1 Proposed Method

1.1 General Approach

This study employs an exploratory quantitative research design (Jaeger and Halliday, 1998; Olston and Najork, 2010). In order to answer the research question on the evolution and spread of agile governance methods and principles in German and British public administrations, the websites of German and British government ministries on the federal and state level were crawled and analysed with respect to the appearance of the term "agil*" (following the keyword search strategy of Mergel et al., 2018). Thereby gathered insights thus are meant to serve as a proxy for the relevance of agile governance methods in the respective institutions (Branco and Rodrigues, 2006; Ghosh et al., 2013). The following three sub-sections describe the methods applied for data collection, preprocessing, and analysis – all of which were programmed and deployed in the programming language Python (Van Rossum and Drake Jr, 1995).

1.2 Data Collection

The purpose of the data collection process was to efficiently gather as many potentially relevant websites as possible. The process followed the steps identified for routine web crawls by Schäfer and Bildhauer (2012). As the first step of data collection, a list of official websites of federal and state ministries in the UK and Germany was gathered manually by consulting the respective web presence listings from official government websites (see Table 1 in Appendix A). These websites then were crawled with the help of the open source web crawling platform Scrapy (Kouzis-Loukas, 2016), specifying the main urls from which the state / federal ministries can be reached (e.g. "https://www.gov.uk") as the "seed url" (Barbaresi, 2015, p. 115). The crawl was limited to stay within the range of the respective domain and its sub domains (e.g. "gov.uk" and "kent.gov.uk"), and only download pages that contained the keyword. For efficiency purposes and building on the assumption that institutions would not hide information they deemed important on deep pages of their websites, links with greater depth than ten pages were ignored (Scrapy, 2018; Wang et al., 2019). Furthermore, the crawler was set to ignore the robots.txt file, since for many sites effective web crawling would not have been possible otherwise (Sun et al., 2007; see also Barbaresi, 2015, p. 125). The crawls were run on an specifically set up instance on the Google Cloud Platform from servers in Belgium, and closely monitored throughout the whole process.

The general data flow in Scrapy is as follows (see Figure 1; adapted from "Architecture overview" in Scrapy, 2018):

1. The Engine gets the initial Requests to crawl from the Spider.
2. The Engine schedules the Requests in the Scheduler and asks for the next Requests to crawl.

3. The Scheduler returns the next Requests to the Engine.
4. The Engine sends the Requests to the Downloader, passing through the Downloader Middlewares.
5. Once the page finishes downloading the Downloader generates a Response (with that page) and sends it to the Engine, passing through the Downloader Middlewares.
6. The Engine receives the Response from the Downloader and sends it to the Spider for processing, passing through the Spider Middleware.
7. The Spider processes the Response and returns scraped items and new Requests (to follow) to the Engine, passing through the Spider Middleware.
8. The Engine sends processed items to Item Pipelines, then sends processed Requests to the Scheduler and asks for possible next Requests to crawl.
9. The process repeats (from step 1) until there are no more requests from the Scheduler.

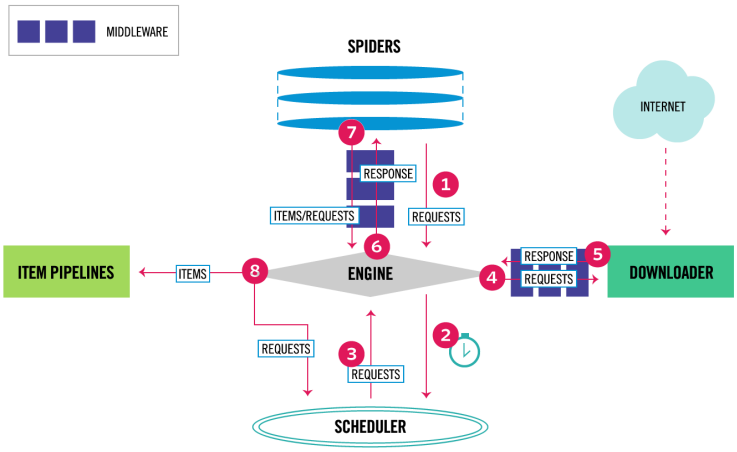

Figure 1: Scrapy Architecture (Source: Scrapy, 2018)

In order to search for the respective keyword on a crawled page as well as retrieve and store the relevant web content, the Spider was setup as follows:

1. The HTML file is parsed into a "soup" object with the help of the *Beautiful Soup* package (Richardson, 2007).
2. The text of the HTML file is extracted.
3. Based on the CONDITION if the search term agil* (in regex: '\bagil.?') is present or not the following steps are performed:

    a. IF the search term is present:

        – A random ID is created for the page, based on its URL and stored as an item for later processing in the Pipeline.

- The page's URL and domain are stored as an item for later processing in the Pipeline.
- Four different (potential) publishing date instances, as well as the document's main heading are extracted with the help of XPath expressions and stored as items for later processing in the Pipeline (w3schools.com, 2020).
- The HTML file is stored in a newly created directory (domain name) under the newly created ID.
- The path name is reported as "saved file" in the log file.

b. IF the search term is NOT present:

- The page is being skipped (reported as "skipping" in the log file).

4. The process repeats from step 1.

To not get stuck with trying to decipher non-HTML files, all other common file extensions (e.g. "flv", "css", "pdf") were denied for extraction. After the HTML files were parsed and analysed in the Spider, the items are passed to the Items Pipeline which writes them into a CSV file which then is stored for further processing.

1.3 Data Preprocessing

The purpose of the data preprocessing was to strip the text from unnecessary web-script elements, extract further date timestamps while selecting the earliest one, and finally, delete duplicates and false positives from the data set (e.g. agile related to military) and save everything in machine readable CSV- and text-file formats. These steps are in line with the recommendations of Lüdeling et al. (2015, p. 19) for the preprocessing of downloaded web pages. To perform these tasks, two algorithms were designed and adapted for their language / country specific application.

First Algorithm: One task of the first algorithm was to remove all HTML code and other "boilerplate" elements such as headers or links to other web pages (ibid.). This step was performed with the help of the *Beautiful Soup* package (Richardson, 2007), and the so extracted text was stored as a variable as well as in separate text-files. Next, the search for the respective agile term was iterated, this time with more specific regex expressions[1]. Also, the preceding and subsequent context (five words each) of the first appearance of the search term was captured in order to facilitate the later search for false positives. Furthermore, another search for the explicit mentioning of the term "agile [...] method*" was conducted.

Since there were cases in which the during scarping defined date instances have yielded no results, the other major task of this algorithm was to retrieve further manually identified instances for (potential) date entries from the web pages so that the majority of cases (more than 80%) would be equipped with at least one date entry. To achieve this, the parser package *lxml*(Faassen, 2006) was used besides

[1] E.g. the regex expression to catch all English matches was specified to be '\bagil\b|\bagility\b'.

Beautiful Soup in order to be able to read XPath (Clark and DeRose, 1999). Since for most relevant cases the earliest date retrieved matched the date when the content of the website was first published,[2] a "final date" variable was created with the earliest date as entry for each case.[3] To correctly parse the dates into Python datetime objects, the *dateutil* package (Niemeyer, 2003) was used. Finally, the algorithm stored all the newly generated variables (search term matches, context of first match, date entries, text, and location of the text files) into a CSV file.

Second Algorithm: The task of the second algorithm was to get rid of all the duplicates and false positives in the data set. For both of these tasks the *pandas* Python package (McKinney, 2010) was used. To erase the false positives two strategies were used. The first strategy was to erase all cases, where the more restrictive regex match on the further cleaned text (see first algorithm above) did not yield any results. The second strategy was to manually look through the context of the remaining matches to identify and remove all cases where the content was about a subject matter not of interest for this study – for example, "agile" military in the UK context or "agile Senioren" (Eng: agile seniors) in the case of German websites. Finally, the duplicate cases where identified based on the heading as well as the context of the first match and thusly erased, so that the version which dates back the longest remained in the data set.

Everything above has so far only been deployed for the UK and all now following descriptions in this section are work in progress.

I plan to have a look into the *SpaCy* package (Honnibal and Montani, 2017) in order to use named entity recognition and word vectors to make the extraction of false positives potentially more automatable.

1.4 Data Analysis

In the data analysis part, I want to show the differences in relevant websites published per year across countries and federal states. One way to do so, could be to use a number of bar charts like Figure 2 and stack them above and next to each other. Furthermore, once all data has been gathered and preprocessed, I will try to create an *"Agile Methods Intensity Index"*. Potentially, this could be done by not only counting the websites published per year but also going into details, as to whether these websites only mention agile / agility once or whether they provide a thorough description of the method's application. To analyse and visualize the data I plan to use the packages *pandas* (McKinney, 2010), *matplotlib* (Hunter, 2007), and *plotly* (Plotly Technologies Inc., 2015). Furthermore, I plan to dive into *SpaCy* to see in which sense it might help me with constructing the index.

[2] This was tested manually by checking a sample of cases with multiple date entries.

[3] The retrieved final date was also compared with the results generated by the *HTMLdate* package (Barbaresi, 2020), which was particularly designed to find publication dates of web pages. It turned out that the approach of the author of this paper was more robust and yielded more accurate results.

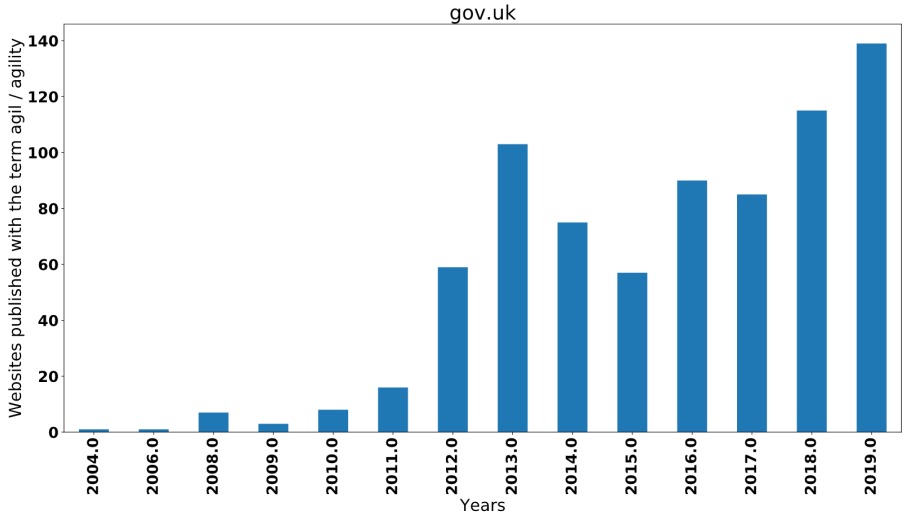

Figure 2: Counts of published websites for gov.uk

2 Experiments

Before the above described crawler setup was run on the Google Cloud Plattform, multiple test runs to iterate and improve the code and the process have been conducted on the author's own machine, a MacBook Pro 2018 with 2.9 GHz Intel Core i9 Processor and 32 GB of RAM memory. Also, the steps for data preprocessing and data analysis were piloted on the data set of the UK.

Is that enough for the Experiments part? Or should I just leave that section out and put the information about pilots and tests at the end of the Data Analysis section?

References

Barbaresi, A. (2015). *Ad hoc and general-purpose corpus construction from web sources* (Doctoral dissertation). ENS Lyon. https://doi.org/HALId:tel-01167309

Barbaresi, A. (2020). Adbar/htmldate: Find original and updated publication dates of web pages using common patterns, heuristics and robust extraction. https://zenodo.org/record/3611241#.XlgETRP0mhc

Branco, M. C. & Rodrigues, L. L. (2006). Communication of corporate social responsibility by Portuguese banks: A legitimacy theory perspective. *Corporate Communications*. https://doi.org/10.1108/13563280610680821

Clark, J. & DeRose, S. (1999). XML Path Language (XPath). *W3C Recommendation*.

Faassen, M. (2006). lxml - XML and HTML with Python. https://lxml.de/

Ghosh, T., Anderson, S. J., Elvidge, C. D. & Sutton, P. C. (2013). Using nighttime satellite imagery as a proxy measure of human well-being. *Sustainability (Switzerland)*. https://doi.org/10.3390/su5124988

Honnibal, M. & Montani, I. (2017). *SpaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing*.

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science and Engineering*. https://doi.org/10.1109/MCSE.2007.55

Jaeger, R. G. & Halliday, T. R. (1998). On Confirmatory versus Exploratory Research. *Herpetologica*.

Kouzis-Loukas, D. (2016). Learning Scrapy. Packt Publishing Ltd.

Lüdeling, A., Evert, S. & Baroni, M. (2015). Using web data for linguistic purposes (M. Hundt, N. Nadja & C. Biewe, Eds.). In M. Hundt, N. Nadja & C. Biewe (Eds.), *Corpus linguistics and the web*. https://doi.org/10.1163/9789401203791{_}003

McKinney, W. (2010). Data Structures for Statistical Computing in Python. *Proceedings of the 9th Python in Science Conference*.

Mergel, I., Gong, Y. & Bertot, J. (2018). Agile government: Systematic literature review and future research. *Government Information Quarterly*, *35*(2), 291–298. https://doi.org/10.1016/j.giq.2018.04.003

Niemeyer, G. (2003). Dateutil - powerful extensions to datetime. https://github.com/dateutil/dateutil

Olston, C. & Najork, M. (2010). Web crawling. *Foundations and Trends in Information Retrieval*. https://doi.org/10.1561/1500000017

Plotly Technologies Inc. (2015). Collaborative data science. https://plot.ly

Richardson, L. (2007). Beautiful Soup Documentation. https://www.crummy.com/software/BeautifulSoup/bs4/doc/

Schäfer, R. & Bildhauer, F. (2012). Building large corpora from the web using a new efficient tool chain. *Proceedings of the 8th International Conference on Language Resources and Evaluation, LREC 2012*, (2006), 486–493.

Scrapy. (2018). Scrapy 1.8 Documentation. https://docs.scrapy.org/en/latest/index.html

Sun, Y., Zhuang, Z. & Giles, C. L. (2007). A large-scale study of robots.txt, In *16th international world wide web conference, www2007*. https://doi.org/10.1145/1242572.1242726

Van Rossum, G. & Drake Jr, F. L. (1995). *Python Tutorial*. Amsterdam, The Netherlands, Centrum voor Wiskunde en Informatica.

w3schools.com. (2020). XPath Syntax. https://www.w3schools.com/xml/xpath_syntax.asp

Wang, C., Zhao, S., Kalra, A., Borcea, C. & Chen, Y. (2019). Webpage Depth Viewability Prediction Using Deep Sequential Neural Networks. *IEEE Transactions on Knowledge and Data Engineering*. https://doi.org/10.1109/TKDE.2018.2839599

Appendix A

Table 1: List of ministry websites

Country / Level / Ministry	Source / Website
Britain – Federal & State Level	www.gov.uk/government/organisations
All institutions are reachable under one domain	www.gov.uk
Germany – Federal Level	www.protokoll-inland.de/Webs/PI/DE/rang-titulierung/amtliche-reihenfolgen/bundesministerien/liste-der-bundesministerien-node.html
Auswärtiges Amt	www.auswaertiges-amt.de
Bundesministerium der Finanzen	www.bundesfinanzministerium.de
Bundesministerium der Justiz und für Verbraucherschutz	www.bmjv.de
Bundesministerium der Verteidigung	www.bmvg.de
Bundesministerium des Innern, für Bau und Heimat	www.bmi.bund.de
Bundesministerium für Arbeit und Soziales	www.bmas.de
Bundesministerium für Bildung und Forschung	www.bmbf.de
Bundesministerium für Ernährung und Landwirtschaft	www.bmel.de
Bundesministerium für Familie, Senioren, Frauen und Jugend	www.bmfsfj.de
Bundesministerium für Gesundheit	www.bundesgesundheitsministerium.de
Bundesministerium für Umwelt, Naturschutz und nukleare Sicherheit	www.bmu.de
Bundesministerium für Verkehr und digitale Infrastruktur	www.bmvi.de
Bundesministerium für Wirtschaft und Energie	www.bmwi.de
Bundesministerium für wirtschaftliche Zusammenarbeit und Entwicklung	www.bmz.de
Bundesregierung	www.bundesregierung.de
Germany – State Level	
Baden-Wuerttemberg	www.baden-wuerttemberg.de/de/regierung/landesregierung/ministerien
Ministerium der Justiz und für Europa	www.justizministerium-bw.de
Ministerium für Kultus, Jugend und Sport	www.km-bw.de
All other institutions are reachable under one domain	www.baden-wuerttemberg.de
Bavaria	https://www.bayern.de/staatsregierung/staatsministerien
All institutions are reachable under one domain	www.bayern.de
Berlin	www.berlin.de/rbmskzl/regierender-buergermeister/senat
Senatsverwaltung für Stadtentwicklung und Wohnen	www.stadtentwicklung.berlin.de
All other institutions are reachable under one domain	www.berlin.de

Brandenburg	https://service.brandenburg.de/lis/list.php?page=behoerdenverzeichnis_art&sv[adr_art]=lb_Min&sort=org_name1,org_name2,org_name3,org_name4&_grid=Ministerien
All institutions are reachable under one domain	www.brandenburg.de
Bremen	https://landesportal.bremen.de/senat/ressorts
All institutions are reachable under one domain	www.bremen.de
Hamburg	www.hamburg.de/behoerden
All institutions are reachable under one domain	www.hamburg.de
Hesse	www.hessen.de/regierung/staatskanzlei-und-ministerien
All institutions are reachable under one domain	www.hessen.de
Lower Saxony	www.niedersachsen.de/politik_staat/landesregierung_ministerien/die-niedersaechsische-landesregierung-20076.html
All institutions are reachable under one domain	www.niedersachsen.de
Mecklenburg-Vorpommern	www.regierung-mv.de/
All institutions are reachable under one domain	www.regierung-mv.de
North Rhine-Westphalia	www.land.nrw/de/ministerien-und-vertretungen
Ministerium der Finanzen	www.fm.nrw.de
Ministerium der Justiz	www.justiz.nrw
Ministerium des Innern	www.im.nrw.de
Ministerium für Arbeit, Gesundheit und Soziales	www.mags.nrw
Ministerium für Bundes- und Europaangelegenheiten sowie Internationales	www.mbei.nrw
Ministerium für Heimat, Kommunales, Bau und Gleichstellung	www.mhkbg.nrw
Ministerium für Kinder, Familie, Flüchtlinge und Integration	www.mkffi.nrw
Ministerium für Kultur und Wissenschaft	www.mkw.nrw
Ministerium für Schule und Bildung	www.schulministerium.nrw.de
Ministerium für Umwelt, Landwirtschaft, Natur- und Verbraucherschutz	www.umwelt.nrw.de
Ministerium für Verkehr	www.vm.nrw.de
Ministerium für Wirtschaft, Innovation, Digitalisierung und Energie	www.wirtschaft.nrw
Staatskanzlei	www.land.nrw
Rhineland-Palatinate	www.rlp.de/de/landesregierung/ministerien
All institutions are reachable under one domain	www.rlp.de
Saarland	www.saarland.de/12341.htm
All institutions are reachable under one domain	www.saarland.de
Saxony	www.sachsen.de/regierung-verwaltung-egovernment.html
All institutions are reachable under one domain	www.sachsen.de

Saxony-Anhalt	www.sachsen-anhalt.de/lj/politik-und-verwaltung/die-landesregierung/ministerien/ministerien
All institutions are reachable under one domain	www.sachsen-anhalt.de
Schleswig-Holstein	www.schleswig-holstein.de/DE/Landesregierung/landesregierung_node.html
All institutions are reachable under one domain	www.schleswig-holstein.de
Thuringia	www.thueringen.de
Thüringer Ministerium für Arbeit, Soziales, Gesundheit, Frauen und Familie	www.tmasgff.de
Thüringer Staatskanzlei	www.staatskanzlei-thueringen.de
All institutions are reachable under one domain	www.thueringen.de