of the relevant column names is quite analogue to the approach used to extract geographical references from the file metadata (see below).

It is possible to extract plain text content bodies from a variety of text-based file formats, including tabular data like Excel sheets. If the file can, according to its mime type, be suspected to hold natural language (e.g. application/pdf or any word processing format), then a linguistic toolset can be applied to perform NER. If the mime type indicates an origin from any spreadsheet analysis program or a proper comma separated value file (CSV), then Tika's default output in fact is a CSV list. The software further tries to detect the CSV format (delimiter, quotes, column headers).

However, conversion of complex spreadsheets into regular tabular data is problematic, information may get lost or corrupted. For this reason, solely genuine CSV files are being processed by *GeoCrunch*. This is due to the fact, that tabular data becomes converted to CSV for reasons of long term accessibility by IANUS and are offered next to the original format. Evaluation of other formats than CSV would lead to redundant results. Alternatively, the content analysis framework Tika (discussed in section 4.1.2)

has an XML parser, which parses tabular documents into an XHTML table. So far it has not been evaluated if this would offer a more reliable way to read the table headers and their according values while not dealing with the correct detection of CSV delimiters or quotation format. However, the use of this XML parser would enable analysis on a greater range of tabular document types. Using the plain text parser does provide access to spreadsheet types, but the results can hardly be processed. If the resulting XHTML structure of tabular data files, including CSV, is uniform enough, the XML parser is an option.

A related problem emerges, when file names are being examined (level 1). They rarely form a correct sentence, similar to table column names. Nevertheless, often meaningful semantic information is provided by the deeply structured collections and their file and folder names. To extract geographical references, a two step approach has been implemented. Basically, the names are being split by their suspected word boundaries and subsequently geocoding the fragments is attempted. Word boundaries for the purpose of place name identification can be expected at any character, which typically is not part of a proper name, like _.,\/#$@|~+”’;?!()<>=&[]{}, whitespace and all digits from zero up to nine. In addition, the hyphen or minus (-) and capitalization provides further hints, like *camelCased* or *TITLECased* notation. Those are sometimes used to put a weak but meaningful link on the name fragments, like *SüdIberia* or *Köln-Mülheim*. Applying a very strict splitter rule would lead to erroneous results, because Köln-Mülheim is a distinct sub-entity of greater Cologne, while *Mülheim* alone would be identified as *Mülheim an der Ruhr* by a geocoder. After some testing, it appeared the best solution, not to consider capitalization and hyphens for file name splitting (a more strict regular expression is used for column name detection on file content data, though). The name fragments then become filtered for a configurable minimun length. This is to avoid getting too many irrelevant results, because file names often happen to include abbreviations up to three letters, which in most cases are not a reference to a geographic place. Wikipedia provides an astonishing overview of short place names[16], that would be confused with those abbreviations if not ignored by the minimum length rule otherwise. It is recommended to adjust this setting, if one is interested in place names containing less than three characters. The minimum name length can be altered in the configuration file (section 4.1.8).

For the extraction of relevant information from the file metadata (level 2), an outline of the considered metadata standards and obvious fieldnames being used to retrieve the information of interest has already been given in section 3.4. While the same approach is not only being used for metadata extraction, but also for the extraction of semistructured

[16] https://en.wikipedia.org/wiki/List_of_short_place_names

tabular data from file contents (level 3), things have to be discussed a little more in detail. The analysis of a tabular document (or CSV) can be broken down into performing the almost same steps on each row as they are being done on the metadata. The file metadata, usually regarded as a key-value list, can also be seen as a table with a header row and just one data row. Where only one row of data has to be extracted, it can be read into a key-value structure. The relevant fields then are detected using the bag of words approach already discussed above, which also contains the keys of all respected standards. Extraction of the relevant values will then occur immediately.

If more than one row is available, as it is the case in tabular data from file content bodies (level 3), then it makes sense to identify the relevant field names only once and apply that field set on each loop cycle over the table data body. Thus, there is a slight difference, compared to the metadata evaluation process. To collect information from data columns that are not defined by any standard, but by more or less obvious names, the same name splitter function is used as for the file name analysis, but with a more strict regular expression to detect the word boundaries. In the collection of coordinate or place reference name parts, no strings containing hyphens or changing capitalization are present. Weak ties do not play the same role for column names as for place names.

Summarizing this section, the various approaches on how the relevant spatial information can be accessed and extracted from the three levels *file names, metadata* and *file contents* have been introduced. The implementation of these approaches finally will be outlined in section 4.1.7. As the file names on level 1 can be accessed quite easily by recursively iterating over a directory tree, the file metadata and contents on level 2 and 3 need some more advanced methods, which will be discussed in the next section.

4.1.2 Data Access

To extract spatial references from all kind of files and their metadata, it makes sense to make use of frameworks which already encapsulate the manifold tasks that have to be performed. For this bachelor thesis, the content analysis framework Apache Tika (The Apache Software Foundation n.d.[b]) has been chosen. It is capable of looking into a great variety of file types, parse and extract their contents into various formats, retrieve their metadata and can do some basic analytics, like language or mimetype detection. Metadata extraction includes IPTC-IIM, EXIF and XMP data, as depicted in section 3.4. While EXIF is primarily intended to provide additional information on image data, XMP was introduced by Adobe (Adobe Systems Incorporated 2012) to store metadata on a greater variety of file types. For these characteristics, it quickly became clear, that an evaluation of Apache Tika with regard to its extraction and analysis capabilities will be part of this bachelor thesis. Tika therefore is a central component in the development of the spatial

(meta-)data extraction tool and determines also the probably most fundamental design decision to do the entire project in Java.

Tika contains a number of special parsers dedicated to different file types. It is possible to select a specific parser during the implementation of the Tika software, but in most cases one would wisely leave that decision to Tika, which parser to select. Tika makes use of mime type and content inspection on varying levels, the parser becomes selected on this basis. Some examples: for an image file only the metadata parser would be enabled, as there is no other textually readable content. Most office documents actually resemble a container of zipped XML files and require at least an XML-parser to retrieve the plain text file content body.

Language detection plays role to determine the appropriate language model for named entity recognition and can be demanded from Tika. Named entity recognition is a text analysis technique, which relies on proper detection of the document language. Text bodies consisting of natural language (e.g. no tabular documents) can afterwards be parsed to identify entities that have a proper name, e.g. individual persons or place names. To perform this task on applicable file types, Tika extracts the file contents line by line, until it has sufficient content for a comparison with its built-in language corpora. Currently, English and German language models are implemented.

Apart from language detection and access of file metadata and contents, additional techniques have to be implemented to finally achieve spatial reference extraction on content level.

4.1.3 Metadata Reader

Metadata delivered from Tika are key-value lists, technically a Java map of string pairs. Now the challenge lies in detecting the field names of interest. If corresponding coordinate pairs are in question, then related field names have to be identified. Place indications may be dispersed across several fields, each ranging on a different hierarchical level, e.g. country: Namibia, city: Amis, region: Brandberg (see listing 1). So a staged list of stop words has been defined, to collect all information indicating a place (table 2), where the level keys on the left correlate with a variable of class *Place* (see section 4.1.7), which stores the found values, preserving the hierarchy. This way, which ever information is provided in the metadata, it can be handed over to the geocoders to narrow down the results. Geocoders partially support some variant of hierarchic queries (multi-field or single line query string). To collect associated coordinate pairs or more complex geographical references containing a larger keyset, a similar approach has been chosen. Instead of the hierarchic levels, an array holds arrays of corresponding keys, that are derived from metadata standards, quasi standards or just obvious (table 3). While iterating over the lists of corresponding keys, the chosen method checks whether all specified fields are present in

Level	Stop Words
continent	continent, kontinent
country	country, land, staat
state	state, province, bundesland
region	region
county	county, kreis
city	city, village, stadt
place	place, spot, findspot, ort, standort, standpunkt, fundort

Table 2: Hierarchic place stop words

Corresponding Keys	Standard
lon, lat	
long, lat	
longitude, latitude	
länge, breite	
geo:long, geo:lat	RDF
gps longitude, gps latitude, gps longitude ref, gps latitude ref	EXIF
x_koord, y_koord	ADEX
x_wert, y_wert	
e_wert, n_wert	
east, north	

Table 3: Corresponding coordinate keys

the dataset. Exact string comparisons are applied, no substring matches, because e.g. the string 'lat' from the first list could be found in 'geo:lat' and 'lon' in 'gps longitude'. This is problematic, as 'gps longitude' should not be returned in conjunction with 'geo:lat'. Both arise from distinct metadata systems and make use of different number formats, as illustrated in listing 2. In some metadata schemas there are more than two keys required to have a valid geographical reference. This is the case with EXIF, where cardinal indicators are provided and have to be evaluated as well.

Quite similar to reading metadata, also tabular data can be parsed using the above approach. The table header line has to be combined with the selected table data row to resemble a key-value list. Vice versa, a key-value list can be seen as a small table with a header row and a single data row. Nevertheless, parsing tabular data works a little differently. Detection of the right columns should happen only once per table for performance reasons. As soon as the keyset has been determined, their values can be read row by row in a loop, performing the field detection over and over again.

4.1.4 CSV Reader

The plain text file content body of CSV files, as extracted from Tika, has to be interpreted using a specialized approach, as CSV format specifications diverge. As no direct open-source solution for the proper detection and interpretation of CSV formats could be found, a custom implementation has been created. The implementation details are discussed in section 4.1.7, where the class *FieldDataReader* is outlined as well. The *CsvReader* class engages with the latter to retrieve spatial references according to the keyset, which previously has to be detected there.

4.1.5 Named Entity Recognition

As the content analysis tool Tika does not encompass named entity recognition (NER), other specialised libraries have to be included to get this task done. For each language, analyses shall be ran in, a language model is required. These language models or classifiers contain the indicators and rules the library uses to identify a named entity. Most models are capable to identify and classify several types of entities, like person, location, organisation or geo-political entities. Two widely used implementations are Apache OpenNLP (The Apache Software Foundation n.d.[a]) and Stanford NER (Stanford NLP Group n.d.).

OpenNLP have no German classifiers available, which is critical because IANUS is an archive located in Germany and dedicated to German archaeology. So further attention was paid to Stanford NER, because German classifiers are available. The utilized models for English and German provide different classification tag sets, however a location tag denoting the found place names is included in both of them.

For the English language, the 3-class model (location, person, organization) created by Finkel, Grenanger, and Manning (2005) shipped with the Stanford NER package has been used. The German models from Faruqui and Padó (2010) can be downloaded from the Stanford NER homepage[17]. They offer two different models, *deWAC* (web as corpus) and *HGC* (huge german corpus). Both originate from the internet, as the *HGC* classifier has been trained on a collection of news-wire texts. Therefore, the *HGC* classifier is recommended for the classification of this exact genre, while the *deWAC* corpus contains texts of all kinds and should be used for everything else, as stated in the README file[18].

There exists also an implementation of Stanford NER for Apache Tika, which has been evaluated. This works fine, however it only returns the identified entity names and their types. To display the context of the entity mentions, their position in a plain text content body is also required. Thus, this approach has to be abandoned, in favor of a native implementation of Stanford NER into the extraction analysis tool.

[17] http://nlp.stanford.edu/software/CRF-NER.shtml#German
[18] https://www.nlpado.de/~sebastian/software/ner/README.txt

4.1.6 Geocoding

The term *geocoding* stands for looking up place names or addresses in a database in order to retrieve the corresponding coordinates. Usually this involves huge databases, that can only be maintained commercially, or by scientific interest groups and the public domain. These databases can be queried over the web using a webservice API.

As already stated in section 3.1, the selection of geocoding webservices has been driven by the archaeological context. For that reason, the iDAI.gazetteer[19] has been implemented, which is being maintained by the DAI itself. Though the documentation is protected from public access, authentication to the API is optional. However, anonymous requests will result in fuzzy coordinates on some places, as they are subject to sensitive archaeological research and thus endangered to be destroyed carelessly if revealed freely. No packaged Java implementation is available, and a geocoding webservice interface had to be created.

Other important resources, that have been evaluated for *GeoCrunch*, are Geonames[20] and the OpenStreetMap project (OSM) with its search interface Nominatim[21]. While Geonames exposes a public API with generous hourly and monthly limits to issue queries programmatically, Nominatim has some restrictions affecting bulk geocoding. Due to limited server capacity, they do not encourage sending more than one request per second, which would very likely be the case for the *GeoCrunch* application. Thus they ask to make use of other providers, which run a copy of the Nominatim webservice on servers with greater capabilities. One of these providers mentioned on the Nominatim usage policy page[22] is Mapquest. Mapquest is a commercial service and maintains two databases. One called Mapquest-Free, which contains proprietary data, is available for free within fair query limits. The other, Mapquest-Open[23], replicates the Nominatim service. Query limitations are applied here as well.

Open-source Java implementations are available for Geonames[24] and Mapquest-Open[25], so that integration into *GeoCrunch* is quite easy and has been done also to Mapquest-Open for this reason. For the iDAI.gazetteer, custom implementations had to be created. This could be achieved by modifying the Mapquest-Open package.

Evaluation of the results is a matter of scale. Having a street address's coordinate returned, would only make sense if the query provided a street address. Geonames is a well established gazetteer for geographic place names worldwide, that holds names of large scale geospatial structures, either administrative or geographical, like continents,

[19] https://gazetteer.dainst.org
[20] http://geonames.org
[21] https://nominatim.openstreetmap.org/
[22] http://wiki.openstreetmap.org/wiki/Nominatim_usage_policy
[23] https://developer.mapquest.com/documentation/open/geocoding-api/
[24] http://www.geonames.org/source-code/
[25] https://github.com/gjordi/mapquest-open-geocoding

countries, cities or certain spots like 'Stonehenge'. The iDAI.gazetteer is built in a similar manner. They are not centered on street level addresses, as opposed to the Nominatim services (Mapquest-Open), which can be a problem: the latter sometimes return results, that do not have any recognizable similarity with the query string at issue. Especially the Nominatim service showed this behavior and returned even company addresses, if the query string was contained in the company name. It never returns an empty result. In these cases it would be better to have no result instead of an irrelevant one.

Within *GeoCrunch* It has been attempted to to achieve avoiding irrelevant results by querying the webservices in a certain order. The services that are most specific with regard to the subject are queried first. If a result is returned, then further querying is halted. Otherwise the next service is requested until the end of the geocoder list. For this reason, services that are likely to return an irrelevant result, are listed last. In terms of IR, this would increase the precision measure, with relatively low impact on recall (see figure 2). The Mapquest-Open geocoding service finally has been removed from the precedence list, as it more often returns an irrelevant result than a relevant one, that the other services would not have found before. The internal matching mechanisms are too fuzzy, even though the actual implementation has already been tweaked to avoid, for instance, company addresses as results and the minimum query string length has been increased up to six characters. Another reason to deactivate the Mapquest-Open service was, that the servers seemed to be overloaded frequently, which causes the extraction script to terminate with an error. Nevertheless, the implementation is still available in the source code and can be activated in the Extractor configuration on demand.

As all the inquired geocoding webservices have different result schemas, therefore a result ranking is impossible. Most services do not even return information about the match accuracy, nor can their APIs be tweaked to restrict results up to a certain level of spatial hierarchy (e.g. no street addresses, or company names) or to specific feature types (e.g. mountains, settlements etc.).

All geocoding webservices are connected to *GeoCrunch* using the abstract *Geocoder* class. This way it is quite easy to add further geocoders as required. Only an adapter class has to be written, that extends the *Geocoder* class and and implements optional third party libraries. In case no third party libraries are available, the generic webservice package inside the *GeoCrunch* source code may be utilized to do all the HTTP communication. This makes sure the new geocoding service meets the requirements of *GeoCrunch*. The order of the geocoding webservices can be defined in the configuration file (see section 4.1.8) by lining up their full qualified classnames. The configuration file also holds the account information of the utilized geocoding services. Most of them require an API key in order to relate their query limits to single accounts.

4.1.7 Class Descriptions

This section gives an overview of the *GeoCrunch* package, classes and subpackages and how they are involved in the spatial data extraction process.

There are two entrance points to get the process started, either using the command line interface (CLI) by invoking the *Commandline* class or by importing the *GeoCrunch Core* class directly into other Java applications. The class are outlined in the following text.

Commandline class

This class only has the single static method *main*, so it can be called without constructing the class before. The method is able to take an array of arguments, all of which are optional. The provided arguments are intended to control the behavior of the tool directly, where otherwise settings from the configuration file would be in effect as a default. The method *main* handles the provided arguments, invokes the *GeoCrunch Core* class, starts the process and prints some lines to the console to inform the user when the extraction has ended. Argument handling has only been implemented for two positions:

1. input directory
2. output directory

Core class

The *Core* constructors import the configuration file and overwrite particular settings, if arguments have been provided. So far, the only available constructors beside the default constructor can take the following arguments:

1. input directory
2. input directory, output directory

Further processing is started by the non-static method *main*, which has to be invoked from outside. This will in turn cause the software to iterate over the collection directory tree at the provided input directory, using a recursing method. This method *readDir* creates an instance of class *CollectionObject* on every object, either file or directory, it comes across. On instantiation of the new object, the object takes care of its own analysis and stores the results internally. The completed *CollectionObject* finally is being added to the result tree, which becomes exported to JSON as soon as the extraction process is complete. If the current object is a directory, then a further recursion cycle is started to iterate over the more deeply nested contents. Hidden files or folders are skipped, likewise those beginning with a tilde (~). Besides that, the *Core* class contains further auxiliary methods, e.g. the name splitter method *splitName*, which was already described in section 4.1.1.

CollectionObject class

CollectionObjects describe either files or folders. This class analyzes itself: all one needs to do, is passing a reference of the *Core* class, the file or folder being analyzed and the position of the object in the result tree to the constructor. Here, the object detects whether it is a file or directory, and triggers the according analysis steps, as outlined in section 4.1.1. For a directory, this would be complete with the dissection of the filename[26]. Files have more levels than just the filename, where spatial data could be found. Apache Tika is invoked to retrieve the file's metadata. The retrieved information contains also a notice about the media type, which enables content analysis as the third level of the object's breakdown for some content types. Metadata and content analysis in turn are handled by specialized classes, which are described below. The nested *CollectionObject* structure is finally being displayed in the *GeoCrunch* Viewer, after being exported as JSON. An object representing a directory lists all *CollectionObjects* of files and folders the described directory contains.

FieldDataReader class

The general aims and approaches of this class have already been illustrated in section 4.1.3. The class was created initially to extract spatial references from metadata, which is passed as key value list to the respective methods. Parsing the metadata for places in a hierarchical manner is provided by the method *findPlacenames*, the results are stored in an instance of class *Place*. For collection of associated fields for coordinate pairs or more complex geographical references, the method *findCoordinate* has been created.

Reading data from key-value lists is also quite similar to reading tabular data. The relation between tabular and key-value data becomes clear if one imagines a table with the header row (the keys) and just a single data row (the values). Iterating over a table row by row would thus generate a key-value list on each cycle. As the relevant keyset should be determined only once per table for performance reasons, the method *getCoordinateKeyset* attempts to detect the relevant column names for coordinate extraction. The method *getRowValues* returns the corresponding values of the detected coordinate keyset for each row, that is passed as a key-value list in conjunction with the table headers. Up to now, only coordinate column detection is implemented for tabular data.

CsvReader class

The plain text file content body of CSV files, as extracted from Tika, can be interpreted using this class. As the content body is passed to the constructor, it invokes the method *autodetectFormat*, which tests some combinations of delimiters and quotes. The combination with the greatest number of columns, where the number of header columns equals the

[26] Technically, filename and foldername is the same.

data columns, is regarded as valid and the instantiated class is ready to read the contents using the method *getRow* row by row into key-value arrays, that can be processed from the *FieldDataReader*. This class surely has its problems and might return unexpected results, if the right CSV format could not be detected properly. A well proven open-source solution to perform the format detection has been expected to exist, but could not be found. Although the *CsvReader* has been tested successfully on the provided sample data, it would be worthwhile to circumvent the format detection problem by letting Tika deploy a different content parser by the time of content retrieval. Tika's XML parser might provide an easier way to access tabular data including CSV, as it manages to parse the CSV data into a XML table structure. No further effort has been taken to investigate how they achieve that and how reliable it is.

NERecognizer class

This is the implementation of the Stanford NER package. All actions have been automated, so that the class just has to be instantiated with the content body. Subsequently, the document language has to be detected first. The Apache Tika package again includes a useful tool, the *Optimaize Language Detector*. *Optimaize* now becomes fed with the content body line by line, until it reports to have sufficient text to identify the document language. This way it is avoided to unnecessarily process lengthy documents, just to detect their language. After the language has been determined, the according language model for the actual NER processing can be chosen. With the current development state of *GeoCrunch*, English and German models are available.

Because loading of the models in some cases takes quite long, the models are cached in a *Core* variable in RAM, avoiding unnecessary disk IO. The model is then loaded as a classifier and generates triples including the character offsets (start, end) and the classification key. Depending on the employed language model, the available name classification keys vary. Usually at least persons, organizations and locations are tagged with the model's keys. Only the key names identifying a location are of interest. The resulting list of triples can be retrieved from the *NERecognizer* afterwards.

geoRef package

This subpackage holds some data structure objects, that are being used throughout *GeoCrunch* for internal data exchange.

geoRef.Coordinate

Coordinate is an intermediate data structure that is used to store single parts of a geographical reference (ordinates, cardinality indicators, CRS hints), as they are found during analysis on the content level. Except for genuine geometry formats (GIS shapefiles, vector data etc.), file contents are a little (tabular data) to much less (free text) structured than

metadata. Due to that, spatial references from file contents are not defined as reliably as in metadata and additional validation steps have to be applied. The approved georeference candidates are converted from *Coordinate* into a *GeoRefObject* afterwards.

geoRef.Place

The class *Place* is a hierarchical structure for place indications, similar as class *Coordinate* is a structure for coordinate references. Objects of class *Place* holds the hierarchical place structure, as their levels are illustrated in table 2 on the left.

geoRef.GeoRefObject

GeoRefObject as the resulting object holds a valid or validated geographical reference that consists at least of a coordinate pair. Geocoders return valid *GeoRefObjects*, as opposed to the *CsvParser*, which returns *Coordinate* objects, which have to be validated and converted to *GeoRefObjects*. Additional information includes the detector class which found this spatial reference, file context information (excerpt, line number) and, if applicable, the place name before it was geocoded. Multiple *GeoRefObjects* can be attached to a *CollectionObject*. The *GeoRefObjects* in the resulting JSON object are displayed on the map in the *GeoCrunch* Viewer.

geoRef.GeoRefSource

GeoRefSource and *RefsysSource* are enumerated lists of values to indicate the origin of either a *GeoRefObject* (FILENAME_MENTION, METADATA_MENTION, CONTENT_MENTION, METADATA_COORDINATE, CONTENT_COORDINATE) or of the CRS declaration (ASSUMED, IMPLICIT, EXPLICIT, PROXIMATE). E.g., for the geocoding webservices it is known, which CRS their responses refer to, even if it is not included in the response. The *RefsysSource* would be IMPLICIT, though. PROXIMATE is a value currently never applied, as it is intended for free text coordinate recognition, where a CRS indication could be found *near* a coordinate.

geoRef.CoordinateParser

Finally, this subpackage holds the *CoordinateParser class*, which is being used to validate coordinate references from the file content level. The entrance method, *parseCoordinate* can take a *Coordinate* object and tries to evaluate its contents using different parsers. Up to now, a parser for polar ordinates has been implemented. Method *parsePolarOrdinate* takes a string value, which is attempted to be parsed into a float value, respecting different number formats. If illegal characters occur or the number string is misshaped, the method returns an error code. Polar coordinate numbers in general are expected to be within a range between -180°and +180°. If the method's second argument is given, a single character which specifies the cardinality, more accurate validation can be performed. Y-axis ordinates (n: North; s: South) are not allowed to exceed 90°positively or negatively. If a

cardinal has been passed as a second argument, a possibly found cardinal in the parsed string must match the axis of the provided cardinal. If the passed or found cardinal does not comply with standard easting or northing (cardinality is West or South), the ordinate value becomes inverted.

webservice package

The last subpackage contains the geocoding webservice implementations.

webservice.Geocoder

The abstract class *Geocoder* provides generic constructors to all implementations, the obligatory variables and method declarations. Two constructors are available: the first is able to take a simple query string, whereas the other expects an instance of *Place*. Here, *Place* is concatenated into a simple string, to provide this as a fallback alternative for geocoding implementations, that do not support multi field queries. It is up to the individual implementation, how optional use of the hierarchical *Place* object or the query string is handled.

webservice.<ClientImplementations>

Next to the above abstract *Geocoder*, the implementations of *DaiGazetteer*, *Geonames* and *MapquestOpen* can be found.

webservice.InputType

InputType is an enumeration of values for the origin of a place entry:
DIRECTORY_NAME, FILE_NAME, ENTITY_MENTION.

4.1.8 Configuration and Optimization

The configuration files are located underneath the folder named config. There are two of them, one called default.properties, the other user.properties. Properties are defined in a simple key-value format. Both are read in by the *Core* class constructor, where default.properties is read first in order to set important default settings. The contents from the other file user.properties will overwrite the default settings if the keys match. An overview of the available options can be obtained by inspecting the default settings file. It is not recommended to make changes or customizations in the file default.properties, so the standard settings can easily be restored by removing, truncating or replacing the user defined properties with a fresh copy of the default properties file.

The file default.properties provides a set of adjustments already optimized for greater precision results. The most relevant keys towards geocoding precision are *minimumNameLength* and *ignoreNames*, which affect the results of the file name splitter function only.

Config Key	**Value Definition**	**Description**
inputDirectory	slash delimited, relative to working directory or absolute path	The data directory being scanned.
outputDirectory	slash delimited, relative to working directory or absolute path	Directory to write the analysis output to.
outputFilename	string filename	
parseCsv	string 'true' or "false"	Whether or not CSV files are evaluated (reduces large result numbers).
parseNamedEntities	string 'true' or "false"	Whether or not free text files are evaluated by NER (in case the geocoding query limits are hit, or to reduce large result numbers).
enableWebservices	string 'true' or "false"	Disable geocoding webservices completely (for testing purposes without consuming the geocoding query budget).
countryScope	optional, comma separated list of 2-letter ISO contry codes	Not supported by the DAI Gazetteer.
boundingBox	optional, comma separated list of floating point numbers	Results outside the box will be discarded.
minimumNameLength	integer greater or equal 0	The minimum name length required for file name geocoding.
ignoreNames	comma separated list	File name fraction strings excluded from geocoding.
geoCoderPrecedence	comma separated list of full qualified geocoder class names	Geocoders listed first are expected to deliver more relevant results. As a prior returns a valid result, all second-tier services are skipped.
geonamesUser	string user name for the geonames service	The provided access keys enable free, but limited queries. If the query limit is exceeded, try again later or issue your own key with the according API provider.
mapquestKey	string API key for the mapquest services	

Table 4: Configuration keys

Ignoring short strings and certain name strings increases precision while it has only little impact on recall. In addition, the process terminates earlier and the geocoding query limits are hit much less. The length of *minimumNameLength* has been increased up to three letters due to the fact, that the name splitter function often returns very small fragments, which in most cases do not denote a place name. Very short place names do exist and one should be aware of that. Especially, when places with short names are in question. Wikipedia offers a listing of such short place names, which provides some orientation[27] whether or not one should consider changing the minimum name length setting. The list of ignored name strings works in a similar manner for the longer file name fragments, which typically show up in some file names. These produce geocoding results (e.g. new, data, dsc, neu, paper), but their meaning mostly indicates other characteristics rather than relate to any place. As file names are no proper natural language, 'part of speech' rules cannot be applied to determine the particular meaning of those fragments. Thus, the only workaround seems to be the creation of such an exclusion list. However, this list should be adapted to meet the specific nature of the analyzed data collection, where special file naming rules might have been applied.

Exclusion is a strategy also applicable to narrow down the geocoding results. If a data collection is focused on a limited region, most geocoding service APIs offer options to confine the query results. Available options in the configuration files are *countryScope* and *boundingBox*, which are explained in table 4.

What might also be specific for use of the *GeoCrunch* software for various purposes, is the setting *geoCoderPrecedence*. As a particular geocoding webservice appears more specific with respect to a certain collection or purposes of subject and field, an adaption of the order of available geocoders should be considered. It is also possible to implement further geocoding services by extending the abstract class *Geocoder*, as specified in section 4.1.7.

4.2 Visualization

The *GeoCrunch* Viewer visualizes the results of the previous spatial data extraction process in such a way, that the analyzed collection can be explored spatially quite comfortably and interactively. The Viewer itself can be found in the folder GeoCrunch/webapp. There, the analysis results can be viewed by opening the file index.html in a standard conform browser. The three subsequent sections discuss the GUI elements, which the Viewer provides.

Some general remarks concern the development of the Viewer software as a browser based application. It has been built using the Bootstrap CSS framework[28] for quick creation of the layout and some GUI elements. Leaflet[29] is used to create the interactive

[27] https://en.wikipedia.org/wiki/List_of_short_place_names
[28] http://getbootstrap.com/
[29] http://leafletjs.com/

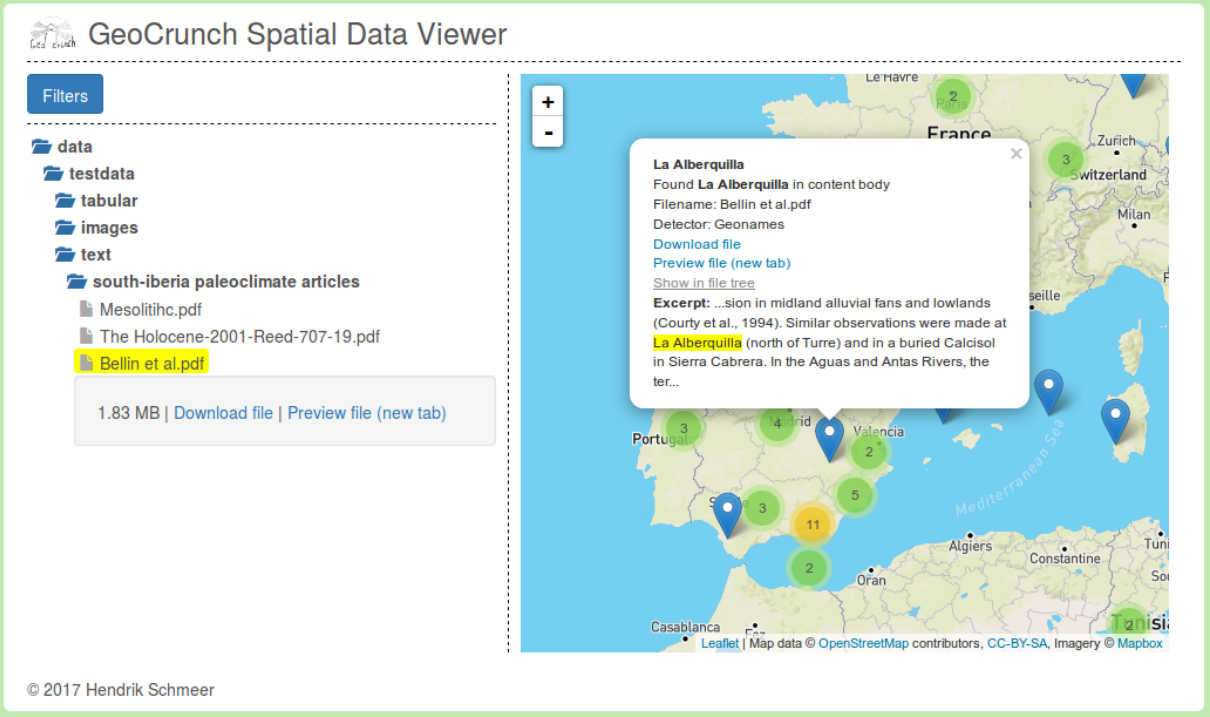

Figure 3: GeoCrunch Viewer

map panel with markers and pop-ups. The map tile layers are drawn from Mapbox[30], where another API key is required, and passed during initialization of the map object in the Javascript file[31]. Markercluster[32] is a Leaflet plug-in which provides clustering of confusingly large marker numbers. Of course the above are built on top of the Javascript framework jQuery[33], which is also being used throughout the development of the Viewer software. All resources except for the map tiles are included in the software in order to make it work more reliably, and also in offline environments. The Javascript file containing all functionality for the *GeoCrunch* Viewer can be found at GeoCrunch/webapp/js/filetree.js.

As for the settings *outputDirectory* and *outputFilename* in the Extractor tool configuration (see table 4), the results of the analysis process are exported as a JSON object to the file GeoCrunch/webapp/json/filetree.json. Contents of this file become visualized by the *GeoCrunch* Viewer in two different ways. First, the filetree JSON object replicates the directory structure of the data collection and the file tree view can be built accordingly. Second, all *CollectionObjects* exported into the JSON object are put on the map as markers. The marker popups provide a neat overview of the attributes, without displaying all of them.

The data input folder for the *GeoCrunch* Extractor is also located in the web application, in order to make the analyzed files available for download in the browser while using

[30] https://www.mapbox.com/

[31] This key has to be replaced, if the provided key does not meet the usage requirements or the provided key has been revoked.

[32] https://github.com/Leaflet/Leaflet.markercluster

[33] https://jquery.com/

relative paths. If the data input directory in the Extractor configuration file has been changed, files cannot be downloaded anymore using the Viewer. Also image previews in the pop-ups require the availability of the data at the pre-configured directory. Similarly, changing the output parameters of the Extractor tool configuration would put the Viewer out of synchronization.

4.2.1 Directory Tree View

Located on the left of the Viewer is a file browser. It becomes rendered using a recursing function by parsing the JSON object, which holds the resulting nested *CollectionObjects* (see sections 4.1.7) of the previous extraction process. Each *CollectionObject* either represents a file or directory. In the case of directories, a new recursion cycle is started to render the deeper nested items. After the generation of the directory tree markup has finished, the root directory receives a click event to render it open. Now, clicking on any of the visible sub-items will give focus to them, which is indicated by the yellow highlighting. Some additional information will also be expanded beneath. If the focused item is a directory, the folder icon will change to a folder-open icon and the enclosed objects will be expanded. At the same time, the map view becomes synchronized with the element in focus.

4.2.2 Map View

The map always shows all the geographical references as markers, that have been found during the analysis of the focused *CollectionObject*. By default, the function generating the map markers recurses more deeply into the directory tree, if nested items are contained in a selected folder. Thus, the map will show all the markers of the entire collection if the root level directory is selected in the tree view pane. Selecting a file or directory that does not contain further elements will only show the place references related to that item. Alternatively the first filter setting (Selected Item Only — Selected Item & Subtree) is intended to cut recursion on demand.

The markers on the map all have a pop-up bound to them, which will appear if one clicks on a marker. Information in the pop-up depends on the origin, type and level of the spatial reference. E.g., a detected coordinate will appear as a coordinate on the first line. Place entries, either from file name, metadata or content, will be displayed with the geocoded place name first, then the original query string. The next information tells about the detector class or geocoder that discovered the spatial reference. Furthermore a download link and an excerpt containing the context of the find spot or an image preview is being offered.

To connect the map back to the directory tree view to the left, the link *Show in filetree* synchronizes the directory browser to the selected marker. Clicking this link will

give focus to the file or folder, that is related to the marker being viewed. All parent directories of the affected element will be opened, and the element becomes scrolled to the top. Now the map view is rendered anew, as it is triggered by passing the focus to another item, either by user click events or programmatically. The markers shown on the map then are confined to those, which are related to the selected collection object or its descendants.

4.2.3 Map Marker Filters

Hidden on top of the left Viewer column, a filter pane can be found, where the display of markers on the map can be narrowed down. A click on the 'filter' button will show the filter form. The available options are explained in table 5.

Filter Key	**Values**	**Description**
DEPTH	Selected Item Only — Selected Item & Subtree	Whether or not the map marker collection recurses on deeper nested objects.
TYPE	Files — Folders — Both	The object types to display.
FILENAME_MENTION	on — off (checkbox)	Display geocoded place names from file names.
METADATA_MENTION	on — off (checkbox)	Display geocoded place names from file metadata.
METADATA_COORDINATE	on — off (checkbox)	Display coordinates from file metadata.
CONTENT_MENTION	on — off (checkbox)	Display geocoded place names from file content.
CONTENT_COORDINATE	on — off (checkbox)	Display coordinates from file content.

Table 5: Map marker filter options

5 Analysis Result Discussion based on Sample Data

The *GeoCrunch* software has finally been used to process some sample data sets which are either chosen to demonstrate the software functionality, originate from the discipline of archeology or otherwise present the limits and the potential with regard to possible research questions. These collections are discussed based on the visualized results of the previous analysis process. A qualitative approach is favored that attempts to determine, how *GeoCrunch* can be utilized to support humanistic research questions in archeology and beyond. The analyzed sample collections can be found in the appendix.

5.1 Test and Sample Data Package

The first package, Appendix A, is intended to provide an overview of *GeoCrunch's* capabilities and to get new users started. It therefore contains a quickstart guide in README.md and an executable .jar file of the *GeoCrunch* Extractor, which can be used to generate spatial analysis results on new data collections and the *GeoCrunch* Viewer.

The embodied data has been picked more or less randomly to test and demonstrate the Extractor functionalities on some real-world data. Three different data types have been enclosed: images, tabular CSV data and a collection of scientific PDF papers as text files. The analysis results are discussed regarding these three categories in the text below.

Images

Digital, geotagged photography has been chosen from different camera types for testing purposes. The coordinate fields contained in the metadata are being detected correctly and show up in the Viewer in every case. Additionally, two images of Namibia rock paintings from the Brandberg collection have been added that becomes discussed more closely below in section 5.2. The images were created long before the GPS era started. However, these images have been curated and now hold proper place references in their metadata, which are also detected and geocoded successfully by the *GeoCrunch* Extractor for display in the Viewer.

Tabular / CSV

Two files have been selected from IANUS' second collection (discussed in section 5.2). Both files contain basically the same data, but as curators decided to make them available to a greater variety of applications, the original .xlsx spreadsheets are simultaneously offered in a CSV format. Hence either of them carry a place reference in their file names, which can be geocoded effectually, both of them are represented by at least one marker on the map. Finally, the implemented CSV parser is able to extract a lot more geographical coordinate references from the according file.

Text / PDF

PDF files are not the only media containing text that could be processed by NER. The list of file types where this strategy can be applied, could be extended in future. Nevertheless, the attached PDF papers can show the potential and limitations of the NER and geocoding approach applied here. Surely a lot of place references are located by the NER implementation, but typically the *recall*, as well as the *precision* or *F-measure* (their harmonic mean), never reach 100%. The exact measures are not of interest, as this bachelor thesis aims to assess the results qualitatively.

Subsequent geocoding makes the situation regarding the IR performance worse. With

regard to *recall*, some of the true-positive place references correctly identified by the NER parser will get lost by falsely or unsuccessful geocoding. The same applies for *precision*, where errant geocoding might add further irrelevant results to the region in question.

Some of the PDF file names could not be geocoded accordingly. Most remarkably, the file containing 'SüdIberia' in its name does not show up on the map, neither does the surrounding folder with the substring 'south-iberia'. As all geocoding implementations have been tweaked towards greater precision results, nothing will be returned. Basically, because a composite entity like this seemingly does not exist in the geocoder databases, as it can hardly be defined due to the fact that it does not have evident borders.

Examining the results of the PDF content analysis further shows several markers which are related to bibliographical quotations. With regard to spatial reference extraction, these results are correct. However, they have nothing to do with the document's subject.

Despite these problems related to NER and geocoding appearing ruinous for the system's technical benchmarks, the results presented in the Viewer have been appreciated as useful. The map view can be zoomed and panned accordingly to cover just the area of interest. Thereafter, the context information provided on every marker allows for a quick decision whether or not the extracted location reference on the map and the linked document is of relevance for the envisioned research on past climates in southern Spain.

5.2 IANUS Data Analysis

Appendix B and Appendix C both are collections of an actual archaeological archival institution and have been processed to investigate the benefits of this bachelor thesis software for the archival context in archeology. While the collection *Holozängeschichte der Tierwelt Europas* (Benecke, Driesch, and Heinrich 2016) covers all of Europe with ten-thousands of markers from CSV data, the second, *Felsbilder im Hohen Brandberg (Namibia)* (Lenssen-Erz and Pager 2016) only has relatively few references originating from geocoded image metadata.

Holozängeschichte der Tierwelt Europas

The number of spatial references in the Holocene collection results in a total number of about 15.000 map markers, which pushes the browser to its technical limits. Initially, the browser freezed for a few seconds on page load. For this reason, chunk-wise loading of the markers had to be activated, which does not block the browser until loading completes.

As the Holocene collection largely consists of tabular data, which can be processed quite reliably, as opposed to the NER-geocoding process, the *recall* of the extraction process would presumably range near 100%. On the contrary, nearly half of the collection's file contents remain unexamined, as this would double the results unnecessarily with

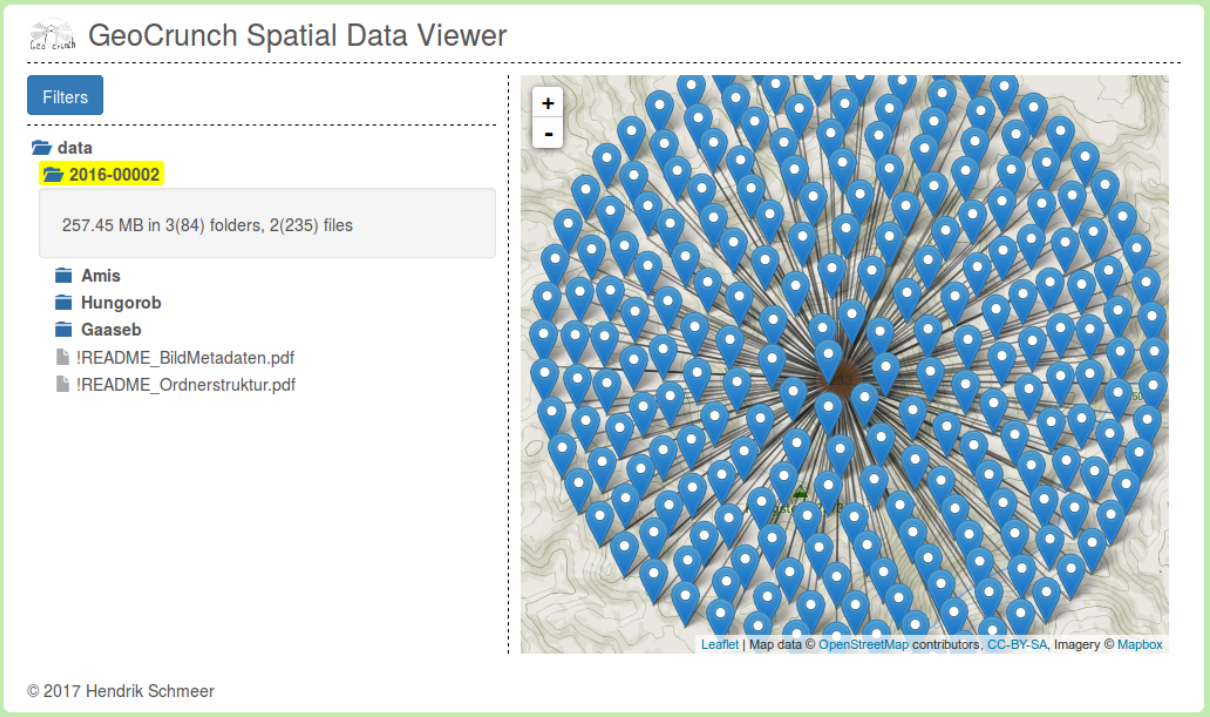

Figure 4: Brandberg analysis

redundant matter: the CSV files duplicate the .xlsx spreadsheets data. Another problem is that files in the subdirectory Katalogdateien/Geodaten summarize the entire data collection, which is divided into separate listings for the European countries. So the extracted data duplicates again.

To utilize the extracted data for purposes of data retrieval or exploration of the archived matter, it becomes apparent that the *GeoCrunch* results need further curation, or thourough selection of the data input to avoid redundant matter. In this sample analysis a bounding box setting has been applied to the NER-geocoding extraction process, as the bounding box is supplied on the collection presentation page[34]. Nevertheless, the *GeoCrunch* Viewer provides an handy overview of the collection and can support selection of particular data sets by displaying locational context and surrounding data next to each other.

Felsbilder im Hohen Brandberg (Namibia)

The Brandberg collection in Appendix C depends completely on the geocoding service capabilities to resolve the detected references, as no explicite coordinate references are included at all. The collection consists of non-geotagged images, but metadata place indications of this collection are nevertheless rather concise. They include the country (Namibia), the region (Brandberg) and the nearby place name or town (e.g. Amis or Hungorob) on the lowest level.

Unfortunately, geocoding services providers such as Geonames and the iDAI.gazetteer

[34] http://www.ianus-fdz.de/datenportal/collections/view/2016-00001_DIP

turned out not to include the lover level names. They are solely able to resolve the region name 'Brandberg' in many cases. The resulting visualization is illustrated in figure 4. It seems that in case of this collection *GeoCrunch* cannot provide further support for the exploration of the data, as long as the requested geocoding services do not include these low-level place names.

Tracking the problem further down shows that not all geocoders handle the hierarchically mapped place references in the same way. Where Geonames seemingly is able to identify particular places in folder names (e.g. Hungorob), the detector that returned the image metadata result is the *DaiGazetteer* class, which does not know about Hungorob. Due to the geocoder preference, the iDAI.gazetteer becomes queried first. For simple query strings this is not so much of a problem, because if there is no result returned from the first, the second service is requested. On the one hand, all results will have the same accuracy for simple place names. Geocoded hierarchical place references, on the other hand, may return results of divergent accuracy which do not necessarily match the provided geocoder precedence. A possible solution would be to implement a second geocoder precedence, that is solely applied on geocoding of hierarchical place references and which lists those geocoders first, that provide better handling of such queries. Otherwise, a more advanced handling of hierarchical query maps can be implemented within the *CollectionObject* class, instead of the current handling in the geocoder webservice implementations.

The next limitation becomes apparent in view of the large number of objects in the collection. Even though the *GeoCrunch* Extractor manages processing the given amount of data and no geocoding limits are exceeded, the Viewer crashes. Despite a wide larger number of markers has been processed by Leaflet and the MarkerCluster plugin for the Benecke collection, this time the huge cumulation of markers onto a small area seems to overstrain the cluster functionality. The provided sample collection therefore only contains a small portion of the original data.

5.3 Paleoclimate Collection

This data collection, Appendix D, was collected by a fellow researcher, who was interested in selecting these papers for her particular focus of historical climates in southern Spain. However the articles cover varying areas, they were tagged with 'Spain' in the IR systems they were retrieved from. The analysis provided in the *GeoCrunch* Viewer facilitates an overview of the embodied locations and allows a more finely nuanced spatial selection of the documents.

Regarding the problem with bibliographical entries mentioned in the results discussion of the sample data set in section 5.1 above, a solution to get rid of exactly this class of references could not be figured out within the scope of this thesis. A specialized

classifier that becomes trained on a corpus of scientific articles would be able to identify bibliographic entries.

For now, the resulting set of markers can solely be reduced by running the analysis with a bounding box covering the area of interest. This provision would cut a fair amount of irrelevant spatial references coming from bibliography entries, but not all of them. The factual analysis performed on the packaged data has been performed without that box, as it is easy enough to zoom down the map until solely South-Spain with its located text references becomes visible. This way, the same collection can moreover be explored with other regional interests as well.

6 Conclusion

The previous discussion of analysis results on different data collection shows that *GeoCrunch's* envisioned purpose to support exploration of data collections can be achieved under most circumstances. In the archaeological archive setting several use cases are at issue. On the one end, prior to data ingestion into the archive, metadata has to be collected and extracted out of the available data. It is the curation process, where *GeoCrunch* can support exploring the data spatially. On the other end, the archive users need to explore the archive with their respective research questions themselves, either by running queries against the archive or by just looking on the spatial structure of a certain collection.

The missing ability to perform spatial queries against larger corpuses distinguishes *GeoCrunch* from GIS or IR systems, as they do provide this feature. Its ability to extract spatial metadata may help to build such a system, though. Nevertheless, an intermediate curation would be required to review the extracted data before they can be incorporated into a search index.

As opposed to IR systems, the *GeoCrunch* Viewer shifts the judgment about the spatial relevance of a document towards the human user, who has to handle the remaining uncertainty of the previous information extraction process. With regard to the ideas of a GIS for the humanities depicted in section 3.2.2, *GeoCrunch's* interactive Viewer would comfortably fit into the humanistic setting, which demands for fuzzy queries on various resources. *GeoCrunch* can extract the spatial attributions of them. Fuzzyness is being achieved by providing a visual interface, that allows the user to explore the presented data freely. The user's interests are not limited by a previous IR process. However, currently the extent of a collection is the limit that the user cannot overcome.

Finally, using *GeoCrunch* on a personal computer environment is possible without the need to setup any local server software, as the Viewer is completely browser based. Thus, the individual use for data assessment based on its spatial dispersion is within grasp of researchers, as it could be showed with the evaluation of paleoclimate articles in

section 5.3. The software could even cover private interests, for instance travel diaries, which might in turn, whether historical or not, also appear to be a reasonable subject of several humanistic disciplines.

References

Adobe Systems Incorporated (2012). *Extensible Metadata Platform (XMP) Specification: Part 1, Data Model, Serialization, and Core Properties*. URL: http://wwwimages.adobe.com/content/dam/Adobe/en/devnet/xmp/pdfs/XMP%5C%20SDK%5C%20Release%5C%20cc-2016-08/XMPSpecificationPart1.pdf (visited on 08/12/2016).

AG Modellierung der Kommission „Archäologie und Informationssysteme“ im Verband der Landesarchäologen der Bundesrepublik Deutschland (2011). *ADeX, Version 2.0: Archäologischer Daten-Export, Standard für den Austausch archäologischer Fachdaten*. URL: http://www.landesarchaeologen.de/fileadmin/Dokumente/Dokumente_Kommissionen/Dokumente_Archaeologie-Informationssysteme/Dokumente_AIS_ADeX/ADeX_2-0_Doku.pdf (visited on 08/12/2016).

Ayers, E. L. (2010). “Turning toward Place, Space, and Time”. In: *The Spatial Humanities: GIS and the Future of Humanities Scholarship*. Ed. by D. Bodenhamer, J. Corrigan, and T. Harris. Bloomington & Indianapolis: Indiana University Press, pp. 1–13.

Benecke, N., A. v. d. Driesch, and D. Heinrich (2016). *Holozängeschichte der Tierwelt Europas*. URL: http://dx.doi.org/001.mcus7z-2 (visited on 08/12/2016).

Bodenhamer, D. J., ed. (2010). *The Spatial Humanities: GIS and the Future of Humanities Scholarship*. Spatial humanities. Bloomington & Indianapolis: Indiana University Press.

Bodenhamer, D. (2010). “The Potential of Spatial Humanities”. In: *The Spatial Humanities: GIS and the Future of Humanities Scholarship*. Ed. by D. Bodenhamer, J. Corrigan, and T. Harris. Bloomington & Indianapolis: Indiana University Press, pp. 14–30.

Bodenhamer, D., J. Corrigan, and T. Harris, eds. (2015). *Deep Maps and Spatial Narratives*. Bloomington & Indianapolis: Indiana University Press.

Camera & Imaging Products Association (CIPA) and Japan Electronics and Information Technology Industries Association (JEITA) (2010). *Exchangeable image file format for digital still cameras: Exif Version 2.3*. URL: http://www.cipa.jp/std/documents/e/DC-008-2012_E.pdf (visited on 08/12/2016).

Comité International des Télécommunications de Presse (IPTC) (2014). *Information Interchange Model Version 4*. URL: http://www.iptc.org/std/IIM/4.2/specification/IIMV4.2.pdf (visited on 08/12/2016).

Díaz, L. et al. (2007). “Semi-automatic Metadata Extraction from Imagery and Cartographic data”. In: *2007 IEEE International Geoscience & Remote Sensing Symposium* 2007, pp. 3051–3052.

Ell, P. S. (2010). "GIS, e-Science and the Humanities Grid". In: *The Spatial Humanities: GIS and the Future of Humanities Scholarship*. Ed. by D. Bodenhamer, J. Corrigan, and T. Harris. Bloomington & Indianapolis: Indiana University Press, pp. 143–166.

Faruqui, M. and S. Padó (2010). "Training and Evaluating a German Named Entity Recognizer with Semantic Generalization". In: *Proceedings of KONVENS 2010*. Saarbrücken.

Finkel, J. R., T. Grenanger, and C. Manning (2005). "Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling". In: *Proceedings of the 43nd Annual Meeting of the Association for Computational Linguistics*, pp. 363–370. URL: http://nlp.stanford.edu/~manning/papers/gibbscrf3.pdf.

Gregory, I. et al. (2015). "Spatializing and Analyzing Digital Texts: Corpora, GIS, and Places". In: *Deep Maps and Spatial Narratives*. Ed. by D. Bodenhamer, J. Corrigan, and T. Harris. Bloomington & Indianapolis: Indiana University Press, pp. 150–178.

Grossman, D. and O. Frieder (2004). *InInformation Retrieval, Algorithms and Heuristics*. Dordrecht: Springer.

Kemp, K. K. (2010). "Geographic Information Science and Spatial Analysis for the Humanities". In: *The Spatial Humanities: GIS and the Future of Humanities Scholarship*. Ed. by D. Bodenhamer, J. Corrigan, and T. Harris. Bloomington & Indianapolis: Indiana University Press, pp. 31–57.

Lenssen-Erz, T. and H. Pager (2016). *Felsbilder im Hohen Brandberg (Namibia)*. URL: http://dx.doi.org/10.13149/r5f.3fyhip-x (visited on 08/12/2016).

Moretti, F. (2016). *Distant Reading*. 2016th ed. Konstanz: Konstanz University Press.

Soja, E. (2001). "In Different Spaces: Interpreting the Spatial Organization of Societies". In: *Proceedings 3rd International Space Syntax Symposium*. (Atlanta).

Stanford NLP Group (n.d.). *Stanford Named Entity Recognizer*. URL: http://nlp.stanford.edu/software/CRF-NER.html (visited on 08/12/2016).

Tally, R. J. (2013). *Spatiality*. London & New York: Routledge.

The Apache Software Foundation (n.d.[a]). *Apache openNLP*. URL: https://opennlp.apache.org/ (visited on 08/12/2016).

— (n.d.[b]). *Apache Tika - a content analysis toolkit*. URL: https://tika.apache.org/ (visited on 08/12/2016).

W3C Semantic Web Interest Group (2009). *WGS84 Geo Positioning: an RDF vocabulary*. URL: https://www.w3.org/2003/01/geo/wgs84_pos (visited on 08/12/2016).

List of Figures

List of Tables

Listings

Appendices

Appendix A

demo.zip

A package containing the *GeoCrunch* Extractor as an executable .jar file, the Viewer, some sample data and the according preprocessed results. Please explore this package and read the README.md first, in order to get a first impression of what the software does and how it works.

Appendix B

benecke.zip

The package contains the *GeoCrunch* Viewer with the processed collection *Holozängeschichte der Tierwelt Europas* (Benecke, Driesch, and Heinrich 2016).

Appendix C

brandberg.zip

The package contains the *GeoCrunch* Viewer with the processed collection *Felsbilder im Hohen Brandberg (Namibia)* (Lenssen-Erz and Pager 2016).

Appendix D

paleoclimate.zip

This sample package provides an analysis of scientific .pdf articles, which were collected in an attempt to retrieve information regarding historical climates in southern spain.

Appendix E

sourcecode.zip

The complete *GeoCrunch* sourcecode of the Extractor, written in Java, and the Viewer, as a HTML/Javascript web application.
Alternatively, the sourcecode can be retrieved online at GitHub:
`https://github.com/hashmich/geoCrunch`.

Glossary

Abbreviation	Transliteration	Explanation
API	Application Programming Interface	
CLI	Command Line Interface	
CRS	Coordinate Reference System	
CSV	Comma Separated Value list	also formats with other delimiters are summarized as CSV often
DAI	Deutsches Archäologisches Institut	German Archaeological Institute
disk IO	disk Input/Output operations	read/write operations on the hard drive
GIS	Geographical Information System	usually regarded as a mapping tool, based on relational databases for storage and analysis
GUI	Graphical User Interface	
IR	Information Retrieval	how to retrieve just the relevant information?
IT	Information Technology	
JSON	JavaScript Object Notation	a simple way to build nested and key-value based data structures, that can finally be processed by Javascript software running on a client/browser
NER	Named Entity Recognition	a linguistic technology to detect and classify nouns
RAM	Random Access Memory	the computers main memory
UTM	Universal Traversal Mercator projection	
WGS84	World Geodetic System 1984	
XML	eXtended Markup Language	

Hiermit versichere ich **an Eides Statt**, dass ich diese Bachelorarbeit selbstständig verfasst und keine anderen also die angegebenen Quellen und Hilfsmittel benutzt habe.

Die Stellen meiner Arbeit, die dem Wortlaut oder dem Sinn nach anderen Werken und Quellen, einschließlich der Quellen aus dem Internet, entnommen sind, habe ich in jedem Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht. Dasselbe gilt sinngemäß für Tabellen, Karten und Abbildungen.

Diese Arbeit habe ich in gleicher oder ähnlicher Form oder auszugsweise nicht im Rahmen einer anderen Prüfung eingereicht.

Ich versichere zudem, dass der Text der eingereichten elektronischen Fassung mit dem Text der vorgelegten Druckfassung identisch ist.

Köln, 10.1.2017 Unterschrift: