Fachhochschule Südwestfalen
University of Applied Sciences

Fachbereich Informatik und Naturwissenschaften, Iserlohn
Studiengang Informatik (B.Sc., BPO 2013)

Bachelorarbeit

Interaktive geographische Darstellung von Wahlergebnissen unter Einsatz von Open Data

Interactive geospatial visualisation of election results using open data

Kevin Arutyunyan

<arutyunyan.kevin@fh-swf.de>

Beginn: 26. Oktober 2020
Abgabe: 21. Dezember 2020

Referent: Prof. Dr. Christian Gawron
Korreferent*in: Prof. Dr. RyLee Hühne

Interaktive geographische Darstellung von Wahlergebnissen unter Einsatz von Open Data

Kurzfassung

Die für eine mit Wahldaten arbeitende Anwendung benötigten Daten werden bislang nicht in einem einheitlichen Format bereitgestellt. Damit Daten in einer Anwendung wie der im Rahmen dieser Arbeit erstellten Web-Anwendung verwendet werden können, ist es wichtig, dass sie maschinenlesbar vorliegen, sowie dass verschiedene Elemente, die über die reinen Stimmenanzahlen hinaus gehen, bereitgestellt werden. In dieser Bachelorarbeit werden Daten gemäß des neuen, sich in Entwicklung befindlichen Standards für Offene Wahldaten verwendet, durch den Datensätze in einem höheren Umfang als bisher verwendbar werden. Hinsichtlich des Datenmodells lässt die Dokumentation allerdings einige Fragen offen, auch die Maschinenverarbeitbarkeit kann optimiert werden. Bei der Implementierung der Datenverarbeitung für die entwickelte Anwendung musste mit Annahmen und Problemumgehungen gearbeitet werden. Der Stand der Bereitstellung offener Daten insbesondere in Zusammenhang mit benötigten Geodaten, sowie Alternativen für die eigene Herleitung dieser Daten, wenn sie nicht offiziell bereitgestellt werden, werden dargestellt. Entstanden ist eine Web-Anwendung, die auf Basis von Leaflet und vektorbasiertem Rendering mit Tangram unterschiedliche Einfärbungen von Gebieten in Form einer Choroplethenkarte darstellt. Es kann zwischen verschiedenen Darstellungen gewechselt werden, außerdem werden Tooltips und Popups bei Interaktion mit Kartenobjekten dargestellt. Damit auf den Erfahrungen und dem Code aufgebaut werden kann, werden sämtliche Ergebnisse offen bereitgestellt.

Interactive geospatial visualisation of election results using open data

Abstract

The data needed for applications that work with election data is not yet published in a standardised format. For it to be used in an application like the web application created in this thesis, it should be available in a machine-readable and complete way, containing more than just vote counts. In this thesis, data in a proposed, work-in-progress format (Standard für Offene Wahldaten) that provides more of the required types of election-related data is used. However, the documentation of the data model leaves questions unanswered and there is room for improvements of the programmatical processability of the data. It was required to work with assumptions and workarounds for the implementation of the data processing based on the current standard. The current state of availability of open data is explored, especially related to the required geodata, as well as alternative ways of data creation in case the required data is not made available officially. The result of this thesis is a web application using Leaflet and vector based rendering with Tangram to show coloured election districts as a choropleth map. It is possible to switch between types of shown data and to get more information interactively, using tooltips and popups. All results of this thesis are made freely available, so that the practical experience and code can be built upon.

Veröffentlichung

Das Projektrepositorium mit Code, der im Rahmen dieser Bachelorarbeit entstanden ist, befindet sich auf GitHub:

`https://github.com/fhswf/wahlkarte`

Dessen Lizenzierung ist den dortigen Angaben zu entnehmen.

Inhaltsverzeichnis

1 Einführung

Informationen zum Ausgang von Wahlen sind nicht nur wichtig für die daraus folgende Bestimmung der Verhältnisse und Mandate zur Repräsentation der Bevölkerung in den Parlamenten. Die Wahlforschung befasst sich mit der Beschreibung und Erklärung des Wahlverhaltens mit verschiedenen Theorien und Ansätzen [1]. Unter den Begriff Wahlgeographie fällt dabei die Wahlforschung in Zusammenhang mit geographischen Phänomenen rund um Wahlen. Das Hauptaugenmerk liegt bei der geographischen Verteilung der Partei- und Kandidatenanteile und den sie beeinflussenden räumlichen Faktoren. Nach dem Verständnis von Wahlgeographie *als Verfahren* wird sie eingeengt auf den Vergleich von Karten, die Wahlergebnisse darstellen, mit entsprechenden Karten anderer Erklärungsgrößen [2].

Auch Datenjournalismus, als insbesondere in diesem Jahrzehnt angewachsene Form des Journalismus, ist zu erwähnen. Der Datenjournalismus setzt auf Datensätze nicht nur als Recherchequelle, sondern macht die Daten zum zentralen Gegenstand, wozu auch entsprechende Visualisierungen gehören [3]. Diese Entwicklung wird dadurch unterstützt, dass es technische Hilfsmittel gibt, die die Datenauswertung erleichtern, und dass die Open-Data-Bewegung wächst. Zum Datenjournalismus gehört auch ein investigativer Aspekt, beispielsweise in Zusammenhang mit Aufdeckung von Korruption [4]. In Zusammenhang mit Wahlen gibt es in der Hinsicht Beispiele zur Anwendung mathematischer Methoden zur Aufdeckung von Wahlfälschung [5]. Für die geographische Darstellung von Wahlergebnissen auf lokaler Ebene als Form des Datenjournalismus gibt es beispielsweise verschiedene Karten des Interaktiv-Teams der Funke Mediengruppe [6]. Veröffentlichungen wie diese sind aber üblicherweise nicht in freier Form als Open-Source-Software zur eigenen Erstellung und Anpassung verfügbar. Außerdem ist davon auszugehen, dass diese Software nicht mit einem einheitlichen Datenstandard wie dem neuen Standard für Offene Wahldaten, der in dieser Bachelorarbeit verwendet werden wird, arbeitet. Auch ist unklar, wie viel Vorbearbeitung für die Inbetriebnahme der Software je zusätzlicher Instanz erfolgen muss.

1.1 Wahlergebnisdarstellungen

Daten zu Wahlen werden beispielsweise im lokalen Zusammenhang von Kommunen in Wahlergebnisportalen bereitgestellt. Zusammenfassungen von Ergebnissen bei regionalen, landes- oder bundesweiten Wahlen sind auch in ähnlicher Form bei den jeweiligen Wahlleitenden zu erhalten.
Dabei handelt es sich allerdings nicht immer auch um Darstellungen in geographischer Form, grundsätzlich werden in solchen Portalen die Gesamt- oder Gebietsergebnisse mindestens mit Grafiken und Tabellen dargestellt.

Geographische Darstellungen von Ergebnissen sind entweder nicht Bestandteil der Wahlergebnisportale oder ihre Verfügbarkeit hängt von einer vorherigen Einpflegung geographischer Daten ab.
Eine geographische Darstellung kann aber von Vorteil sein, um schnelle oder vertiefend

verknüpfende Einblicke in die Ergebnisse zu ermöglichen. Bei einer tabellarischen Darstellung von Ergebnissen je nach Gebiet ist nicht erkennbar, welche geographisch beieinander liegen und möglicherweise übergreifend zusammenhängende Einblicke liefern könnten. Bei der zusammengefassten Darstellung, beispielsweise auf Stadtteil- statt auf Nachbarschaftsebene ist die Anzahl an Gebieten wiederum geringer und lokale Besonderheiten fallen zunehmend weniger auf.
Die Darstellung auf einer Karte ermöglicht durch die Darstellung der jeweiligen Gebiete und einer thematischen Einfärbung ein schnelles Erkennen des Wahlausgangs in sämtlichen Gebieten und das Erkennen räumlicher Zusammenhänge.

Eine geographische Visualisierung erfolgt klassischerweise mit einem Geoinformationssystem (GIS), dies benötigt Vorkenntnisse und Aufwand für das Ermitteln, Einbinden, und Vernüpfen von allen benötigten Daten. Das Ergebnis kann eine statische Karte für den Druck oder die Bildschirmdarstellung, oder aber auch eine exportierte Web-Karte sein [7]. Alternativ gibt es Möglichkeiten, mit weniger erforderlichen Vorkenntnissen solche statistisch basierten Kartendarstellungen über Web-Anwendungen wie Datawrapper [8] zu erstellen.
Die Ergebnisse einer solchen Kartenerstellung sind allerdings häufig dann nur auf den konkreten Fall mit den vorliegenden Daten bezogen und bei gewünschter Darstellung anderer Daten sind gegebenenfalls Schritte zu wiederholen. Außerdem kann daraus keine vollwertige Web-Anwendung, die verschiedene Auswahlen automatisch ermittelt und bereitstellt, resultieren.

Stattdessen kann die Visualisierung in einer nutzendenfreundlichen, auf Wahldaten zugeschnittenen, Web-GIS-Anwendung erfolgen. Für die Erstellung einer solchen Anwendung ist es notwendig, die Wahldaten maschinell verarbeiten zu können, um entsprechende Analysemöglichkeiten zu ermitteln und in einem Interface bereitzustellen, und schließlich auch Verknüpfungen mit den relevanten Geodaten herzustellen.
Außerdem kann eine solche interaktive geographische Visualisierung zusätzlich als Einstiegspunkt für die klassische Darstellung der Wahlergebnisdaten dienen, indem über die Interaktion mit den dargestellten Gebieten die jeweiligen Detailergebnisse in klassischer, nicht-geographischer Form aufgerufen werden können.
Nutzende einer solchen Anwendung benötigen keine Vorkenntnisse der Geoinformatik oder Statistik und müssen schließlich auch keine Entscheidungen hinsichtlich der Dateneinbindung treffen, denn die Verantwortung dafür liegt in dem Fall bei der Stelle, die die Anwendung betreibt und bereitstellt, und dafür nur Grundkonfigurationen durchführen muss, darunter insbesondere, welche der zugrundeliegenden maschinenlesbaren Daten woher geladen werden.

1.2 Maschinenlesbare Wahldaten

Maschinenlesbare Daten zu Wahlen können in Wahlergebnisportalen oder in Open-Data-Portalen vorliegen, dies geschieht aber abhängig von den verwendeten Systemen in unterschiedlichen Formaten und in unterschiedlichem Umfang.

Dies erschwert die Auswertung von Wahldaten insbesondere bei Auswertungen, für die Daten von mehreren Stellen benötigt sind, da es sich schon bei Daten aus Nachbarkommunen um unterschiedliche Formate handeln kann.

Um einen einheitlichen und offenen Standard für Wahldaten zu entwickeln, wurde 2018 eine Arbeitsgruppe unter Beteiligung eines kommunalen Rechenzentrums, Kommunen, dem Wahlsoftwarehersteller vote iT, und der Open-Data-Community gegründet. Ziel sei es, beim Standard „einfach strukturierte Daten nicht durch komplexe Strukturen“ zu ersetzen, sondern zu ergänzen, und somit einen Standard zu schaffen, „der für jeden zugänglich ist, seien es affine Menschen oder Pressevertreter“ [9].
Die Beschreibung des Datenstandards im aktuellen Zustand erfolgt mittlerweile auf der Website offenewahldaten.de.

1.3 Ziele

Kernziel dieser Bachelorarbeit ist die Erstellung einer Web-Anwendung zur interaktiven geographischen Darstellung von Wahlergebnissen.
Es wird erforscht, wie sich Daten nach dem Offenen Wahldatenstandard zur maschinellen Auswertung verwenden lassen und welche Schwierigkeiten sich im Umgang mit dem aktuellen Stand ergeben.
Auch die bisherige Realitätstauglichkeit von Open Data sowie eventuell notwendige Problemumgehungen, insbesondere in Zusammenhang mit Geodaten für die zu entwickelnde Anwendung, werden behandelt.

Die Wahldatenauswertung soll dann eingesetzt werden, um in Verbindung mit Geodaten eine Web-Anwendung zu erstellen, die eine interaktive Darstellung von Wahlergebnissen in Relation zu den jeweils zugrundeliegenden geographischen Flächen ermöglicht.
Die Interaktivität bezieht sich dabei unter anderem auf die Anpassungsfähigkeit des Kartenausschnitts, Auswahl der Analyse (bei Stimmen beispielsweise die Darstellung nach Partei oder nach Platzierung), und die Darstellung zusätzlicher Informationen bei Interaktion mit Kartenobjekten (darunter die Darstellung gebietsbezogener Informationen bei Klick auf ein bestimmtes Gebiet).
Die Grundlage dafür, welche Daten in der Anwendung darzustellen sind, soll eine möglichst aufwandsarme und hauptsächlich auf die Angabe der Datenquellen zugeschnittene Konfiguration sein. Das Interface der Anwendung soll sich dann aus diesen Angaben und den dahinter stehenden zu ladenden Daten automatisiert erschließen, so dass Nutzende der Anwendung keinerlei Datenverknüpfungen oder andere Vorverarbeitungen durchzuführen haben.

Da die verwendeten Daten aus Kommunen stammen, die die Wahlsoftware votemanager von vote iT verwenden, werden im Rahmen dieser Bachelorarbeit insbesondere die dort relevanten Gegebenheiten behandelt, auch weil das Offene Wahldatenformat praktisch sehr auf diesem System basiert, obwohl es grundsätzlich nicht nur auf dieses System beschränkt sein soll.
Eine Abhängigkeit des zu entwickelnden Codes soll allerdings vermieden werden, indem

Faktoren, die sich unmittelbar auf Eigenschaften der Software votemanager beziehen, die aber nicht öffentlich strukturiert dokumentiert sind, möglichst nicht besonders behandelt werden.
Bei verwendeten Wahldaten handelt es sich größtenteils um Daten zu Kommunalwahlen aus Nordrhein-Westfalen. Die Anwendung soll aber nicht darauf beschränkt sein und unterschiedliche Wahlarten unterstützen.

Die Ergebnisse dieser Bachelorarbeit sollen offen bereitgestellt werden, damit auf diesen aufgebaut werden kann. Dies bezieht sich nicht nur auf Anwendungscode, in dessen Zusammenhang insbesondere für den Umgang mit dem Offenen Wahldatenformat bislang keine Implementierung verfügbar ist, sondern auch auf den Ausarbeitungstext, damit es möglich wird, bei der Weiterentwicklung des Datenstandards Erfahrungen mit der maschinellen Verarbeitung der Daten zu berücksichtigen.

1.4 Gliederung

Zunächst werden die Grundlagen zu den Themen Wahlen und Open Data erläutert.
In den Kapiteln Wahldaten und Geodaten werden die notwendigen Datengrundlagen erläutert, auch hinsichtlich der Datenverfügbarkeit, Praxiserfahrungen, und der eigenständigen Erstellung fehlender Daten.
Im Kapitel Umsetzung wird schließlich die Implementierung der Web-Anwendung mit den technischen Grundlagen, der zugrundeliegenden Datenverarbeitung sowie der Visualisierung der Daten behandelt.

2 Wahlen

Die Verwendung des Begriffs „Wahl“ sollte zunächst definiert werden: Oft wird beispielsweise von „der Kommunalwahl“ gesprochen, dabei finden aber unter diesem Begriff üblicherweise mehrere Wahlvorgänge statt, beispielsweise eine Stadtoberhauptswahl oder die Wahl der Vertretung einer Gemeinde („Ratswahl“).
Folgend soll der Begriff so verwendet werden, dass es um so einen spezifischen Wahlvorgang geht. Sollen mehrere gleichzeitig stattfindende Wahlen beschrieben werden, wird von einem Wahltermin gesprochen. An einem Wahltermin können auch ganz unterschiedliche Arten von Wahlen stattfinden, beispielsweise wenn Europawahl und Kommunalwahlen gleichzeitig stattfinden.

2.1 Arten von Wahlen

Wahlen können sich hinsichtlich verschiedener Faktoren unterscheiden, darunter Arten von Kandidaturen, die potenzielle Anzahl unterschiedlicher Stimmzettel, und die Anzahl von Stimmen.

Die einfachste Art von Kandidaturen liegt vor, wenn einzelne Personen zur Wahl stehen, ohne Unterschiede bezüglich Wahlkreisen oder Listen, also beispielsweise bei der Wahl des Stadtoberhaupts. Bei Wahlkreis-/Wahlbezirkskandidaturen gibt es je Wahlkreis unterschiedliche Kandidaturen [10, 11], in der Regel mit je einer Person pro politischer Gruppierung (bzw. eine unabhängige Einzelbewerbung). Als einheitlicher Begriff für solche Gebiete, zwischen denen sich Kandidaturen unterscheiden, wird folgend Kandidaturgebiet verwendet.
Listenkandidaturen liegen je politischer Gruppierung in einer bestimmten Reihenfolge [11] vor und sind bei vielen Wahlarten die einzige Form der Kandidatur (z. B. bei der Europawahl, in NRW bei Bezirksvertretungswahlen oder der Wahl der Verbandsversammlung des Regionalverbands Ruhr).
In anderen Fällen gibt es beide Kandidaturarten, beispielsweise bei Bundestags-, Landtags-, und Kreistags-/Ratswahlen. Auf die Besonderheiten bei Stimmzetteln und bei der Stimmenzahl wird folgend noch eingegangen.

Bei Kreistags-/Ratswahlen zur Kommunalwahl in Nordrhein-Westfalen gibt es zwar aufgrund unterschiedlicher Wahlkreiskandidaturen so viele unterschiedliche Stimmzettel wie es Wahlkreise gibt, doch grundsätzlich ist die Reihenfolge der Wahlvorschläge die gleiche. In diesem Fall kann also von einem einheitlichen Stimmzettel gesprochen werden, da sich jeweils nur Namen ändern und einzelne Einträge übersprungen sein können.
Bei Bezirksvertretungswahlen in kreisfreien Städten unterscheiden sich die Stimmzettel hingegen von Stadtbezirk zu Stadtbezirk auch in der Reihenfolge und dem grundsätzlichen Vorhandensein von Wahlvorschlägen, da es sich dabei eigentlich um Wahlen zu separaten Gremien handelt. Werden die Bezirksvertretungswahlen in Datensätzen als eine stadtweite

Wahl abgebildet, kann von unterschiedlichen Stimmzetteln und Stimmzettelgebieten gesprochen werden.
Grundsätzlich ist anzumerken, dass sich die Reihenfolge von Wahlvorschlägen auf dem Stimmzettel üblicherweise nach der Stimmenzahl richtet, die die politische Gruppierung oder die Einzelbewerber∗in bei der letzten Wahl erreichten, alle weiteren Wahlvorschläge werden in alphabetischer Reihenfolge angehängt [11].

Die Anzahl von Stimmen ist je nach Wahlart unterschiedlich: Aus überregionalen Wahlen wie Bundestagswahlen oder (nicht allen) Landtagswahlen ist das System der Kombination aus Personenwahl in Wahlkreisen und der Verhältniswahl von (Landes-)Listen bekannt, was über die Abgabe von zwei Stimmen umgesetzt wird [10].
Bei Wahlen im Rahmen der Kommunalwahlen in NRW haben die Wählenden immer nur eine Stimme, die sie für einen Wahlvorschlag abgeben können [11]. Dies gilt auch für Kreistags-/Ratswahlen, obwohl dabei sowohl Personen in Wahlkreisen als auch gebietsübergreifende Listen gewählt werden — im Gegensatz zu den zuvor erwähnten Wahlarten erfolgt dies hier aber über ein und dieselbe Stimme[1]. Im deutschlandweiten Vergleich von Kommunalwahlsystemen ist das nicht als übliche Methode einzuordnen. In den meisten anderen Bundesländern handelt es sich um Wahlen mit freien Listen, bei denen Wählende eine höhere Anzahl von Stimmen als 1 haben [12, 13]. Sie können mehrere Stimmen für eine kandidierende Person abgeben (Kumulieren), und auch die Stimmabgabe für Kandidierende unterschiedlicher Wahllisten ist möglich (Panaschieren). Diese Stimmen haben Auswirkungen auf die Sitzzuteilung innerhalb der jeweiligen Liste. Die Summe der einzelnen Kandidierenden-Stimmen je Liste ist relevant für die Bestimmung des Verhältnisses für die zugrundeliegende Verhältniswahl. Diese Summe muss beim Umgang mit Wahlergebnisdaten gegebenenfalls erst aus den einzelnen Kandidierenden-Stimmen ermittelt werden.

2.2 Informationsquellen

Auf höheren Ebenen wie beim Landes- oder Bundeswahlleiter sind zusammengefasste Ergebnisse zu finden, bereits am Wahlabend gibt es Darstellungen, die sich aus den vorläufigen Endergebnissen aus den Gemeinden bilden.
Die Informationen in diesen Quellen sind aber nicht dazu geeignet, Ergebnisse auf lokaler Ebene zu analysieren. Dafür muss auf Informationen aus den Gemeinden zurückgegriffen werden, die ein eigenes oder in ihrem Namen bereitgestelltes Wahlergebnisportal haben, über das Ergebnisse feiner aufgelöst abrufbar sind.

In Nordrhein-Westfalen hat die Software votemanager von der vote iT GmbH einen Anteil von 88 % [14] und wird Stand Mai 2020 deutschlandweit bei über 2.200 Behörden eingesetzt[2] [16]. Die öffentliche Wahlergebnisportalfunktionalität ist dabei nur ein Teil des Umfangs, die Software wird auch im Vorfeld des Wahlgeschehens und für die Einarbeitung von Schnellmeldungen am Wahlabend durch die Kommunalverwaltungen verwendet.
Zu weiteren Softwareprodukten im Bereich Wahlen zählen voteplus von der WRS Software

[1] Das führt auch dazu, dass eine politische Gruppierung nur in den Wahlbezirken Stimmen bekommen kann, in denen eine für sie kandidierende Person aufgestellt ist.

[2] In Bayern wird votemanager exklusiv unter dem Namen OK.VOTE vertrieben [15].

GmbH und IVU.elect (nunmehr elect iT) von der IVU.elect GmbH, die zum Verkehrs-IT-Dienstleister IVU Traffic Technologies AG gehörte. Im Jahr 2020 hat die vote iT GmbH die Gesellschaftsanteile beider zuvor genannten GmbH erworben [17, 16]. Auch zu erwähnen ist PC-Wahl, dessen Support zum Ende des Jahres 2020 eingestellt wird und dessen Nachfolgeprodukt votemanager ist [18].
Insgesamt hat die vote iT GmbH also spätestens seit 2020 Kontrolle über einen erheblichen Anteil von in Deutschland eingesetzter Wahlsoftware. In diesem Zusammenhang ist zu erwähnen, dass es sich bei den Gesellschaftern der vote iT satzungsgemäß ausschließlich um Institutionen handeln kann, die eine kommunale Trägerstruktur aufweisen [19].

Zu den Funktionalitäten solcher Wahlergebnisportale gehört üblicherweise mindestens die Ergebnisdarstellung in Grafiken und Tabellen — als Gesamtergebnis sowie in Gebieten, in die die Gemeinde unterteilt ist. In manchen Fällen kann ein Vergleich zu vorherigen Wahlen in Form einer Gewinn- und Verlustrechnung und Diagrammen dargestellt werden. Wenn das (vorläufige) Endergebnis vorliegt, kann auch eine Sitzverteilung dargestellt werden.

Geographische Wahlergebnisdarstellungen kann es in Wahlergebnisportalen auch geben. Die „GeoGrafik“-Funktion von votemanager wird in Zusammenhang mit Geodaten(-quellen) in Abschnitt 5.1 behandelt.

Maschinenlesbare Datenexporte können auch zum Umfang eines Wahlergebnisportals gehören. Besonders häufig werden Daten in Form von csv-Dateien bereitgestellt, die keinem an einer bestimmten Stelle generell dokumentierten Format unterliegen, sondern entweder selbsterklärend sein sollen (beispielsweise auch unter Einsatz von Feldnamen, die Name und Partei enthalten) oder dort, wo sie erhältlich sind, dokumentiert sind. Dazu gehören auch die Daten, die bereits in votemanager-Oberflächen unter „OpenData-Info“ (ehemals „Medienvertreter-Info“) bereitgestellt werden. Sie sind nicht ohne weitere, manuelle Datenzuordnung verwendbar, allein schon weil Feldbezeichnungen auf die Stimmzettelzuordnungen verweisen, die aber nicht maschinenlesbar verfügbar sind (Beispiel vgl. [20]).

2.3 Ergebnisse und Gebiete

An dieser Stelle werden insbesondere Gegebenheiten erläutert, auf die im Abschnitt Arten von Wahlen noch nicht vertiefend eingegangen wurde.
Im Kapitel Wahldaten und dem dortigen Abschnitt Praxis werden die Grundlagen unter anderem zum Thema Gebiete in Zusammenhang mit Datenmodellierung weiter vertieft.

In Zusammenhang mit Wahlergebnissen werden üblicherweise verschiedene Werte bereitgestellt, die über die einfachen Stimmenzahlen hinaus gehen. Dazu gehören zunächst Angaben zum Wählendenverzeichnis, mit der Anzahl der Wahlberechtigten. Auch die Anzahl von Wahlberechtigten mit Wahlschein kann angegeben werden. Wahlscheine werden ausgestellt, wenn die Briefwahl beantragt wird. Sie können auch dazu verwendet werden, in anderen Wahllokalen als dem eigentlich vorgesehenen zu wählen, solange die dort verwendeten Stimmzettel übereinstimmen, also im Falle kommunaler Wahlen der Wahlbezirk oder Stadtbezirk übereinstimmt [21].
Über diese Werte kann der Anteil an Briefwählenden abgeschätzt werden.

Mit Eingang der Schnellmeldung am Wahlabend wird die Zahl der Wählenden im Wahllokal bereitgestellt, dazu ggf. auch die Anzahl von Wählenden mit Wahlschein. Für die abgegebenen Stimmen werden dann Angaben der ungültigen und gültigen Stimmen bereitgestellt. Die Summe dieser Angaben soll der Anzahl an Wählenden entsprechen. Dann folgen Angaben der Stimmen je nach Wahlvorschlag, auf den sie entfielen. Die Summe dieser Stimmen muss der Anzahl der gültigen Stimmen entsprechen. Sollen die Ergebnisse je Wahlvorschlag proportional dargestellt werden, wird die Stimmenanzahl für den Wahlvorschlag durch die Anzahl der gültigen Stimmen geteilt.

Die Ergebnisse anhand werden üblicherweise je Wahllokal anhand der Schnellmeldungen eingepflegt. Dabei ist der Begriff Wahllokal in diesem Zusammenhang nicht als physisches Gebäude oder Raum zu verstehen, da es auch mehrere „Wahllokale“ beispielsweise nebeneinander in einer Halle geben kann, jeweils zuständig für die Wahl in einem bestimmten Bezirk.
Dabei können Begriffe leicht verwechselt werden, allein schon weil diese Bezirke bei verschiedenen Wahlen unterschiedlich benannt sein können. Während beispielsweise bei überkommunalen Wahlen von Wahlbezirk gesprochen wird, hat der Begriff Wahlbezirk bei kommunalen Wahlen eine eigene wahlrechtliche Bedeutung und die kleinste Einheit wird Stimmbezirk genannt [11].
Um eindeutigere Begriffe zu verwenden, wird für die Gebiete, in denen „die Wahl stattfindet“, folgend von Gebieten niedrigster Ebene oder Bezirken niedrigster Ebene gesprochen.

Gebiete höherer Ebenen lassen sich als Aggregationen von Gebieten niedrigster Ebene beschreiben. Zu solchen Gebieten können beispielsweise die Wahlbezirke bei kommunalen Wahlen zählen, sowie Stadtbezirke, Stadtteile, Gemeinden und letztendlich beliebige andere Arten von Gebieten.
Besonders hervorzuheben ist der Umgang mit Briefwahlbezirken. Die Briefwahlauszählung kann nicht nur gemeinsam mit der Auszählung in Wahllokalen stattfinden, sondern auch in separaten Briefwahlbezirken, die so wie andere Bezirke niedrigster Ebene behandelt werden. Der Umfang dieser Briefwahlbezirke kann sich unterscheiden: es kann einen Briefwahlbezirk für die gesamte Gemeinde geben, oder sie entsprechen Gebieten auf einer bestimmten Ebene, beispielsweise Wahlbezirken bei einer kommunalen Wahl (was aufgrund der unterschiedlicher Stimmzettelinhalte notwendig werden kann). In Zusammenhang mit dem Datenmodell wird dieses Thema im Abschnitt 4.3.5 vertieft.
Es ist darauf hinzuweisen, dass die Wahlbeteiligung aufgrund von Briefwahlen über 100 % liegen kann. Dies geschieht beispielsweise, wenn eine Gemeinde Briefwahlzettel einer anderen Gemeinde mit auszählt, oder wenn eine Gemeinde ihre Briefwahlbezirke zusammengefasst hat. Es handelt sich dabei um Wahlen, bei denen die Gebietsunterschiede nicht zwingend ausschlaggebend sind, also beispielsweise eine Landtagswahl in mehreren Nachbargemeinden, oder eine Stichwahl, für die eine Gemeinde die sonst aufgrund der Ratswahl vielen Briefwahlbezirke zusammengefasst hat[3]. Auf aggregierenden Ebenen kann es also

[3] Als Beispiel kann Weilerwist genannt werden: Bei der Landratsstichwahl gibt es nur noch 5, nicht mehr die zuvor erforderlichen 15 Briefwahlbezirke: https://wahlen.kdvz-frechen.de/kdvz/kw2020/05366040/html5/Landratstichwahl_NRW_123_Uebersicht_stbz.html

zu einer Wahlbeteiligung von über 100 % kommen, sowie zu falschen geographischen Aussagen, während es im Gesamtergebnis wiederum ohne so eine Erscheinung wiedergegeben wird.
Briefwahlstimmen sind auch ohne bezirkliche Besonderheiten möglicherweise mit inhaltlichen Unterschieden verbunden — auf Gebietsebenen, bei denen keine Briefwahlstimmen berücksichtigt werden können, kann es sinnvoll sein, darauf hinzuweisen, dass nur Urnenwahlergebnisse dargestellt sind, ebenso wie ein Hinweis auf die Zusammenfassung von Briefwahlauszählungen angemessen erscheint.

In Zusammenhang mit Ergebnissen sind nach Vorliegen des (vorläufigen) Endergebnisses auch die Sitzverteilungen und Mandate zu thematisieren. Abhängig vom örtlichen Wahlrecht und der Vollständigkeit der vorliegenden Daten können Sitzverteilungen berechnet werden, sowie die dazugehörigen gewählten Personen anhand von Kandidaturgebieten und Listenplätzen. Im Rahmen dieser Bachelorarbeit wird dies nicht weiter vertieft, da diese Darstellungen bei einer geographischen Wahlergebnisdarstellung nicht zwingend notwendig sind.

3 Open Data

Open Data bezeichnet die wiederverwendbare Bereitstellung von nicht personenbezogenen Daten, in erster Linie durch öffentliche Stellen, die diese gesammelt, erstellt oder bezahlt haben, und diese der Allgemeinheit kostenlos zur Verfügung stellen [22].
Folgend werden Grundlagen von Open Data im Allgemeinen erläutert. In den Kapiteln 4 und 5 wird das Thema in Zusammenhang mit den für die zu entwickelnde Anwendung verwendeten Daten praktisch behandelt.

Die Bereitstellung offener Daten ist ein wesentliches Element der ersten Phase, Transparenz, des Open Government (offenes Regierungs-/Verwaltungshandeln) [22]. Partizipation und Kollaboration sind weitere Phasen des Open Government, was als „umfassende Neugestaltung von Politik- und Verwaltungshandeln im Sinne eines modernen Public Managements bzw. von Public Governance“ beschrieben werden kann [22, 23].

Forderungen nach Open Data können auf verschiedenen, separat zu betrachtenden Argumentationslinien beruhen [22, 24]. Zu den technischen Argumenten gehören die Nutzbarmachung von Daten in maschinenlesbarer Form und die Verbesserung der Auffindbarkeit und Nutzbarkeit durch die Beschreibung von Daten mit Metadaten. Als wirtschaftliches Argument wird der volkswirtschaftliche Mehrwert angeführt, sowie das Potenzial für lokale Wirtschaftsförderung, und die Kosten im Falle der sonst anfallenden Vermarktung, Verkauf, und Verrechnung. Gesellschaftlich kann argumentiert werden, dass die Erstellung der Daten bereits von der Öffentlichkeit finanziert ist und diese daher frei zur Verfügung stehen sollten. Außerdem sind organisatorische Argumente und Effekte für Verwaltungen nicht zu unterschätzen: Daten können auch von anderen Stellen innerhalb der Verwaltung genutzt werden, Datenfehler können gemeldet werden, und die Anzahl an Anfragen/Anträgen nach Informationen, die sie erhalten, kann sinken [22, 25].

Dass Open Data auch einen Paradigmenwechsel in der Frage, wie der Zugang zu Daten des öffentlichen Sektors geregelt ist, bedeutet, wird auch aus dem Open Data Handbuch des Kompetenzzentrums Open Data bei dem Bundesverwaltungsamt deutlich: Während bisher galt, dass sinngemäß alles nicht öffentlich ist, was nicht ausdrücklich als öffentlich gekennzeichnet ist, gilt im neuen Umgang mit Daten, dass sie grundlegend öffentlich sind. Es soll zu proaktiver, vollumfänglicher, und zeitnaher Veröffentlichung kommen. Es gibt ausdrückliche Rechte zur Nutzung, Weiterverarbeitung, und Weiterverbreitung von Daten für jegliche Zwecke. Dass die Daten sich im „Besitz“ der Behörde befinden, wird auch als zum alten Paradigma gehörend aufgeführt. Außerdem erschließt sich aus den Daten neuer Nutzen, der über den hinaus geht, wegen dem die Daten in der Behörde erhoben wurden. Abschließend kann gesagt werden, dass Open Data auch mit einer Veränderung von Abläufen einhergehen muss [26].

3.1 Schritte der Datenöffnung

In dem von der Bertelsmann Stiftung publizierten Leitfaden für offene Daten [22] werden Grundlagen der Öffnung von Daten in drei wesentliche Schritte, unterschieden nach rechtlichen, technischen, und organisatorischen Aspekten, unterteilt.

Rechtlich geht es um die Wahl der Datenlizenz. Die Verwendbarkeit soll explizit gekennzeichnet werden. Im Wesentlichen sind die Varianten gemeinfrei, „Zero“ (keine Bedingungen), oder Namensnennung anzuwenden, wobei der Grundsatz „So offen wie möglich!“ gelten solle. Weltweit bekannt sind dabei die zivilgesellschaftlich bevorzugten Creative-Commons-Lizenzen, die auch in internationalisierter Form vorliegen. Deutsche Behörden empfehlen [27] allerdings die Verwendung der Lizenzvarianten der aus verschiedenen Gründen umstrittenen Datenlizenz Deutschland 2.0. Dazu gehört, dass der Anschein erweckt wird, dass sie gegenüber anderen Lizenzen besser zum deutschen Rechtsraum passe, obwohl die aktuellen Versionen der Creative-Commons-Lizenzen durchaus auf den europäischen Rechtsraum angepasst sind [28, 29]. Der kleine textliche Umfang der Datenlizenz Deutschland lässt Fragen offen, die bei Creative-Commons-Lizenzen in dessen Langfassungen explizit aufgegriffen werden. Die Nachnutzung soll nämlich explizit dann zu ermöglichen sein, wenn es sich um ein urheberrechtlich geschütztes Werk handelt — bei der Definition der Datenlizenz Deutschland wird dies nicht aufgegriffen, und es kommt erkennbar dazu, dass Behörden solche Datenlizenzen dann anwenden, wenn ein urheberrechtlicher Schutz nicht vorliegt [30]. Vielmehr ist im Falle zu veröffentlichender, nicht urheberrechtlich schützbarer Faktendaten eine Kennzeichnung der Gemeinfreiheit anzubringen, nicht etwa die Nutzung einer „Zero“- oder einer einschränkenderen Lizenz [30, 22].

Bei dem zweiten Schritt, dem technischen Aspekt, geht es um die Verwendung maschinenlesbarer Dateiformate. Probleme bei der Verwendung von anderen, nicht strukturierten oder maschinenlesbaren Datenquellen werden in Zusammenhang mit alternativen Methoden der Datenermittlung später in diesem Abschnitt thematisiert. In Zusammenhang mit dynamischen Daten werden Anwendungsprogrammierschnittstellen (API) als Best Practice aufgeführt.

Der dritte Schritt behandelt im organisatorischen Aspekt Metadaten, die Datensätze zur verbesserten Auffindbarkeit und Verständlichkeit beschreiben und üblicherweise für die Kernfunktionalität von Open-Data-Portalen unerlässlich sind. Der Standard DCAT-AP.de[1] wird für solche Metadaten verwendet, dies ermöglicht auch die Weiterreichung an andere Datenportale („harvesting“) ohne redundante manuelle Einpflegung von Datensätzen in mehreren Portalen.

[1] Definitionen für DCAT-AP.de: https://www.dcat-ap.de/def/

3.2 Entwicklung und Rechtsgrundlagen

In dem Gutachten „Open Data in Deutschland und Europa — Vorschlag zur Weiterentwicklung des rechtlichen Rahmens einer Informationsordnung“, publiziert von der Konrad-Adenauer-Stiftung, werden die Eckpunkte der Entwicklung von Open Data und der aktuelle rechtliche Rahmen dargestellt [31]:

2016 hat sich die Bundesregierung der *Open Government Partnership* angeschlossen, ein Zusammenschluss von Staaten, mit dem Ziel, „Open Data und Open Government-Prinzipien auf nationaler und internationaler Ebene voranzubringen“. 2017 wurde der erste Nationale Aktionsplan mit Projekten und Verpflichtungen zur Förderung von Open Data verabschiedet. Seit 2019 gibt es den Zweiten Nationalen Aktionsplan, der nach einem öffentlichen Konsultationsprozess verabschiedet wurde und auch Verpflichtungen einiger Bundesländer, darunter Nordrhein-Westfalen, enthält [32].

Als gesetzgeberische Meilensteine auf nationaler/EU-Ebene werden im Gutachten unter anderem aufgezählt:

- 2006: Informationsfreiheitsgesetz (IFG), Informationsweiterverwendungsgesetz (IWG) auf Bundesebene
- 2007: Richtlinie 2007/2/EG des Europäischen Parlaments und des Rates vom 14. März 2007 zur Schaffung einer Geodateninfrastruktur in der Europäischen Gemeinschaft (INSPIRE)
- 2013: Directive of the European Parliament and of the Council of 26 June 2013 amending Directive 2003/98/EC on the re-use of public sector information 2013/37/EU
- 2017: Bundestag beschließt das sogenannte Open-Data-Gesetz (§ 12a EGovG)
  - nur auf die unmittelbare Bundesverwaltung anwendbar
  - Daten eingegrenzt durch Strukturiertheit, Beschränkung auf Tatsachen, Beschränkung auf unbearbeitete Rohdaten
  - keine Pflicht zur Digitalisierung analoger Datenbestände
  - Proaktive Pflicht zur „unverzüglichen“ Datenbereitstellung unter Beachtung von Ausnahmen
- 2019: Richtlinie 2019/1024 vom 20. Juni 2019 über offene Daten und die Weiterverwendung von Informationen des öffentlichen Sektors, ABl L 172/56 v. 26.6.2019.
  - Dokumente in Übereinstimmung mit § 12a EGovG künftig EU-weit grundsätzlich „open by design“ und „open by default“; auch Forschungsdaten mit einbezogen
  - Besondere Behandlung von „hochwertigen Datensätzen“ — Bereitstellung soll über Programmierschnittstellen erfolgen (als Kategorien aufgelistet werden „Geodaten, Erdbeobachtung und Umwelt, meteorologische Daten, Statistiken, Unternehmensdaten und Mobilität“)
  - Vorschriften über die Weiterverwendung werden über den öffentlichen Sektor hinaus ausgedehnt auf öffentliche Unternehmen

– Die Richtlinie ist bis zum 17. Juni 2021 in nationales Recht umzusetzen

Die folgende Abbildung 3.1 vom Branchenverband Bitkom stellt die Open-Data-Landschaft in Deutschland dar. Auf die Situation in Nordrhein-Westfalen wird im folgenden Abschnitt 3.3 eingegangen.

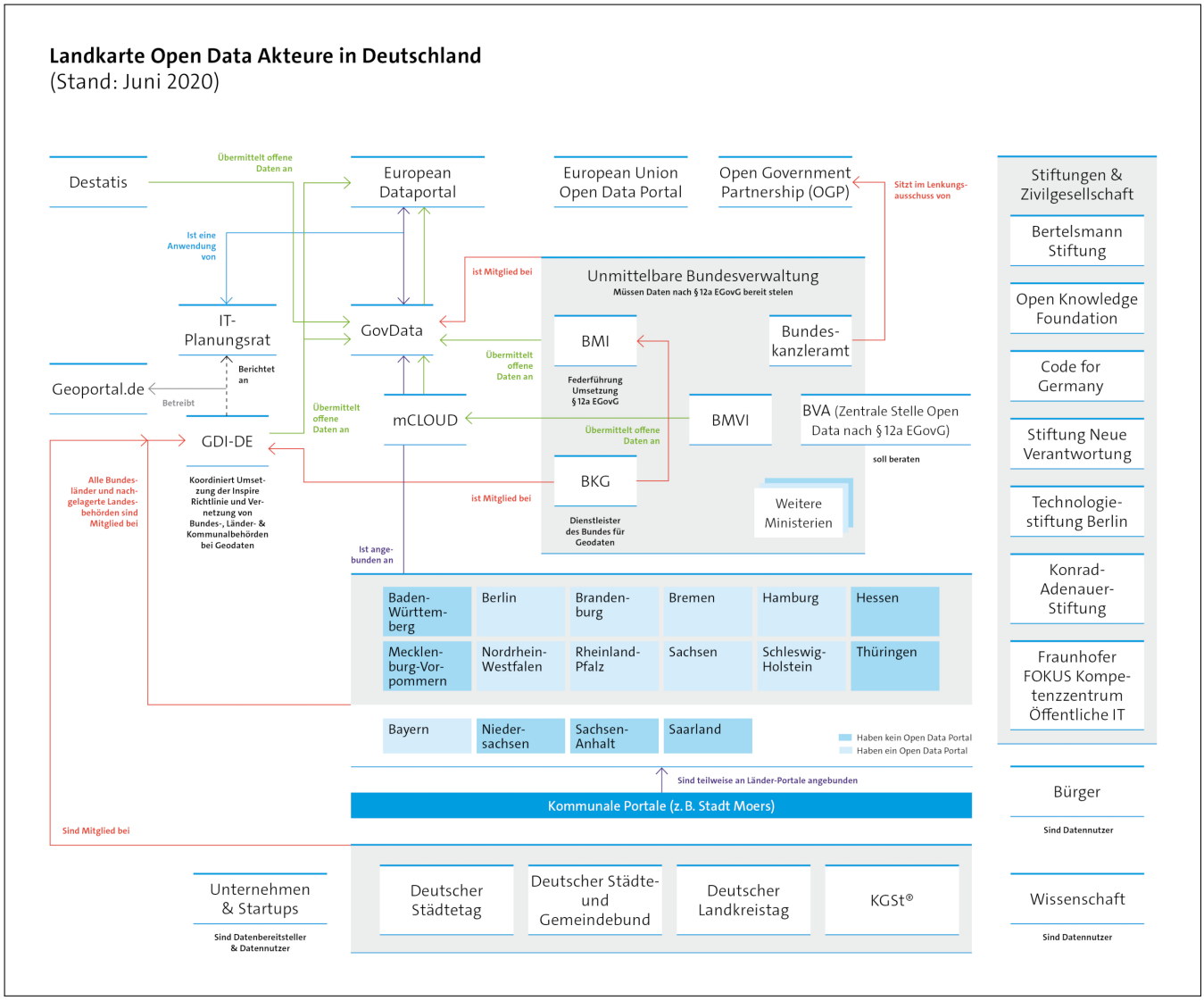

ABBILDUNG 3.1: Landkarte Open Data Akteure in Deutschland [33]

3.3 Open Data in Nordrhein-Westfalen

Als wegweisende Open-Government-Strategie wurde 2014 die „Open.NRW-Strategie“[2] beschlossen. Dabei sollte nicht nur der Aspekt Transparenz des Open Government realisiert werden [34].

2016 wurde in NRW eine Rahmenvereinbarung über die Zusammenarbeit zwischen Land und Kommunen bei der „Verankerung von Open Government in den Verwaltungen“ geschaffen. Die Einigung erfolgte unter Beteiligung diverser kommunaler Verbände. Die Umsetzung wird seither durch einen *Arbeitskreis Open Government* realisiert und durch die Geschäftsstelle Open.NRW koordiniert. Aufgrund des erforderlichen „erheblichen Kulturwandels“ verfolgt der OGP eine Strategie der „unterschiedlichen Geschwindigkeiten“, bei der die Verwaltungseinheiten vor Ort über Art und Umfang jeglicher Initiativen entscheiden — wobei von einem Mehrwert für alle Teilnehmende durch Bündelung und Austausch ausgegangen wird [35].
Im Rahmen des Pilotprojekts Kommunales Open Government wurden 2017 Projekte gefördert, die die praktische Umsetzung des Open Government in Kommunen zeigen sollten. Ein Leitfaden stellt die Projekte, Erkenntnisse, und Empfehlungen für kommunales Open Government vor [36].

Die aktuelle Rechtslage im Land NRW ergibt sich aus § 16 und § 16a EGovG NRW [37]. Unmittelbar betroffen sind die Behörden des Landes, während in Zusammenhang mit Kommunen eine Kann-Formulierung enthalten ist. Es werden ähnliche Definitionen, Anforderungen, und Verpflichtungen in Bezug auf Daten und Datenveröffentlichung angegeben wie im zuvor erwähnten „Open-Data-Gesetz“ des Bundes.

Die Kernverantwortlichkeit der Geschäftsstelle Open.NRW beim Ministerium für Wirtschaft, Innovation, Digitalisierung und Energie (MWIDE) liegt in Entwicklung und Betrieb des Open-Data-Portals Open.NRW. Es dient als zentraler Knoten für offene Daten in NRW, inklusive Berücksichtigung kommunaler Daten und Weiterreichung von Metadaten an das Bundesportal GovData [34].
Als besondere Entwicklung ist zu erwähnen, dass eine unmittelbare Einpflegung offener Datensätze durch Kommunen im Open.NRW-Portal seit 2020 möglich ist, so dass auch Kommunen ohne eigenes Open-Data-Portal ihre Daten im Open.NRW-Portal berücksichtigen lassen können [38].

In Zusammenhang mit Geodaten ist die Vorreiterrolle des Landes Nordrhein-Westfalen hervorzuheben. Seit 2017 werden Geobasisdaten unter einer freien Lizenz bereitgestellt. Dies erfolgt auf Landesebene über diverse Zugangsmöglichkeiten. Die Datenpflege liegt bei diesen Daten in vielen Fällen, insbesondere bei Katasterdaten, auf einer kommunalen Ebene. Zuvor erfolgte die Bereitstellung von Geodatensätzen mit einer komplizierten Gebührenordnung und einem Online-Shop, der den Open-Data-Prinzipien nicht gerecht werden konnte und abgeschaltet wurde [39].
Während die Open-Data-Lizenz zunächst die Datenlizenz Deutschland – Namensnennung

[2] https://open.nrw/system/files/media/document/file/opennrwt1web.pdf

Version 2.0 war, werden sämtliche Geobasisdaten seit März 2020 in der „Zero“-Variante der Lizenz einschränkungs- und bedingungsfrei bereitgestellt [40].

Neben den Einrichtungen auf der Landesebene ist ein wesentlicher regionaler Akteur der Regionalverband Ruhr (RVR). Mit dem Geonetzwerk Metropole Ruhr erfolgt bereits seit 2013 interkommunale Zusammenarbeit zur Bearbeitung von Aufgaben in Zusammenhang mit regionalen Geoinformationen [41]. Auch das Thema Open Data an sich wird vom RVR bearbeitet: 2019 startete das interkommunale Portal opendata.ruhr in eine Testphase, und in einem Leitfaden werden gemeinsame Grundsätze bezüglich Aufgaben, Geschäftsprozessen, technischen und rechtlichen Fragen definiert [42].

3.4 Lage in Kommunen

Im Oktober 2020 wurde eine Studie [43] zu einer Kommunalbefragung von über 200 Kommunen zur Bereitstellung offener Daten veröffentlicht. Sie wurde durch die Bertelsmann Stiftung und das Deutsche Institut für Urbanistik durchgeführt.
Aus der Befragung geht hervor, welche Chancen und Herausforderungen Kommunen bei der Bereitstellung offener Daten sehen. Eingangs wird bereits der bislang regional stark unterschiedliche Grad der Bereitstellung offener Daten in Kommunen erwähnt.
Bei der Frage, ob offene Daten eher mit Chancen oder Risiken verbunden werden, verbindet knapp die Hälfte persönlich eher Chancen mit dem Thema, wozu besonders Befragte aus Großstädten beitragen. In den Mittel- und Kleinstädten liegt der Wert bei rund 40 %. Geht es um die Bewertung durch die Kommunalverwaltungen als Ganzes werden mehrheitlich sowohl Chancen als aus Risiken gesehen — bei mehr als einem Viertel werden eher Risiken mit offenen Daten verbunden [43].

Bei der Frage, ob eine Kommune überhaupt offene Daten veröffentlicht, liegt der Wert bereits bereitstellender Kommunen bei insgesamt 35 %, wobei der Wert bei Großstädten bei 71 % liegt. Die Frage ist nicht darauf hinaus gestellt, ob die Kommune diese über ein Open-Data-Portal veröffentlicht, oder zumindest in einem gewissen (zahlenmäßigen oder thematischen) Umfang, wodurch vermutlich auch Kommunen mit sehr wenigen Datensätzen sagen konnten, dass sie bereits offene Daten bereitstellen. Insgesamt ist klar erkennbar, dass der Wert bei Klein- und Mittelstädten deutlich geringer ist. Von diesen sagen rund 40 bis 50 Prozent, dass sie keine Daten bereitstellen und keine Maßnahmen ergriffen haben. Eine Koordinierungsperson für Daten(-veröffentlichungen) wurde insgesamt erst in jeder sechsten Kommune benannt [43].

Dafür, dass sie keine offenen Daten bereitstellen, geben Kommunen unterschiedliche Gründe an — am häufigsten werden dabei mangelnde personelle Ressourcen mit insgesamt 81 % angeführt. 74 Prozent geben einen fehlenden gesetzlichen Auftrag als Grund an. Weitere Gründe, die insgesamt (trifft mindestens eher zu) mit knapp 60 % vorkommen, sind die Befürchtung von Datenmissbrauch, Datenschutzrechtliche Bedenken, Mehrkosten durch Bereitstellung/Aufbereitung, und fehlende Kompetenzen. Ein Grund, den insgesamt zwei Drittel der Kommunen als eher oder ganz zutreffend (davon die meisten, absolut 54 % als eher zutreffend) bezeichnen, ist der unklare Mehrwert für die Kommune [43].

3.5 Alternativen

Angesichts der bisherigen Situation der Datenveröffentlichung durch Kommunen kann sich häufig die Frage stellen, was alternativ getan werden kann, wenn die gewünschten Daten nicht als Open Data verfügbar sind. Methoden wie das Stellen einer Anfrage nach dem Informationsfreiheitsgesetz oder anderweitiges Kontaktieren lokaler Behörden können insbesondere kurzfristig niedrige Erfolgschancen haben [44]. Das kann sich möglicherweise dadurch erklären lassen, dass die geforderten Daten oftmals nicht veröffentlichungsfähig vorliegen, beispielsweise weil eine Kommune ihre wahlbezogenen Flächendaten zuerst von Grund auf digitalisieren müsste.
Solche strukturellen Probleme und die Nichtbereitstellung von Daten sorgen dafür, dass (potenzielle) Datennutzende auf alternative Wege der Datenermittlung zurückgreifen müssen, um ihre Ziele beispielsweise für ein Projekt zu erreichen. Darauf wird auch im Rahmen dieser Bachelorarbeit in Zusammenhang mit Wahldaten in Abschnitt 4.4, und in Zusammenhang mit Geodaten in Abschnitt 5.2 eingegangen.
Eine solche alternative Datenermittlung hat negative Auswirkungen hinsichtlich des Arbeits- und Zeitaufwands und auf die Datenqualität, da Fehler auftreten können [45]. Wenn Kommunen bewusst unter Aufführung des Risikos der möglichen Falschdarstellungen und Datenmissbrauchs [43] auf die Bereitstellung offener Daten verzichten möchten [46], kann genau das dafür sorgen, dass bei Projekten schließlich Daten einer niedrigeren Qualität dargestellt werden, als es sonst möglich wäre. Dies geschieht dann nicht aus Gründen der vermuteten Böswilligkeit oder wegen mangelnder Kompetenz der Datennutzenden, sondern weil die Datennutzenden die bessere Datengrundlage nicht zur Verfügung haben, und stattdessen auf Methoden wie manuelle Datenerfassung oder eine automatisierte Form der Datenermittlung (wie Scraping) setzen müssen.
Darüber hinaus besteht dann auch eine unsichere Situation in Zusammenhang mit der Lizenz der verwendeten Inhalte. Häufig sind bei behördlichen Webauftritten oder Publikationen keine Lizenzangaben vorhanden, oft sind im Impressum explizit pauschale Einschränkungen der Nutzbarkeit sämtlicher Inhalte enthalten [22]. So kann es sogar dazu kommen, dass Datennutzenden mit rechtlichen Schritten gedroht wird [47, Anlagen].

4 Wahldaten

Als Wahldaten werden folgend die Daten bezeichnet, die sich unmittelbar auf die Struktur und den Ausgang der Wahlen beziehen, darunter die Gebietseinteilung, die kandidierenden Gruppen und Personen, und Ergebnisse.

Ergänzend zu den Darstellungen in der Einführung und im vorherigen Kapitel Open Data kann in Zusammenhang mit Wahldaten auf den Blogbeitrag „Warum wir über Wahldaten reden müssen“ bei dem zivilgesellschaftlichen Netzwerk Code for Germany verwiesen werden. In diesem werden insbesondere im Kontext der damaligen Bundestagswahl 2017 Erfahrungen und Wünsche bezüglich Datenformat und -qualität dargestellt und auf die Wichtigkeit der Entwicklung eines einheitlichen Standards für zukünftige offene Bereitstellung von Wahldaten hingewiesen [44].
Auf der Website des zivilgesellschaftlichen Projekts Offene Wahlen Österreich wurden 2017 konkrete Forderungen zu Wahldaten formuliert. Darunter Open-Data-Grundsätze wie Offenheit, Maschinenlesbarkeit und Zeitnähe im Allgemeinen, sowie die Vollständigkeit auch hinsichtlich des Wahlprozesses, und eine möglichst hohe Granularität. Die Daten sollten möglichst konstant und langfristig erhalten bleiben, und Daten vergangener Wahlen sollen vergleichbar aufbereitet sein [48].

Wahldaten für die zu entwickelnde Anwendung werden Daten im Format des *Standards für Offene Wahldaten* (OWD) sein. Das Format wird im folgenden Unterkapitel näher beschrieben.

Quelle für diese Daten sind Beispieldatensätze von der Website offenewahldaten.de oder selber erstellte Datensätze. Die Erstellung neuer Datensätze wird im Unterkapitel Datenerstellung beschrieben.
Perspektivisch ist davon auszugehen, dass automatisiert erstellte Daten gemäß des Standards für Offene Wahldaten in den Oberflächen von votemanager-Instanzen bereitgestellt werden.
Es war auch aufzugreifen, dass perspektivisch über eine Web-Programmierschnittstelle/API nachgedacht wird, allerdings ohne weitere Details dazu.
Ebenfalls wird das Thema der Wahldaten und ihrer Bereitstellung im Rahmen eines landesweiten Projektes zur Umsetzung des Onlinezugangsgesetzes (OZG) angegangen, in dessen Kontext auch das Projekt „Offene Wahldaten“ erwähnt wird [49].

4.1 Offener Wahldatenstandard

Die Entwicklung des Offenen Wahldatenstandards wurde ins Leben gerufen, weil es keinen einheitlichen Standard für Wahldaten gibt, und unterschiedliche Daten, die im Umfeld einer Wahl anfallen, bislang an unterschiedlichen Stellen bereitgestellt werden [50]. Ziel sei es, beim Standard „einfach strukturierte Daten nicht durch komplexe Strukturen“ zu ersetzen, sondern zu ergänzen, und somit einen Standard zu schaffen, „der für jeden zugänglich ist, seien es affine Menschen oder Pressevertreter“ [9].

Die Entwicklung begann 2018 durch das kommunale Rechenzentrum kdvz Rhein-Erft-Rur mit den Projektpartnern Open Knowledge Foundation, vote iT, mehreren Kommunen, und weiteren Vertreter∗innen der Open-Data-Community [50]. Seitdem gab es mehrere Treffen einer Projektgruppe, um Aspekte des Datenformats und weitere Planungen zu diskutieren[9, 51].

Im Januar 2020 ging die Website offenewahldaten.de online, auf dieser werden seitdem die Ideen sowie Datenformatsbeschreibungen und Beispieldatensätze bereitgestellt.
Im November 2020, während des Bearbeitungszeitraums dieser Bachelorarbeit, wurden erstmals Datensätze mit aktuellen Realdaten bereitgestellt und geringfügige Aktualisierungen von Datenformaten durchgeführt.
Eine ergänzende Grundlage ist das Konzeptpapier [52], welches bereits länger vor der Veröffentlichung der Website als Informationssammlung diente und ergänzende Darstellungen von Überlegungen enthält.

4.2 Datenmodell

Als Dateiformat wird grundsätzlich csv mit Trennzeichen Semikolon verwendet. Als Gründe dafür werden die höhere Kompatibilität und Verbreitung angegeben. Zu den weiteren Vereinbarungen gehört die Verwendung des Encodings UTF-8, einer Headerzeile, Quoting von Datenfeldern mit bestimmten Sonderzeichen, Verwendung des deutschen Datums-/Zeitformats, und dass beim Einlesen die Spaltennamen anstelle der Spaltenabfolge verwendet werden sollen [53].

Für jede Wahl ist je ein separater Satz an Dateien zu laden. Eine Zusammenfassung mehrerer Wahlen eines Wahltermins erfolgt nicht.
Für die Dateinamen gibt es auch eine feste Vorschrift (Darstellung direkt [53] entnommen):
Gemeindeschlüssel der Behörde _ Wahltag (YYYYMMDD) _ Wahlname _ Datensatzbeschreibung _ Version _ Zeitstempel (YYYYMMDD T HHMMSS) . Dateiendung
Bsp.: 05362028_20140525_Buergermeisterwahl-NRW_Wahlparameter_V0-1_20140415T132552.csv

Jede Datei enthält mindestens vier Spalten: Die verwendete Version des offenen Wahldatenstandards (`version`), ein Gemeindeschlüssel der Wahlbehörde (`wahl-behoerde-gs`), der Wahltag (`wahl-datum`), und der behördenübergreifend einheitliche Wahlname (`wahl-name`) [53].

Folgend werden die Dateiformate einzeln beschrieben, anhand der Dokumentation, die den jeweils über die Übersichtsseite [54] erreichbaren Detail- und Versionsseiten zu entnehmen ist.

In **„Wahlparameter“-Dateien** wird ein ausführlicher Wahlname (`wahl-bezeichnung`), die Bezeichnung der Ebene von Kandidaturen (`kandidat-gebiet-bezeichnung`), und für bis zu 5 Wahlgebietsebenen die Bezeichnung (`gebiet-ebene-X-bezeichnung`, Ebene 1 mit `bezirk-bezeichnung`) angegeben. Da unterschiedliche Wahlen separat voneinander abgebildet werden, enthält die Wahlparameter-Datei nur einen Datensatz.

Die Definition der Wahlgebiete erfolgt in **„Wahlgebietseinteilungen“-Dateien**, wo eine zusätzliche Unterscheidung zwischen der Behörde, in der die Wahl stattfindet, und der für

die Wahl zuständigen/wahlleitenden Behörde erfolgt. Dies ist bei Kreistagswahlen relevant, wenn Ergebnisse nicht nur zusammengefasst auf Kreiswahlbezirksebene, sondern bis zum Wahllokal in jeder einzelnen Gemeinde herunter angegeben werden.
Der Schlüsselwert — die Bezirksnummer (`bezirk-nr`), sowie weitere Angaben wie Name, Art, und Repräsentativität sind für jeden Datensatz, also für jedes Wahlgebiet, enthalten.

In weiteren Feldern ist für die Gebiete auf höherer Ebene jeweils eine Nummer und Bezeichnung angegeben. Dies ist die einzige Möglichkeit, zu ermitteln, welche Gebiete es überhaupt auf einer höheren Ebene gibt. Sie sind also nicht explizit abgebildet und müssen anhand ihrer mehrfachen Vorkommnisse als Aggregationen mehrerer Wahlgebiete niedrigster Ebene virtuell abgebildet werden.

Weitere Angaben jedes Wahlgebiets sind die Gebietsnummer und -bezeichnung für Kandidaturgebiete und Stimmzettelgebiete. Es erscheint so, als wären die dortigen Angaben den sonstigen Gebietsangaben zuzuordnen, da die Nummer und Bezeichnung in der Regel übereinstimmen, doch dies ist nicht so vorgesehen. Bei der Verarbeitung von Kandidatur- und Stimmzettelgebieten erfolgt die Verknüpfung nur mit (mehreren) Wahlgebieten niedrigster Ebene, sie kann formell nicht mit einem virtuellen Gebiet auf einer der übergeordneten Ebenen erfolgen.

In **„Stimmzettel“-Dateien** sind Vorkommnisse von Parteien (bzw. sonstigen Gruppierungen oder Einzelkandidierenden) auf Stimmzetteln enthalten. Ein vollständiges „Stimmzettel“-Objekt existiert nicht explizit und ergibt sich als Aggregation aller Datensätze in dieser Datei mit einer übereinstimmenden Angabe der Stimmzettelgebietsnummer.
Es kann auch nicht von explizit vorkommenden „Partei“-Objekten gesprochen werden, es handelt sich nicht um Parteiangaben, die für die jeweilige Wahl generell verwendet werden können. Gibt es mehrere Stimmzettelgebiete, kommen Parteien und ihre Angaben mehrfach vor, nur jeweils mit einer anderen Stimmzettelzuordnung und Position.
Bei den Parteiangaben handelt es sich um ein optionales Kürzel (`partei-kurzname`), den Namen (`partei-langname`), einen Hex-Farbwert (`partei-rgb-wert`), und Typ (`partei-typ`, dokumentiert sind P=Partei und E=Einzelbewerber).

Datensätze in **„Wahlergebnisse“-Dateien** sind je einem Wahlgebiet (also auf der niedrigsten Ebene) zugeordnet. Darüber hinaus ist nur ein Feld, der Zeitstempel der Erfassung/Anpassung (`zeitstempel-erfassung`) standardmäßig enthalten. Alle weiteren Felder mit numerischen Angaben zu Wahlberechtigten („A-Zahlen“, Angaben aus dem Wählendenverzeichnis), Wählenden, und abgegebenen Stimmen werden je nach Wahlart unterschiedlich dokumentiert, da es unterschiedliche Bezeichnungen oder Bedeutungen der Feldbezeichnungen geben kann. Die bisher verfügbare Dokumentation zu den NRW-Kommunalwahlen ist in Tabelle 4.1 dargestellt.

[1] Hier wird in der Quelle von Erststimmen gesprochen, im Kontext von Kommunalwahlen ist dieser Begriff aber nicht zutreffend. Gibt es eine Unterscheidung von Erst- und Zweitstimme wie beispielsweise bei Bundestagswahlen, kommen voraussichtlich zusätzlich die Felder E und Fn für Zweitstimmen zum Einsatz [53].

[2] https://offenewahldaten.de/wp-content/uploads/2020/11/Datensatzbeschreibung_Wahlergebnisse_V0-3.pdf

A1	Wahlberechtigte ohne Wahlschein
A2	Wahlberechtigte mit Wahlschein
A3	Wahlberechtigte, die nicht im Wählerverzeichnis eingetragen sind
A	Wahlberechtigte insgesamt
B	Wählende
B2	Wählende mit Wahlschein
C	Ungültige Stimmen[1]
D	Gültige Stimmen
D1	Stimmen für Pos. 1 laut Stimmzettel
Dn	Stimmen für Pos. n laut Stimmzettel

TABELLE 4.1: Feldbezeichnungen für Wahlen im Rahmen von NRW-Kommunalwahlen auf Basis der Datensatzbeschreibung Wahlergebnisse Version 0.3[2].

„Kandidaten“-Dateien enthalten verschiedene Angaben zu kandidierenden Personen, darunter neben dem vollständigen Namen auch Beruf und Geburtsjahr. Die Personen sind anhand von Partei-Kürzel und Langname als virtuelle Schlüsselwerte Parteien zuzuordnen, die wie zuvor beschrieben auch nur virtuell bestehen. Welche Kandidierende in einem Gebiet zu berücksichtigen sind, lässt sich über die Angabe `kandidat-gebiet-nr` verknüpft ermitteln. Es gibt formell keinen Zusammenhang mit den Angaben zu Stimmzettelgebieten. Die `kandidat-gebiet-bezeichnung` aus der „Wahlparameter“-Datei hat hier ebenfalls keine semantische Bedeutung.
Auch die Angabe des Listenplatzes (`kandidat-listenplatz`) erfolgt bei den Kandidaturdatensätzen. Einzelne Listenplatz-Werte können mehrfach pro „Partei“ vorkommen, wenn es sich um eine Wahl mit mehreren Stimmzettelgebieten handelt. Dies ist bei den Kandidaturdaten dann daran erkennbar, dass es mindestens eine `kandidat-gebiet-nr` gibt, die pro Partei mehrfach vorkommt.
Handelt es sich um eine Personenwahl unabhängig von Kandidaturen in unterschiedlichen Wahlgebieten, sind beide Felder nicht befüllt [55].

Zum Datenstandard gehören auch **„Strassen“-Dateien** (Straßen) mit Informationen zu Wahlgebietszuordnungen (niedrigste Ebene) von Straßenbereichen, und **„Wahlraeume“-Dateien** (Wahllokale), ebenfalls direkt Wahlgebieten niedrigster Ebene zugeordnet.

4.3 Praxis

In diesem Abschnitt wird dargestellt, welche Schwierigkeiten sich bei der maschinellen Verarbeitung von Daten gemäß des offenen Wahldatenstandards im aktuellen Zustand (Version 0.3) ergeben.
Zunächst geht es um die Dateitypen und bereitgestellten Dateien gemäß des aktuellen Standards, im Unterkapitel Datenformat wird grundsätzlich die Wahl des Datenformats diskutiert.

4.3.1 Datenfehler

Bedingt durch die bisherige manuelle Datenbereitstellung kommt es bei der Verarbeitung der Daten zu diversen Fehlern unter anderem aufgrund ungültiger Feldbezeichnungen oder fehlerhaften Schlüsselwerten. Da davon auszugehen ist, dass Fehler dieser Art bei automatisierter Bereitstellung ohne menschlichen Einfluss nicht mehr geschehen, wird an dieser Stelle nicht näher darauf eingegangen.

Einige Vorkommnisse sind nicht sicher als Fehler einzuschätzen, teilweise aufgrund mangelnder Dokumentation. Bei welchen Feldern es sich um Schlüsselfelder (sinngemäß Primär-/Fremdschlüssel) handelt ist beispielsweise nicht explizit und nicht vollständig dokumentiert.
Ob es sich bei existierenden Definitionen, bei denen Eigenschaften mit Nummer und Name bestehen, bereits bei einer Namensangabe ohne Nummer um gültige Daten handelt kann also nicht sicher gesagt werden und muss im Zweifel als Fehler betrachtet werden, da, sofern nicht andere Vereinbarungen bestehen, ID-Felder üblicherweise als Schlüssel zur Zuordnung auch z. B. zu externen Datenquellen dienen, und nicht etwa ein kombinierter Schlüssel aus einer ID und einem Namen.

4.3.2 Wahlarten

Die Unterscheidung von Wahlarten soll über das Feld `wahl-name` möglich sein, die Werte von diesem Feld sollen sogar behördenübergreifend einheitlich sein [53]. Damit wird auch eine gewisse Verwendbarkeit bei der maschinellen Verarbeitung nahegelegt. Dafür spricht ebenfalls, dass die in diesem Feld anzutreffenden Bezeichnungen den Bezeichnungen nicht welche sind, die als geeignet für die lokale Darstellung in einem Interface erscheinen (Beispiel: „Ratswahl NRW“, nicht etwa „Ratswahl“)[3]. Diese Bezeichnungen sind bereits unabhängig vom Wahldatenformat bei votemanager-Portalen in URLs oder Dateinamen anzutreffen gewesen.

Da diese Wahlarten nicht dokumentiert sind und nicht einmal der Quellcode der Software votemanager öffentlich verfügbar ist, kann die Angabe der Wahlart zurzeit also überhaupt nicht maschinell im Rahmen der Verarbeitung von Wahldaten verwendet werden, sie ist nichts weiteres als eine weitere Bezeichnung.
Bei der Implementierung einer Anwendung wie der im Rahmen der Bachelorarbeit zu entwickelnden kann die Angabe also nicht verwendet werden, um Grundannahmen/Parameter, die die auszuwertende Wahl betreffen, festzustellen und in der Verarbeitung zu beachten, beispielsweise um Fehlermeldungen bei unerwarteten Konstellationen in den Daten bereitstellen zu können. Stattdessen wird es notwendig, beispielsweise die Tatsache, ob es sich um eine Bezirksvertretungswahl handelt, anhand der Daten während der Datenverarbeitung zu ermitteln (vgl. Abschnitt 6.3.2.3).

[3] Gleichzeitig sind die Angaben in der Wahlparameter-Datei im Feld `wahl-bezeichnung` zu ausführlich für die Darstellung z. B. in einem Menü. Wirklich kurze Namen müssen also bisher anwendungsseitig supplementiert oder über ein Abkürzungsverzeichnis ermittelt werden.

Da auch unabhängig von der Angabe der Wahlart bisher nur eine Art von Ergebnisdateien (für Kommunalwahlen in NRW) dokumentiert ist, können zum jetzigen Zeitpunkt nicht unmittelbar weitere Wahlarten bei der Wahldatenauswertung berücksichtigt werden, darunter insbesondere die, die sich nicht am Modell einfacher Stimmen oder Erst- und Zweitstimmen orientieren, also beispielsweise das System des Kumulierens und Panaschierens.

4.3.3 Annahmen zu Gebieten

Mangels Dokumentation können im Umgang mit dem Datenmodell zunächst Annahmen getroffen werden, die sich als falsch herausstellen und zu Fehlimplementierungen führen könnten.

In Zusammenhang mit Gebietsdefinitionen muss klargestellt werden, dass die Gebiete auf verschiedenen Ebenen nicht „ineinander“ liegen.
Gebiete höherer Ebenen werden wie zuvor beschrieben nur virtuell anhand von Eigenschaften an jedem Gebiet niedrigster Ebene definiert und ergeben sich als Menge der Gebiete niedrigster Ebene, die auf der jeweiligen Ebene die Angaben des Gebiets der jeweiligen höheren Ebene führen.
Daraus sollte nicht abgeleitet werden, dass eine Speicherung von Gebieten in einer geschachtelten Form wie einer Baumstruktur erfolgen kann, indem unmittelbare Beziehungen beispielsweise von einem Ebene-4-Gebiet zu Ebene-3-Gebiet, von 3 zu 2, und schließlich von 2 zu den Gebieten niedrigster Ebene abgebildet werden.
Es ist nur möglich, jeweils von einem Gebiet höherer Ebene zu einem Gebiet niedrigster Ebene eine direkte Verknüpfung herzustellen.

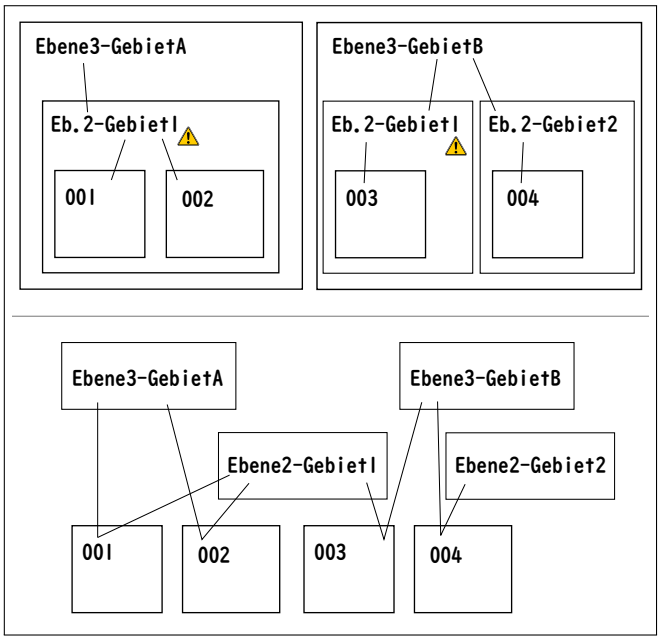

ABBILDUNG 4.1: Gegenüberstellung der Möglichkeiten der Vorhaltung und Verknüpfung von Gebieten verschiedener Ebenen. Bei einer geschachtelten Datenspeicherung kann es dazu kommen, dass es auf einer über der niedrigsten liegenden Ebene mehr als ein Objekt geben muss, welches ein bestimmtes Gebiet darstellt. Eine geschachtelte Darstellung sollte nur indirekt ermittelt werden und rein nachrichtlich sein.

Ist es gewünscht, beispielsweise für ein Gebiet der Ebene 3 zu ermitteln, welche Ebene-2-Gebiete darin liegen, muss dies anhand der Ebene-2-Zuordnungen der Gebiete niedrigster Ebene erfolgen, die das Ebene-3-Gebiet ausmachen.
Diese Informationen können gegebenenfalls nützlich sein, um sie nachrichtlich darzustellen, doch sie dürfen keinesfalls dazu dienen, Berechnungen durchzuführen, indem beispielsweise die Wahlergebnisse für das Ebene-3-Gebiet als Aggregation der Ergebnisse der jeweiligen Ebene-2-Gebiete und nicht als Aggregation der Gebiete niedrigster Ebene berechnet werden.
Hintergrund ist, dass ein relevantes Ebene-2-Gebiet mehr Gebiete niedrigster Ebene umfassen kann, als tatsächlich auch Teil des jeweiligen Ebene-3-Gebietes sind, da die Eingrenzung der Gebiete verschiedener Ebenen ineinander überhaupt nicht als gegeben gesehen werden kann.
Zu dieser möglichen Falschannahme ist festzuhalten, dass sie nicht etwa aufgrund einer solchen Suggestion im Interface von votemanager auftreten kann, da aus der dortigen Darstellung eine solche geschachtelte Datenspeicherung gar nicht unterstellt wird. Wird beispielsweise ein Ebene-2-Gebiet aufgerufen, wird nur aufgelistet, welche Gebiete niedrigster Ebene es ausmachen, nicht aber, in welchen höherliegenden Gebieten es liegt.
Die Falschannahme kann vielmehr dadurch entstehen, dass Anwendungsentwickelnde die geschachtelte zunächst als geeignete und „clevere“ Form der Datenspeicherung sehen könnten — was auch in vielen Fällen problemlos funktioniert, da es solche „Überschneidungen“ nicht immer gibt — die mögliche Problematik fällt erst dann auf, wenn andere Daten betrachtet werden, und die Anwendung, die immer ein Objekt je Gebiet anlegt, plötzlich ein weiteres erstellen soll und die Datenverarbeitungslogik nicht darauf ausgelegt ist.

Eine weitere Falschannahme kann die Einzigartigkeit von Bezirksnummern sein. Dies ist zwar nirgends explizit dokumentiert, es fällt aber bei Betrachtung der Beispieldatensätze einer Kreiswahl auf[4]. Für die Einzigartigkeit muss der Gemeindeschlüssel der für das Gebiet relevanten Behörde hinzugezogen werden[5].
Dies zu einer Grundvoraussetzung zu machen ist allerdings fraglich, zumindest solange es nicht explizit so dokumentiert ist, dass der Schlüssel für Gebietsdatensätze auch das Gemeindeschlüssel-Feld beinhaltet[6]. Es fehlt bislang auch die explizite, ebenenbezogene Angabe, auf welchen Gebietsebenen zu erwarten ist, dass sich Bezirksnummern aufgrund von verschiedenen Gemeinden überschneiden können, und auf welchen Gebietsebenen es sich um Gebietsnummern handelt, die nicht in einer Gemeinde, sondern „global“, also quasi wahlweit, eindeutig sein sollen. Die Angabe der Kandidaturgebietsebene in den Wahlparametern lässt sich nicht verwenden, weil sie in diesem Zusammenhang keine völlig eindeutige Bedeutung hat[7] und auch nur bei Wahlen vorhanden ist, bei denen es Kandidaturgebiete

[4] An dieser Stelle ist auch auf den Abschnitt Datenumfang zu verweisen — Es ist unerwartet, dass bei den bisher bereitgestellten Datensätzen überhaupt Daten bis herunter zur Gemeindeebene bereitgestellt werden.

[5] Quelle dafür ist bislang ein Tweet der kdvz: https://twitter.com/OpenDataKDVZ/status/1325789043133706241

[6] Es fehlen, wie an anderen Stellen in diesem Kapitel aufgeführt, brauchbare Angaben, woraus Schlüssel überhaupt bestehen.

[7] Darüber hinaus ist diese Angabe nicht einmal mehr so gestaltet, dass eine maschinelle Verwendbarkeit suggeriert wird. Dies wird im nächsten Absatz, in dem die Kandidaturgebiete thematisiert werden, behandelt.

gibt[8].
Solange keine expliziten Angaben bestehen, muss eine Anwendung, die die Daten verarbeitet und eventuelle Duplikate als Fehler erkennen möchte, die Information bezüglich der erwarteten Eindeutigkeit der Gebietsnummern separat supplementiert bekommen und diese Situationen gegebenenfalls unterschiedlich behandeln.

Ein weiteres Thema im Zusammenhang mit Falschannahmen zu Gebieten ist der Umgang mit Kandidaturgebieten und Stimmzettelzuordnungsgebieten.
Wie bereits bei der Datenformatserläuterung beschrieben, stimmen die Nummern und Bezeichnungen oft mit denen von Gebieten überein, das Datenmodell ist aber in Wirklichkeit nicht darauf ausgerichtet, dass eine Zuordnung von Kandidatur- oder Stimmzettelgebieten zu tatsächlichen Gebieten irgendeiner Ebene geschehen soll. Diese Annahme wurde zunächst dadurch unterstützt, dass es in der Wahlparameter-Datei das numerische Feld `kandidat-gebiet-ebene` gab, wodurch eine gewisse Verknüpfbarkeit und Verwendbarkeit suggeriert wurde. Darüber hinaus war es verwunderlich, dass es eine solche numerische Angabe nicht auch für Stimmzettelzuordnungsgebiete gab, obwohl der Umgang damit ähnlich zu dem Umgang mit Kandidaturgebieten ist.
Dieser Umstand wurde in Version 0.2 dadurch verbessert, dass das genannte Feld durch das textliche Feld `kandidat-gebiet-bezeichnung` ersetzt wurde, was weniger stark auf eine angedachte maschinelle Verwendung zur Verknüpfung hinweist.
Weiterhin ist es problematisch, dass laut der unter „Allgemein“ zur Verfügung gestellten Grafik[9] bei Kandidaturgebieten die Relation über die `kandidat-gebiet-nr` geschieht, bei Stimmzetteln aber per `stimmzettel-gebiet-bezeichnung`, obwohl auch dort die Angabe der Nummer vorhanden ist [53]. Da ansonsten generell keine Angaben dazu vorliegen, bei welchen Feldern es sich um Schlüsselfelder handelt, ist es an dieser Stelle nur möglich, bei der Datenverarbeitung so vorzugehen, wie es sich anhand der bereitgestellten Datensätze als korrekt erweist, und nicht etwa so, wie es in der Grafik oder Dokumentation dargestellt ist, die zum jetzigen Zeitpunkt als unvollständig zurückzuweisen ist.

4.3.4 Datenmodell

Folgend soll die mangelnde Normalisierung und mangelnde explizite Abbildung von Objekten behandelt werden. Die unvollständige Dokumentation des Datenformats lässt auch in diesem Zusammenhang Fragen unbeantwortet, es ist nicht zu erschließen, ob das jetzige Datenmodell „aus gutem Grund“ so gewählt wurde. Zum jetzigen Zeitpunkt müssen bei der Datenverarbeitung zusätzliche Schritte erfolgen, beispielsweise müssen bei der Abbildung von Objekten teilweise Klassen berücksichtigt werden, die es im Datenformat nicht explizit gibt, ohne die die Verarbeitung aber schwer möglich wäre. Dies wird insbesondere bei dem Umgang mit Parteien und Gebieten klar.

In Zusammenhang mit Parteien fällt spätestens bei Vorhandensein mehrerer Stimmzettelgebiete auf, dass die Eigenschaften jeder Partei bei jedem Vorkommnis auf einem Stimmzettel

[8] Der Grund dafür, dies überhaupt als potenzielle Idee einzubringen liegt darin, dass ein Tweet von der kdvz so interpretiert werden kann: https://twitter.com/OpenDataKDVZ/status/1326808322624909313

[9] https://web.archive.org/web/20201217163146/https://offenewahldaten.de/wp-content/uploads/2020/01/Beziehunge_Dateien_offene_WahldatenXX3X.png

redundant wiederholt werden. Wenn in einem Interface auswählbar sein soll, die Ergebnisse einer bestimmten Partei anzuzeigen, wäre es suboptimal, die formell definierten „Parteien“ (Stimmzetteleinträge) alle separat, möglicherweise mit einem Stimmzettelgebiets-Suffix zur Unterscheidbarkeit, anzuzeigen. So könnte eine stadtweite Darstellung auch nicht umgesetzt werden. Zurzeit muss bei der Datenverarbeitung eine Problemumgehung (vgl. 6.3.2.3) stattfinden, wenn beispielsweise bei einer Bezirksvertretungswahl die Ergebnisse stadtweit dargestellt werden sollen und die Parteinamen nicht mehrfach aufgeführt werden sollen.

Sinnvoller wäre es, wenn das Datenmodell so erweitert werden würde, dass bei den Stimmzettelzuordnungen nur mit Schlüsselwerten zu arbeiten ist, und Parteien mit ihren Eigenschaften für die gesamte Wahl einheitlich definiert werden. So kann auf ein Partei-Objekt bei mehreren Stimmzetteln verwiesen werden, und die Eigenschaften müssen nicht mehr bei jedem Stimmzetteleintrag redundant wiederholt werden. Abbildung 4.2 stellt dies vereinfacht dar.

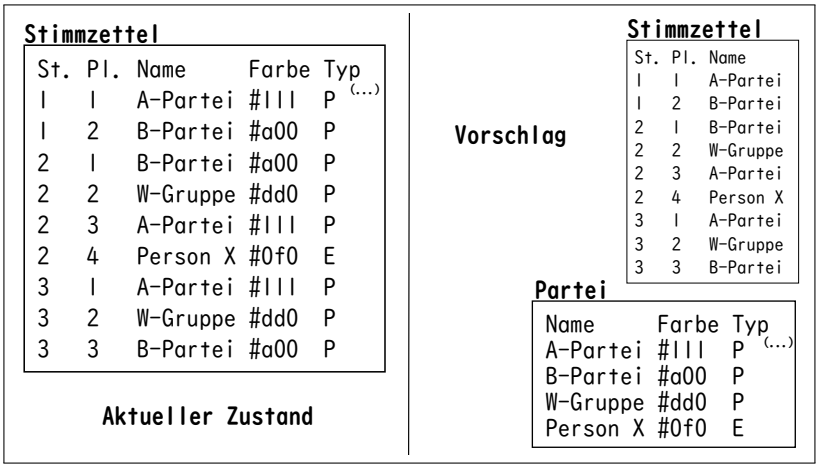

ABBILDUNG 4.2: Eine vereinfachte Darstellung der aktuellen Definition von Parteien und einer möglichen Alternative. In Wirklichkeit erfolgt eine Unterscheidung von Parteikürzel und vollem Namen, außerdem ist nicht klar, was als Schlüsselfeld(er) verwendet werden soll. Im vereinfachten Vorschlag für den Zweck der Erläuterung ist der „Name“ als Schlüsselfeld zu verwenden.

Es ist nicht bekannt, ob das Datenmodell in der Software votemanager so gestaltet ist, dass die Parteien durchaus explizit wahlweit und nicht stimmzettelweit definiert sind, oder ob es dort ebenfalls so suboptimal gestaltet ist und eine Problemumgehung wie die folgend beschriebene im Interface stattfindet.[10] Ist dies der Fall, erklärt dies die Beschaffenheit des aktuellen Datenmodells im Offenen Wahldatenstandard, da dann anzunehmen ist, dass sich an dem Datenmodell der Ursprungssoftware orientiert wurde.

Vorsicht ist im Zusammenhang mit dem bisherigen Datenmodell der Stimmzettelzuordnungen/Parteien auch deswegen geboten, weil es Gebiete auf höheren Ebenen geben kann,

[10] Beim Beispiel der Bezirksvertretungswahlen ist eine Unterscheidung ein und derselben Partei auf der Gesamtübersichtsseite im votemanager-Interface nicht zu erkennen, es ist ein einheitliches Erscheinungsbild, obwohl dort mehrere Stimmzettelgebiete zugrundeliegen. http://wahlergebnisse.stadt-hagen.de/prod/KW2020/05914000/html5/Bezirksvertretungswahl_NRW_245_Gemeinde_Stadt_Hagen.html

dessen Gebiete niedrigster Ebene in unterschiedlichen Stimmzettelgebieten liegen[11].
Bei der Datenverarbeitung für die Darstellung in einem Interface müssen die Parteien mit übereinstimmenden Angaben zuvor gemeinsam behandelt werden, so dass nicht auffällt, dass es sich eigentlich um unterschiedliche Datenobjekte gehandelt hat. Dies kann über die Herleitung „virtueller“ wahlweiter Partei-Objekte, dessen Eindeutigkeit mit bestimmten Schlüsselfeldern definiert werden soll, geschehen.
Die daraus resultierende einheitliche Darstellung entspricht auch der Vorgehensweise, die scheinbar im votemanager-Interface zum Einsatz kommt[12]. Bei dem Beispiel der *Gesamtübersichtsseite* der Bezirksvertretungswahl handelt es sich schließlich um eine ähnliche Situation, die aber offensichtlicher ist als die soeben beschriebene.

Auch Gebietsdaten sollten in Zusammenhang mit expliziter Abbildung von Objekten nicht unerwähnt bleiben. Die bisherige Methode, eine Gebietszuordnungstabelle zu haben, in der ein Datensatz einem Gebiet niedrigster Ebene entspricht, und die Gebiete höherer Ebenen nur virtuell über die Eigenschaften zu ermitteln sind, ist zu hinterfragen. Es kommt zu redundanten Angaben von Eigenschaften, zurzeit zumindest bei der Gebietsbezeichnung. Wäre es gewünscht, beispielsweise eine URL-Eigenschaft hinzuzufügen, die auf eine menschenlesbare Website für das jeweilige Gebiet führt, müsste der jeweilige Wert bei jedem Vorkommnis dieses Gebiets redundant aufgeführt werden.
Außerdem würde eine explizite Abbildung von Gebieten aller Ebenen auch die Perspektive bieten, die Zuordnung von Kandidatur- und Stimmzettelgebieten über diese Gebiete geschehen zu lassen, und nicht mehr, wie in vergangenen Absätzen beschrieben, über „Gebiete“, die sich als Aggregation von anderen, nicht auf echte Gebiete bezogene Eigenschaften, bilden. Damit dies dann auch bei der Datenverarbeitung vorteilhaft umgesetzt werden kann, braucht es aber weitere Parameter, die definiert werden müssen, und möglicherweise auch ein anderes Datenformat, welches für solche Datenstrukturen und -beziehungen geeigneter ist.

4.3.5 Datenumfang

Teilweise ist noch nicht klar, welche Daten entweder inhaltlich bereitgestellt, oder von ihrer Art her im Format berücksichtigt werden.

Zunächst ist es als unerwartet zu bezeichnen, dass Daten, die auf Kreisebene bereitgestellt werden, bis zur Gemeindeebene herunter gehen. Von der Darstellung im votemanager ausgehend wird bei Wahlen auf Kreisebene normalerweise nur die Möglichkeit bereitgestellt, auf Kreiswahlbezirksebene oder Gemeindeebene Ergebnisse darzustellen, nicht aber auf Ebene beispielsweise von Gemeindewahlbezirken oder auf der niedrigsten Ebene, den Stimmbezirken.

[11] Beispiel: Es werden Daten für einen Wahlbezirk (Ebene 2) angezeigt, der über die Grenzen der Stadtbezirke (entsprechen inhaltlich der Ebene 3 und sind ebenfalls Basis der Stimmzettelgebiete) hinweg verläuft. Bei den Parteien gibt es viele Überschneidungen, einzelne kommen möglicherweise nur auf einem der Stimmzettel vor. Grundlegend handelt es sich um gemäß des Datenmodells separate Stimmzettelzuordnungsobjekte mit bei ein und derselben Partei übereinstimmenden Angaben, die von sich aus aber nichts miteinander zu tun haben.

[12] Beispiel: http://wahlergebnisse.stadt-hagen.de/prod/KW2020/05914000/html5/Bezirksvertretungswahl_NRW_245_Wahlbezirk_20_EilpeZentrumOberhagen.html

Es ist fraglich, ob dies bei der realen Datenbereitstellung in den Wahlergebnisportalen so beibehalten wird, oder ob dies nur eine durch die manuelle Datensatzerstellung bedingte Erscheinung ist.
Technisch ist es zwar möglicherweise durchaus umsetzbar, weil die Wahlergebnisportale des Kreises und der angehörigen Gemeinden erfahrungsgemäß oftmals gemeinsam gehostet werden, es ist aber davon auszugehen, dass das nicht immer der Fall ist. Bei Gemeindegruppierungen anderer Art, beispielsweise bei Wahlen der Verbandsversammlung des Regionalverbands Ruhr, oder wenn Ergebnisse anderer Wahlen gar landesweit in einem Datensatz zusammengefasst angeboten werden sollen, ist davon auszugehen, dass eine Zusammenführung nicht ohne Einsatz gegebenenfalls noch nicht existierender Mittel geschehen kann, allein schon weil in dem Fall verschiedene Systeme berücksichtigt werden müssen.

Ein anderes Thema in Zusammenhang mit dem Datenumfang ist die Bereitstellung von Zuordnungen von Gebieten niedrigster Ebene zu ihren Briefwahlbezirken.
Briefwahlbezirke sind an sich, als eine weitere Quelle von Wahlergebnissen, ebenfalls als Gebiete auf der niedrigsten Ebene abgebildet. Die Zuordnung von Gebieten niedrigster Ebene zu dem Briefwahlbezirk, in dem die Wahlberechtigten aus dem jeweiligen Gebiet per Brief wählen können, erfolgt je nach örtlicher Situation unterschiedlich. In manchen Städten entsprechen Briefwahlbezirke Gebieten auf einer höheren Ebene[13], in manchen gibt es teilweise eins-zu-eins-Zuordnungen von Gebieten niedrigster Ebene[14], und in manchen Städten können die Briefwahlbezirke als Aggregationen „zwischen“ Gebieten auf Ebene 1 und 2 beschrieben werden[15].

Bei der geographischen Darstellung von Ergebnissen anhand von Gebieten niedrigster Ebene können Ergebnisse aus Briefwahlbezirken nicht berücksichtigt werden, während auf der nächsthöheren Ebene gegebenenfalls zu viel räumliche Auflösung verloren gehen kann. Wenn es eine maschinenlesbare Zuordnung von Briefwahlbezirken und ihren dazugehörigen Gebieten niedrigster Ebene gäbe, könnte in den Städten, wo die Briefwahlbezirkszuordnungen eine virtuelle, formell nicht existierende „Zwischenebene“ ausmachen, eine korrektere und dabei trotzdem feiner (als auf der nächsten formellen Ebene) aufgelöste Darstellung ermittelt werden (vgl. Abbildung 4.3). Die dafür notwendigen Geodaten müssten dafür allerdings zusätzlich ermittelt werden, da diese virtuelle Zwischenebene unwahrscheinlich Teil von Geodatenbereitstellungen sein wird. Eine Herleitung der Zwischenebenenflächen in einer Anwendung wäre möglicherweise über das clientseitige Verschmelzen von Gebieten der niedrigsten Ebene zu implementieren.

Im Interface von votemanager-Portalen ist eine solche aggregierte Darstellung bisher nicht möglich, diese virtuellen Briefwahlbezirkszuordnungen existieren auch nicht explizit als aufrufbare Detail- oder Übersichtsseite, die sowohl die Briefwahl- als auch die zugehörigen

[13] Beispiel: Hagen http://wahlergebnisse.stadt-hagen.de/prod/KW2020/05914000/html5/Ratswahl_NRW_205_Uebersicht_stbz.html. Es gibt 26 Briefwahlbezirke, die die Stimmbezirke so zusammenfassen, wie es die 26 Ratswahlbezirke (Ebene 2) tun.

[14] Beispiel: Stadt Euskirchen https://wahlen.citkomm.de/KW2020/05366016/html5/Ratswahl_NRW_137_Uebersicht_stbz.html. Es gibt eins-zu-eins-Zuordnungen, aber auch Briefwahlbezirke, die mehreren (z. B. besonders kleinen) Stimmbezirken zugeordnet sind.

[15] Beispiel: Wuppertal https://wahlen.wuppertal.de/kw2020/05124000/html5/Ratswahl_NRW_39_Wahlbezirk_ElberfeldMitte.html (beispielhaft ein Link auf einen konkreten Ratswahlbezirk, ebendieser wird in der Erläuterungsgrafik verwendet).

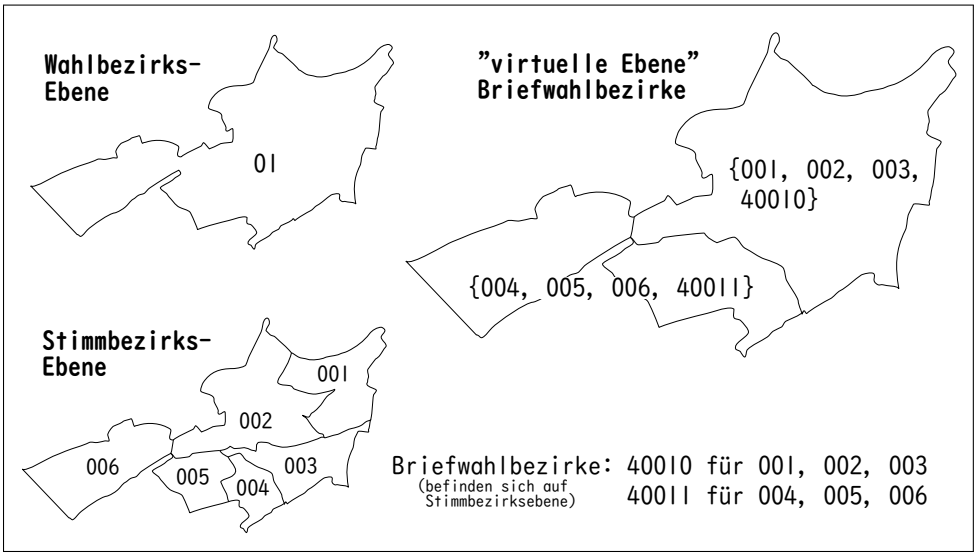

ABBILDUNG 4.3: Die Berücksichtigung von Briefwahlbezirkszuordnungen ermöglicht eine Darstellung, die (im Gegensatz zu der niedrigsten Ebene, den Stimmbezirken) sowohl *alle* Ergebnisse berücksichtigt als auch räumlich feiner aufgelöst ist, als die nächsthöhere Ebene, auf der die Berücksichtigung der Briefwahlergebnisse sonst erst möglich werden würde. Basis der eigenen Darstellung: Bezirkseinteilungen aus dem Geoportal Wuppertal [56, Wahlbezirke].

Urnenwahlergebnisse beinhaltet.

Die Zuordnungen sind in votemanager, wenn sie in den zugrundeliegenden Daten überhaupt abgebildet sind, nur als Angaben auf den jeweiligen Detailseiten der Gebiete niedrigster Ebene vorhanden. Bei einem normalen Gebiet steht dann also, zu welchem Briefwahlbezirk es gehört, und bei einem Briefwahlbezirk steht, welche anderen Gebiete niedrigster Ebene diesem zugeordnet sind.

Ebenfalls im Kontext von Briefwahlbezirken fehlt bislang ein expliziter Hinweis darauf, dass Briefwahlbezirke wie unter 2.3 erwähnt zusammengefasst wurden oder eine Auszählung von Briefwahlergebnissen bspw. aus anderen Gemeinden erfolgt. Es sollte eine Möglichkeit geben, dies nicht nur dadurch zu erkennen, dass die Wahlbeteiligung in einem Gebiet über 100 % liegt, weil es nicht immer der Fall ist. In votemanager ist so etwas auch nicht besonders erkennbar — es wird nur im Falle einer Wahlbeteiligung von über 100 % darauf verwiesen, dass die Wahlbeteiligung „zur Zeit“ aufgrund eines hohen Briefwahlanteils nicht ermittelt werden könne, was eine eher unspezifische Aussage ist[16].

Ein drittes Thema im Kontext Datenumfang hängt mit dem Straßen-Dateityp zusammen. Eine Auflistung von Zuordnungen auf Hausnummerebene wäre wünschenswert, ist aber laut Hersteller aus technischen Gründen nicht möglich [57, 52]. Da es sich also nur um Wertebereiche handelt, müssen zur Zuordnung mit einzelnen Hausnummern zusätzliche Vorberechnungen erfolgen. Bei Ansicht der bisher existierenden Straßen-Datensätze fällt auf, dass in manchen Fällen sehr spezifische Zuordnungen einzelner Hausnummern geschehen, obwohl es sich dabei nicht einmal um Wohngebäude handelt. Es ist also davon auszugehen, dass für die Datenverwaltung in votemanager gewisse vorhandene Daten reduziert werden, um

[16] Beispiel vgl. https://wahlen.kdvz-frechen.de/kdvz/kw2020/05366040/html5/Landratstichwahl_NRW_123_Wahlbezirk_103_Weilerswist_WLW_SuedNord.html

einem Datenmodell zu entsprechen, welches auch dann funktionieren soll, wenn eben bei *gewissen* Datenschnittstellen keine einzelnen Hausnummern zur Verfügung gestellt werden. Ob diese Beschränkung Auswirkungen auf die Definition eines eigentlich generell anwendbaren Wahldatenformats haben sollte, ist fraglich.

Immerhin gab es in Version 0.2 eine Verbesserung in der Definition des Offenen Wahldatenformats: Jetzt werden die Wertebereiche separat voneinander ausgegeben, mit Nummern und Zusätzen (Buchstaben) getrennt. So kommt es nicht mehr zu Mehrdeutigkeiten, was bei einer maschinellen Verarbeitung zu tun ist, wenn bisher beispielsweise ein Bereich angegeben war, dessen letzte Hausnummer in Wirklichkeit auch Gebäude mit Hausnummer mit Zusatz hatte und es falsch wäre, anzunehmen, dass die Buchstaben dazu gehören.

Im Rahmen dieser Bachelorarbeit wurden Zuordnungsdaten zur Generierung von Gebiets-Geodaten verwendet (vgl. Abschnitt 5.2), allerdings im bisher vorhandenen csv-Format aus votemanager. Dabei wurden die Zuordnungsdaten verwendet, um einzelnen Hausnummern Gebiete zuzuordnen, anhand der (teilweise wie beschrieben mehrdeutigen) Wertebereiche. Mit dem verbesserten Datenmodell seit Version 0.2 des Offenen Wahldatenstandards wäre es denkbar, dass eine Referenzimplementierung für die Zuordnung zu einzelnen Hausnummern mit der Dokumentation des Formats bereitgestellt wird, und zunächst die nicht auf einzelnen Hausnummern basierende Datenbereitstellung beibehalten werden kann.

4.3.6 Datenformat

Grundsätzlich ist die Wahl des Dateiformats csv zu diskutieren. Als Gründe werden zwar unmittelbar auf der Website nur die höhere Kompatibilität und Verbreitung angegeben [53], es lässt sich aber auch ableiten, dass davon ausgegangen wird, dass der Umgang mit dem Dateiformat nicht kompliziert ist und eine niedrige Einstiegshürde besteht. Insbesondere mit Blick auf das bisherige csv-Format im votemanager kann dies als zutreffend bewertet werden, auch wenn die Verwendung einer csv-Ergebnisdatei alleine nicht ausreichend war, und die Stimmzettelpositionen manuell verknüpft werden mussten. Im Gegensatz zu Daten nach dem Offenen Wahldatenstandard mussten aber nicht mehrere maschinenlesbare csv-Dateien unter anderem unter Einsatz von Schlüsselwerten verknüpft werden.

Die Verwendung von csv-Dateien nicht als Ergebnis vorheriger Verarbeitung, sondern als Ursprungsdatensatz, anhand dessen die Relationen zwischen dessen Datensätzen abgebildet werden müssen, sorgt dafür, dass die zunächst angedachte Praktikabilität oder Einfachheit schließlich aufgrund von implizit vorausgesetzten tieferen technischen Kenntnissen nicht völlig realistisch bestehen kann.
Dazu kommt die Redundanz, die dadurch entsteht, dass manche Objekte nicht tatsächlich als Datensatz abgebildet werden, sondern implizit aus Eigenschaften von Datensätzen abgeleitet werden müssen und so nur virtuell bestehen. Sind zusätzliche Eigenschaften dieser virtuellen Objekte notwendig, bedeutet dies auch die redundante Angabe der Werte an nicht nur einem Datensatz, sondern an allen relevanten Vorkommnissen.
Besser wäre dafür also die reine Verwendung von Schlüsselwerten, um Relationen abzubilden, und damit auch die möglichst ausschließlich explizite, normalisierte Abbildung von

Objekten, dafür müssten auch neue „Dateitypen“/Modellklassen definiert werden.
In der bisher existierenden Dokumentation sind keine vollständigen oder korrekten Angaben dazu, bei welchen Feldern es sich um Schlüsselfelder handelt, vorhanden.

Um *darüber hinaus* schließlich die Praktikabilität bei dem (maschinellen) Umgang mit Daten gemäß des Offenen Wahldatenstandards zu erhöhen, sollte die Verwendung des csv-Formats durch ein anderes Format wie JSON ersetzt werden. So könnten die Möglichkeiten der Verwendung von Baumstrukturen, Listen und weiteren Eigenschaften von JSON zum Vorteil genutzt werden. Dabei wären große Anpassungen bei der Definition der Datenstruktur angemessen, es sollte sich nicht um eine reine zeilenbasierte Umwandlung des bisherigen Formats handeln, wie es etwa als Möglichkeit im Konzeptpapier erwähnt wird [52].
Es hätte auch den Vorteil, dass eine Aufteilung auf mehrere Dateien, wie es bisher bei dem auf csv-Dateien basierenden Standard zwingend notwendig war, nicht mehr notwendig sein wird. So wäre es möglich, nur noch eine einzige Datei laden zu müssen, und nicht den Umgang mit mehrfachen Datenabfragen und eventuell sogar auftretenden unterschiedlichen Datenständen verwalten zu müssen.
Bisher ist auch grundsätzlich unklar, wie die Pfade zu den verschiedenen csv-Dateien überhaupt verwaltet werden sollen, da Dateinamen auch Zeitstempel enthalten[17]. Dies ist kein haltbarer Zustand, denn für die Verwendung der Daten durch Anwendungen sind stetig verwendbare URLs notwendig, da sie sonst beispielsweise am Wahlabend im Minutentakt in der Anwendungskonfiguration zu aktualisieren sind. Wird die bisherige Vorgehensweise beibehalten, könnte zumindest mit einem stetig erreichbaren „Entrypoint“, über den die aktuellen Dateipfade strukturiert bereitgestellt werden, nachgeholfen werden.

Um dem potenziellen Bedarf nach der Abfrage geringerer Mengen als dem vollständigen OWD-Datensatz mit allen Modelltypen gerecht zu werden, wäre es zunächst denkbar, eine Web-Schnittstelle (sinngemäß API) bereitzustellen, die über Parameter eine Vorauswahl der Rückgabetypen ermöglicht. Eine auf der Abfragesprache GraphQL basierende Schnittstelle kann auch in Frage kommen. Dabei wird grundsätzlich das Datenschema des Servers definiert, und Datenabfragen enthalten darauf basierende Angaben, welche Objekte und Eigenschaften erwünscht sind [58, 59]. So wird bei Abfragen zu jedem Zeitpunkt ein aktueller und auf den anwendungsseitig notwendigen Umfang beschränkter Datensatz bereitgestellt. Für den Umgang mit solchen Schnittstellen gibt es Bibliotheken für viele häufig verwendete Programmiersprachen, darunter eine offizielle Referenzimplementierung in JavaScript [60].

Um die eigentlich gewünschte Praktikabilität beispielsweise für die Verwendung in Tabellenkalkulationssoftware bei geringen Programmierkenntnissen herzustellen, können wiederum csv-Dateien hergeleitet werden, bei denen der Fokus weniger auf maschineller Auswertbarkeit und Normalisierung liegen kann. Diese Herleitung kann mithilfe von Tools geschehen, die auf Daten des dann veränderten Offenen Wahldatenstandards basieren, oder herstellerseitig wie bisher im Wahlergebnisportal geschehen.

[17] Es ist zu berücksichtigen, dass es bisher aber auch gar keine realitätsnahe (vergleichbar mit den bisher bereitgestellten csv-Ergebnisdateien in Votemanger-Portalen) Bereitstellung von OWD-Daten gibt, es handelt sich nur um herunterladbare einzelne Dateien oder eine zip-Datei (oder manuell erstellte Dateien). Das ändert aber auch nichts an der Tatsache, dass die Verwendung von Zeitstempeln in den Dateinamen aktuell vorgeschrieben ist.

Zusammenfassend ist festzuhalten, dass es nicht geeignet erscheint, eine gute Verwendbarkeit für beide Zielrichtungen/-gruppen in einem einzigen Datenstandard umzusetzen, wie es bisher versucht wurde.
Um die Verwendbarkeit des Formats im bisherigen Zustand ohne umfangreiche Umstrukturierungen zu erhöhen, sollte mindestens die Dokumentation verbessert werden, insbesondere hinsichtlich der für die Maschinenverarbeitbarkeit grundlegenden Angaben wie Schlüsselfeldern.

In Zusammenhang mit Grundsätzen der Dateiformate ist auch die Frage der Trennung von Daten nicht nur nach Dateityp, sondern auch nach Wahl zu diskutieren. Es ist unklar, aus welchen Gründen diese Entscheidung getroffen wurde, es war ursprünglich sogar durchaus geplant, alle Wahlen eines Wahltermins gemeinsam in den Dateien abzubilden [52].
Möglicherweise sollte dadurch die Datensparsamkeit durch selektives Laden nur der gewünschten Wahl anstelle aller Wahlen eines Wahltermins ermöglicht werden.
Grundsätzlich möglich wäre es durchaus, Daten mehrerer Wahlen in einer Datei abzubilden, die Unterscheidung erfolgt anhand der drei grundlegenden Angaben (Wahlbehörde, Datum, Wahlname).
Aufgrund der Beschaffenheit des Formats csv könnte es zunächst als problematisch erscheinen, dass bei Ergebnisdateien die Felder sich unterscheiden können, die Auswirkungen beschränken sich bei korrekter Verarbeitung allerdings nur darauf, dass es zu einer erhöhten Anzahl an nicht belegten Feldern kommen kann. Dies ist bereits jetzt in der Ergebnisdatei bei Bezirksvertretungswahlen der Fall (da sich die Länge von Stimmzetteln unterscheiden kann). Bei der Verwendung eines nicht intrinsisch rechteckigen/tabellenförmigen Formats entfällt diese Erscheinung.

4.4 Datenerstellung

Um einen Datensatz gemäß des Standards für Offene Wahldaten zu erstellen, kann teilweise auf bestehende Daten oder alternative Wege der Datenermittlung zurückgegriffen werden.

Bei votemanager-Wahlergebnisportalen als Open Data bereitgestellt werden bisher nur die Wahlergebnisse in verschiedenen Dateien je nach Ebene. Für einen OWD-Datensatz sind nur die Wahlergebnisse auf der niedrigsten Ebene notwendig. Diese Datei kann in einer Tabellenkalkulationssoftware oder in Texteditoren mit erweiterten Funktionalitäten (mehrzeiliges Auswählen und Ausbreitungsfunktion) verwendet werden, um eine „Wahlergebnisse“-Datei nach OWD zu erstellen. Die Ergebnisfelder (A-Zahlen usw.) können unverändert übernommen werden, andere Felder müssen teilweise umbenannt, gelöscht, oder ergänzt werden, so dass schließlich eine standardkonforme Datei besteht. Hier ist bereits auf die Einheitlichkeit der Basisangaben jeder Wahl zu achten (Behörde, Datum, Name), als erster Schritt könnte also zuerst die Wahlparameter-Datei erstellt werden.

Eine „Wahlparameter“-Datei lässt sich ohne maschinelle Unterstützung manuell erstellen. Dazu müssen verschiedene Angaben der votemanager-Darstellung entnommen werden (darunter die „votemanager-Wahlart“ und der Gemeindeschlüssel anhand der URL, der vollständige Wahlname sowie Ebenenbezeichnungen aus dem Interface).

Die „Wahlgebietseinteilungen“-Datei kann nicht anhand der bereits bereitgestellten csv-Ergebnisdatei erstellt werden, da die Zuordnung zu Gebieten höherer Ebene darin nicht enthalten ist. Die manuelle Erfassung ist aufgrund der Fehleranfälligkeit und der hohen Anzahl von Gebieten als unverhältnismäßig einzuschätzen. An dieser Stelle können die Daten über mehrfache automatisierte Abfrage (Scraping) von HTML-Seiten des Wahlergebnisportals ermittelt werden. Dazu kann beispielsweise ein Python-Skript (mit Verwendung einer Bibliothek wie BeautifulSoup[18] zur Verarbeitung der HTML-Dokumentenstruktur) verwendet werden, um zunächst die Übersichtstabelle für die niedrigste Wahlebene zu laden und dann die Detailseite jedes Wahlgebiets zu laden und dort in bestimmten Elementen Verlinkungen zu auf höheren Ebenen gelegenen Gebieten, denen das jeweilige Wahlgebiet zugeordnet ist, zu finden. Das Ergebnis können Daten in einer Struktur sein, die die Angaben passend zum Datenformat enthält, also mit Nummer und Name der Gebiete auf verschiedenen Ebenen jeweils pro Datensatz für jedes Gebiet niedrigster Ebene.
Ein Beispiel für so ein Skript wurde im Rahmen dieser Bachelorarbeit erstellt und ist im tools/-Verzeichnis im Repositorium des Projekts enthalten. Dabei wird explizit kein Anspruch auf Vollständigkeit beziehungsweise allgemeine Verwendbarkeit erhoben, da bei der verwendeten Methode, Scraping, generell Vorsicht geboten sein sollte und in jedem Fall manuelle Überprüfungen und Anpassungen stattfinden müssen.

Daten zu Kandidaturen sind bisher nicht vollumfänglich aus votemanager-Portalen abgreifbar. Insbesondere fehlen Angaben zu vollständigen Wahllisten, nur bei vorliegendem Wahlergebnis werden die gewählten Personen angezeigt, daraus lässt sich nicht ansatzweise eine Liste ermitteln. Ähnliches gilt bei Gebietskandidaturen bei vorliegendem Wahlergebnis, zusätzlich können diese allerdings durchaus vollständig über Detailseiten einzelner Wahlgebiete ermittelt werden, was einen zusätzlichen Aufwand und eventuelle Schwierigkeiten beim Scraping zur Folge hat. Handelt es sich um eine Personenwahl mit behördenweit einheitlichen Kandidaturen lassen sich die Daten in verhältnismäßigem Aufwand manuell abschreiben, da es sich bei solchen Wahlen (Bürgermeister∗innenwahl o. ä.) erfahrungsgemäß um eine einstellige oder niedrige zweistellige Anzahl an Kandidierenden handelt.
Außerhalb der votemanager-Software sind Informationen zu Kandidierenden erfahrungsgemäß in der Regel in Amtsblättern oder alleinstehenden amtlichen Bekanntmachungen in einem nicht ansatzweise maschinenlesbaren Format bereitgestellt.
Insgesamt ist die Ermittlung von maschinenlesbaren Kandidaturdaten, wenn sie nicht maschinenlesbar in einem städtischen Open-Data-Portal veröffentlicht werden, mit individuellem, nicht völlig automatisierbarem Aufwand verbunden. Für die Kernfunktionalität der im Rahmen dieser Bachelorarbeit zu entwickelnden Anwendung sind die Kandidaturdaten nicht notwendig.

Eine Datensatzart, die für die Kernfunktionalität unverzichtbar ist, ist die „Stimmzettel“-Datei. Für die Erstellung einer solchen Datei kann auf die Informationsseite „OpenData-Info“ im votemanager zurückgegriffen werden. Dort sind für jede Wahl, gegebenenfalls je nach Stimmzettelgebiet (im Falle von Bezirksvertretungswahlen), die Parteien mit vollständigem Namen und Stimmzettelposition (nicht in einer strukturierten Form) aufgelistet

[18] Dokumentation von BeautifulSoup: https://beautiful-soup-4.readthedocs.io/en/latest/

(Beispiel vgl. [20]). Die Parteikürzel und -farben sind manuell einer Wahlergebnisseite im normalen Interface, also außerhalb des OpenData-Bereichs, zu entnehmen (über den Seitenquellcode). Anhand dieser an verschiedenen Stellen vorliegenden Informationen kann die „Stimmzettel“-Datei erstellt werden.

Auf die Erstellung von „Strassen“- und „Wahlraeume“-Dateien wird hier nicht näher eingegangen. Die Straßen- und Wahllokaldaten werden gegebenenfalls bereits auf der „OpenData-Info“-Seite im votemanager bereitgestellt, da die Darstellung von Wahllokal-Informationen und Zuordnung von Straßenabschnitten zu Wahlgebieten auch Teil der votemanager-Funktionalität ist. Ist dies lokal nicht der Fall, muss eine alternative Datei z. B. im städtischen Open-Data-Portal gesucht werden oder auf anderem Wege maschinell oder manuell je nach vorliegender Situation ermittelt werden.
Für die Kernfunktionalität der zu entwickelnden Anwendung sind sie nicht benötigt, da diese nicht den Anspruch hat, auch im Vorfeld einer Wahl als Informationsquelle zu dienen. Zuordnungsdaten könnten aber durchaus notwendig werden, wenn wie in Abschnitt 5.2 beschrieben Wahlgebietsflächen anhand dieser generiert werden müssen.

5 Geodaten

Als Geodaten in Zusammenhang mit geographischer Wahlergebnisdarstellung sind insbesondere die Wahlgebietsflächen benötigt. Weitere mögliche Geodaten sind die Standorte von Wahllokalen, Zuordnung von Wohnorten zu Stimmbezirken, und Flächen anderer Ebenen wie Stadtteilen.
Für die zu entwickelnde Anwendung werden nur die Flächen eingesetzt. Im Abschnitt Datenerstellung wird allerdings darauf eingegangen, wie weitere Geodatenquellen zur Erstellung der benötigten Flächendaten eingesetzt werden können.

In Bezug auf die Geometrieart kommen Flächen des Typs Polygon oder MultiPolygon in Frage, dabei handelt es sich um standardisierte Typen für zweidimensionale Oberflächen, wie es in [61] verweisend auf [62] definiert wird.

Die Geometrien brauchen in den Eigenschaften den passenden Schlüsselwert zur Verknüpfung mit den Wahldaten. Ein bestimmter Datentyp wird nicht vorausgesetzt, es wird in der Regel von Zeichenketten ausgegangen. Die Verwendung von numerischen Typen ist nicht immer sinnvoll, da es führende Nullen oder andere Zeichen geben kann.
Eine besondere Behandlung ist im Falle von mehrfach vorkommenden Schlüsselwerten notwendig, wenn beispielsweise eine Datei Wahlgebiete mehrerer Gemeinden enthält, auf einer Ebene, die sich nur auf einzelne Gemeinden bezieht. In dem Fall kann es zu Konflikten kommen und es wird zusätzlich ein Attribut mit einem Gemeindeschlüssel erwartet.

Die Daten können prinzipiell in jedem gängigen, nicht-proprietären Format bereitgestellt werden, da eventuelle Umwandlungen im Vorfeld der Einbindung in die Anwendung durchgeführt werden können, wenn es nicht sowieso aufgrund von nicht zuzuordnenden Schlüsselwerten (z. B. fehlende führende Nullen) notwendig wird. Eine Vorverarbeitung kann auch aufgrund eines unpassenden Koordinatenbezugssystems notwendig werden.
In Web-Kartendarstellungen wird üblicherweise die Web-Mercator-Projektion verwendet, als Datengrundlage werden dabei oft Koordinaten als WGS84-basierte geographische Länge und Breite vorausgesetzt [63], beispielsweise auch an verschiedenen Stellen in den später verwendeten Bibliotheken Leaflet [64] und Tangram [65].

Für die Anwendung wird zunächst das Format GeoJSON verwendet, mit einer Datei pro Gebietsebene.
Das in Web-Mapping-Bibliotheken und Schnittstellen weit verbreitete Format GeoJSON basiert auf JSON und gibt vor, wie Angaben zu geographischen Objekten mit ihren Eigenschaften erfolgen können. Dabei wird auf allgemein bekannte offene Standards für Geodaten zurückgegriffen [61].

Denkbar wäre auch die Verwendung der topologischen Erweiterung TopoJSON, was zu Dateigrößeneinsparungen verhelfen kann. In TopoJSON werden Geometrien nicht mit Koordinaten als Angaben je Objekt abgebildet, sondern über topologieweite Angaben von „arcs“ als Koordinatenabfolgen, auf die verwiesen wird [66]. Tangram unterstützt auch die Verwendung von TopoJSON [65], es müsste nur noch im anwendungsseitigen Code eine

Erweiterung in der Datenquellendefinition vorgenommen werden, wenn bisher statisch die Auswahl des Formats GeoJSON erfolgt.

5.1 Datenquellen

Bestenfalls werden die Daten durch die jeweilige Behörde maschinenlesbar und auffindbar bereitgestellt. Praktisch geschieht dies bisher nur selten so, wie es erwünscht wäre.

Wenn man die Wahlgebietsflächen für den Kreis Euskirchen oder eine der angehörigen Gemeinden sucht, um sie in Verbindung mit den bereitgestellten OffeneWahlDaten-Realdaten zu verwenden, findet man zunächst schriftliche Beschreibungen der Wahlbezirkseinteilungen von Gemeinden und auch auf Kreisebene im PDF-Format, und auf Kreisebene [67] immerhin eine PDF-Karte, die neben den Kreiswahlbezirken auch die Gemeinderatswahlbezirke enthält[1]. Damit lässt sich üblicherweise maschinenlesbar nichts machen, doch es handelt sich um eine *Geospatial PDF*, die georeferenzierte Vektorelemente für die Verwendung in einem Geoinformationssystem enthält [68]. Beim Öffnen fällt allerdings auf, dass die Daten nicht in einem verwendbaren Zustand sind, da es sich teilweise nur um Grenzverläufe handelt und die Gebietsnummern auch nicht korrekt als Eigenschaften abgebildet werden. Dass die Datei überhaupt in dieser Form mit georeferenzierten Vektoren verwendbar ist, scheint eher ein Nebeneffekt als ein bewusst herbeigeführter Zustand zu sein.
Als deutlich hilfreicher stellt sich der Link „Interaktive Karte zum Suchen des eigenen Wahlbezirkes“ heraus, der zu einem Web-GIS führt[2]. Durch die Analyse des Netzwerkverwendung mit Entwicklungstools gängiger Webbrowser lässt sich schnell eine JSON-Datei auffinden, die allerdings nicht im GeoJSON-Format ist, sondern in einem proprietären JSON-Format von Esri. Die Umwandlung nach GeoJSON ist möglich, allerdings muss vorher ein bestimmter Ausschnitt entnommen werden, da das von Esri bereitgestellte Umwandlungstool[3] (JavaScript, Python-Port vorhanden) nicht die vollständige dem Web-GIS entnommene Ausgabe als Input erwartet. Insgesamt handelt es sich insbesondere im Falle eines Erstkontakts mit diesem Format um einen zeitraubenden und gerade im Vergleich mit dem folgenden Beispiel Düsseldorf unverhältnismäßig hohen Aufwand. Dazu kommt, dass in Attributen wieder führende Nullen fehlten und auf Gemeindewahlbezirksebene noch der Gemeindeschlüssel ergänzt werden musste — eine Angabe des Gemeindenamens war immerhin vorhanden.

In der Landeshauptstadt Düsseldorf sind die Wahlgebietsflächen aller Ebenen und auch auf der nachrichtlich verwendeten Stadtteilebene im städtischen Open-Data-Portal auffindbar[4]. Sie liegen in verschiedenen Formaten vor, darunter auch als GeoJSON-Datei mit WGS84-Koordinaten. Sie sind ohne zusätzliche Vorverarbeitung einbindbar, bis auf die Wahlbezirke,

[1] https://www.kreis-euskirchen.de/politik/downloads/Gemeinde_Kreis_Wahlbezirke_2_-_komprimiert.pdf

[2] https://kreis-euskirchen.maps.arcgis.com/apps/View/index.html?appid=eb7c30165f1147ae98af0e693537dae2

[3] GitHub-Repositorium *arcgis-to-geojson-utils*: https://github.com/Esri/arcgis-to-geojson-utils

[4] Hier beispielsweise der Datensatz mit den Kommunalwahlbezirken (entspricht Ebene 2): https://opendata.duesseldorf.de/dataset/wahlgebietseinteilung-d%C3%BCsseldorf-kommunalwahlen

die im Gegensatz zu den Angaben im Wahlergebnisportal und im Wahldatensatz keine führenden Nullen enthielten und z. B. mithilfe eines Geoinformationssystems und einer Feldberechnung (Beispiel: `lpad("Wahlbezirk", 3, '0')`, um links Nullen auf eine Länge von 3 aufzufüllen) korrigiert werden können.

Eine weitere suboptimale Datenquelle ist das auf votemanager basierende Wahlergebnisportal an sich. Dort gibt es die „GeoGrafik"-Funktion[5], die eine GeoJSON-Datei lädt, die auch per Entwicklungstools im Browser abgegriffen und gespeichert werden kann. Die Daten müssen in Bezug auf die Projektion und Schlüsselwerte geprüft werden. Außerdem werden scheinbar möglicherweise keine Multipolygone unterstützt, dies fiel beispielsweise in Wiesbaden auf, wo Probleme bei Enklaven aufgetreten sind.
Die „GeoGrafik"-Funktion wird nicht grundsätzlich bereitgestellt, es muss scheinbar zuvor eine entsprechende Dateneinpflegung seitens der Kommune stattfinden.
Eindeutig fällt allerdings auf, dass es kein Einzelfall ist, dass eine Kommune im votemanager Geodaten für die GeoGrafik einpflegt, diese Geodaten nicht aber als Open Data zur Verfügung stellt, selbst wenn sie ein Open-Data-Portal hat[6]. In diesen Fällen ist die Nichtbereitstellung überhaupt nicht nachvollziehbar, da die Daten offensichtlich digitalisiert und intern mindestens im GeoJSON-Format vorliegen, es kann also nicht damit begründet werden, dass die Daten zuerst erstellt werden müssten.
Um langfristig nicht bei jeder Wahl und Bezirksänderung unter manueller Arbeit Geodaten erstellen zu müssen, bietet sich die Überlegung an, dort, wo es nicht bereits so geschieht, die Bezirksdefinitionen beispielsweise über das Kataster zu verwalten, und nicht mehr in Text- oder Tabellenform. Daraus wären die anderen Darstellungen wiederum ableitbar. Dass die Nutzung des Liegenschaftskatasters dafür möglich ist, zeigt sich beispielsweise bei Betrachtung der Gemeindewahlbezirke aus Weilerswist in der zuvor erwähnten Web-GIS-Darstellung des Kreises Euskirchen (aufgrund der Datenbeschaffenheit).

Ergänzend zu den Darstellungen im Kapitel 3 (Open Data) ist in Zusammenhang mit Geodatenbereitstellung zu Wahlen das Vorhaben zu erwähnen, kommunale Daten zu Wahl- und Stimmbezirken über INSPIRE-Richtlinie-konforme Dienste bereitzustellen. Zu den dort geforderten Daten gehören nämlich auch die hier relevanten Geodaten. Im Rahmen des Geonetzwerk Metropole Ruhr sollte eine ruhrgebietsweite Lösung bis zum Jahr 2020 etabliert werden [69, Anlagen]. Das Land NRW habe sich zusätzlich für einen Betrieb der Dienste auf Landesebene ausgesprochen [70].
Dass diese Vorhaben bis jetzt umgesetzt wurden, kann anhand von einer Suche in öffentlich zugänglichen Quellen nicht bestätigt werden.

5.2 Datenerstellung

Wenn die Möglichkeit, auf bestehende maschinenlesbare Flächendaten zurückzugreifen, nicht gegeben ist, kann auf eine manuelle Methode der Datenerstellung zurückgegriffen werden.

[5] Beispiel: http://wahlergebnisse.stadt-hagen.de/prod/EW2019/05914000/html5/geografik_145_6_.html

[6] Dies ist beispielsweise in Essen (mit eigenem Open-Data-Portal), Hagen, und Wiesbaden der Fall

Grundsätzlich ist dies durch „Handarbeit“ per Einzeichnung in einem Geoinformationssystem möglich. Dabei kann als Grundlage eine vorliegende Raster-Karte, die die Flächen darstellt, georeferenziert und als Hintergrund verwendet werden (in dem Fall kann von Digitalisierung, engl. *to digitize*, gesprochen werden). Liegen Daten vergangener Wahlen vor, können auch diese eine hilfreiche Grundlage sein, da sich die Handarbeit dann auf Anpassungen beschränkt und die grundsätzliche Einzeichnung sämtlicher Flächenumrisse nicht vollumfänglich nötig wird.
Ist dies nicht der Fall, kommen je nach Beschaffenheit der Daten womöglich maschinelle Methoden wie Vektorisierung[7] oder zumindest eine Unterstützung bei dem manuellen Einzeichnen[8] in Frage.
Ist gar keine geographische Datengrundlage vorhanden, beispielsweise wenn die Wahlbezirkseinteilung einer Gemeinde nur in textlicher Form beschrieben wird, muss eine Einzeichnung völlig frei interpretierend unter Zuhilfenahme beispielsweise einer Hintergrundkarte mit Straßen und Gebäuden geschehen.

Die zuvor genannten Methoden sind ab einem bestimmten Umfang nicht mehr verhältnismäßig anzuwenden. Gerade im urbanen Raum handelt es sich auf der niedrigsten Gebietsebene oftmals um eine dreistellige Anzahl von Flächen, selbst auf der höher liegenden Ebene von Kommunalwahlbezirken mit einer Anzahl von 10 bis 45 [11] kann es Schwierigkeiten aufgrund von straßen- und gebäudegenauen Feinheiten in den Flächen geben.

Folgend wird eine Methode beschrieben, wie anhand von Hausnummerpositionen mit Wahlgebietszuordnung in einem Geoinformationssystem Wahlgebietsflächen generiert werden können.

Die Grundidee basiert auf der Verwendung von Voronoi-Polygonen, die in einem GIS für Punktelayer generiert werden können. Dabei wird für jedes Punktobjekt des Input-Layers ein Polygon generiert, das die dem Punkt „gehörende“ Fläche darstellt. Innerhalb dieser Fläche ist jeder Standort näher am zugehörigen Punktobjekt als zu einem anderen Punktobjekt des Input-Layers [71].
Die Attribute der Punktobjekte werden bei den neuen Polygonen übernommen. Besonders wichtig ist dabei in diesem Fall das Vorhandensein einer Bezirksnummer, auf der Ebene, auf der schließlich generierte Bezirksflächen ermittelt werden sollen.
Nachdem Voronoi-Polygone für den Hausnummern-Punktelayer generiert wurden, können sie anhand des zuvor erwähnten Bezirksattributs zusammengeführt/verschmolzen werden (*dissolve*). Dabei kommen so viele unterschiedliche Flächen heraus, wie es unterschiedliche Werte des Bezirksattributs gibt.
Ein weiterer notwendiger Schritt ist das Verschneiden (*clip*) des resultierenden Polygon-Layers mit der Außengrenze des Gesamtgebietes, also beispielsweise mit der Stadtgrenze, da die Voronoi-Polygone insgesamt zunächst ein Rechteck bilden.
Bei der in diesem Abschnitt folgenden Beschreibung eines erweiterten Ansatzes wird auch die hier beschriebene Methodik in einer Abbildung dargestellt. Außerdem werden dort die Nachteile dieser Generierung erläutert.

[7] vgl. https://gis.stackexchange.com/questions/251360/converting-raster-to-vector-by-generating-center-lines
[8] vgl. https://github.com/mkondratyev85/raster_tracer

Die Herkunft des Input-Layers kann eine Datei sein, in der bereits verarbeitungsfähig nicht nur Hausnummern und ihre Koordinaten enthalten sind, sondern auch die Bezirkszuordnung. Wenn nicht so eine Datei verwendet werden kann, muss eine Zusammenführung der Hausnummerdaten mit (Bezirks-)Zuordnungsdaten geschehen.
Für Hausnummern kann es kommunale oder vom Land bereitgestellte Datensätze geben. Ist dies nicht der Fall, wäre es denkbar, Hausnummern aus OpenStreetMap-Daten zu extrahieren. In jedem Fall ist darauf zu achten, dass je nach Lizenz der verwendeten Daten auch das Resultat unter einer bestimmten Lizenz stehen muss oder Bedingungen beachtet werden müssen.
Für Zuordnungsdaten kann es sein, dass es bereits einen hausnummerscharfen Datensatz gibt, der mit dem geographischen Hausnummerdatensatz verknüpft werden kann (*Join*). Werden die Zuordnungen aber wie im votemanager-csv-Format oder wie in Daten nach dem Offenen Wahldatenstandard nur auf Straßen-/Straßenbereichsebene bereitgestellt, muss eine zusätzliche Verarbeitung stattfinden. Für das votemanager-csv-Format wurde eine solche Vorverarbeitung und Verknüpfung mit einem städtischen geographischen Hauskoordinatendatensatz beispielhaft in Form eines Python-Skripts implementiert, diese befindet sich im `tools/`-Verzeichnis des Projektrepositoriums.

Der zuvor beschriebene Ansatz der Bezirksgenerierung kann um die Idee erweitert werden, dass bei der Erstellung der Flächen (bspw. für die niedrigste Ebene, also Kommunalwahl-**Stimmbezirke**) bestehende Flächendaten einer höheren Ebene (bspw. Kommunalwahl-**Wahlbezirke**) berücksichtigt werden [72]. Die zu berücksichtigenden Flächendaten können beispielsweise aus einem Open-Data-Portal oder aus einer zuvor beschriebenen manuellen Erstellung stammen. Für den Input-Punktlayer wird vorausgesetzt, dass nicht nur die Bezirksnummern, anhand derer die Flächen generiert werden, angegeben werden, sondern auch die Nummer des jeweiligen Bezirks auf der höheren Ebene — diese Werte sollten mit den Schlüsselwerten der zuvor erwähnten Flächen auf der höheren Ebene verknüpfbar sein[9].
Die Grenzen dieser Gebiete einer höheren Ebene gelten dann als gesetzt, und die generierten Bezirke werden jeweils **ausschließlich innerhalb der Grenzen ihres übergeordneten Gebiets generiert**. Dies ist nur sauber möglich, wenn die zu generierenden Bezirke vollkommen innerhalb eines Gebiets auf einer höheren Ebene liegen *sollen*, wie es beispielsweise bei kommunalen Wahlen mit Stimmbezirken innerhalb von Wahlbezirken der Fall ist[10]. Außerdem wird diese vorgeschriebene Tatsache dazu genutzt, Punkte herauszufiltern, die ihren Koordinaten nach im „falschen“ übergeordneten Gebiet liegen — so kommt es zu weniger Vorkommnissen kleiner Exklaven, wenn eine Hauskoordinate abseits ihres eigentlichen Gebiets liegt[11]. Dies hat auch den weiteren Vorteil, dass Hausnummern ohne zugeordneten Wert ebenfalls unberücksichtigt bleiben. Exklaven bei den generierten Gebieten können weiterhin vorkommen, insgesamt werden diese Gebiete aber völlig innerhalb ihres übergeordneten Gebiets liegen.

[9] Um diese Bezirksnummern einer höheren Ebene als Attribute zu ergänzen, kann eine Verknüpfung/Join beispielsweise mit der Wahlgebiets-Datei aus einem OWD-Datensatz durchgeführt werden

[10] Die Stimmbezirke sind die kleinste Flächeneinheit und bieten überhaupt die Grundlage des räumlichen Umfangs der Wahlbezirke. Für Stadtbezirke in kreisfreien Städten ist ebenfalls davon auszugehen.

[11] Im Gegensatz zum als Vorbild verwendeten Algorithmus aus der StackExchange-Antwort [72] wird hier also nicht allein nach Standort des Punkts gefiltert.

Für diesen mehrschrittigen Algorithmus kann ein Modell in dem freien Geoinformationssystem QGIS [73] erstellt werden. Das Python-Framework *Processing* ist ein integrierter Bestandteil von QGIS und ermöglicht die nahtlose Verwendung von Geo-Algorithmen aus verschiedenen zugrundeliegenden Bibliotheken. Für Processing-Algorithmen werden menschenorientierte graphische Oberflächen zur Angabe von Ein-/Ausgaben und weiteren Optionen automatisch generiert. Die Erstellung eigener Algorithmen kann über ein graphisches Modellierungswerkzeug erfolgen, in dem andere Algorithmen eingebunden werden können und ihre Ein- und Ausgaben jeweils wie zur Zielerreichung notwendig verknüpft und optional auch dokumentiert werden. Dadurch kann eine hohe Wiederverwendbarkeit und Automatisierung erreicht werden [74]. Ein wichtiger Hinweis aufgrund möglicher unterschiedlicher Input-Koordinatensysteme ist, dass QGIS grundsätzlich damit umgehen kann — aufgrund der On-The-Fly-Reprojektion ist es nicht notwendig, Datensätze vorher manuell zu reprojizieren. Erstellte Processing-Algorithmen werden in einem XML-Format mit Dateiendung model3 gespeichert. Sie können zur Visualisierung beispielsweise als PDF exportiert werden, oder zur Ausführung als generierter Python-Code.
Der mit dem Modellierungswerkzeug erstellte Algorithmus befindet sich im tools/-Verzeichnis des Projektrepositoriums und wird in Abbildung 5.1 dargestellt.
Die generierte Oberfläche zur Ausführung des Algorithmus wird in Abbildung 5.2 dargestellt.

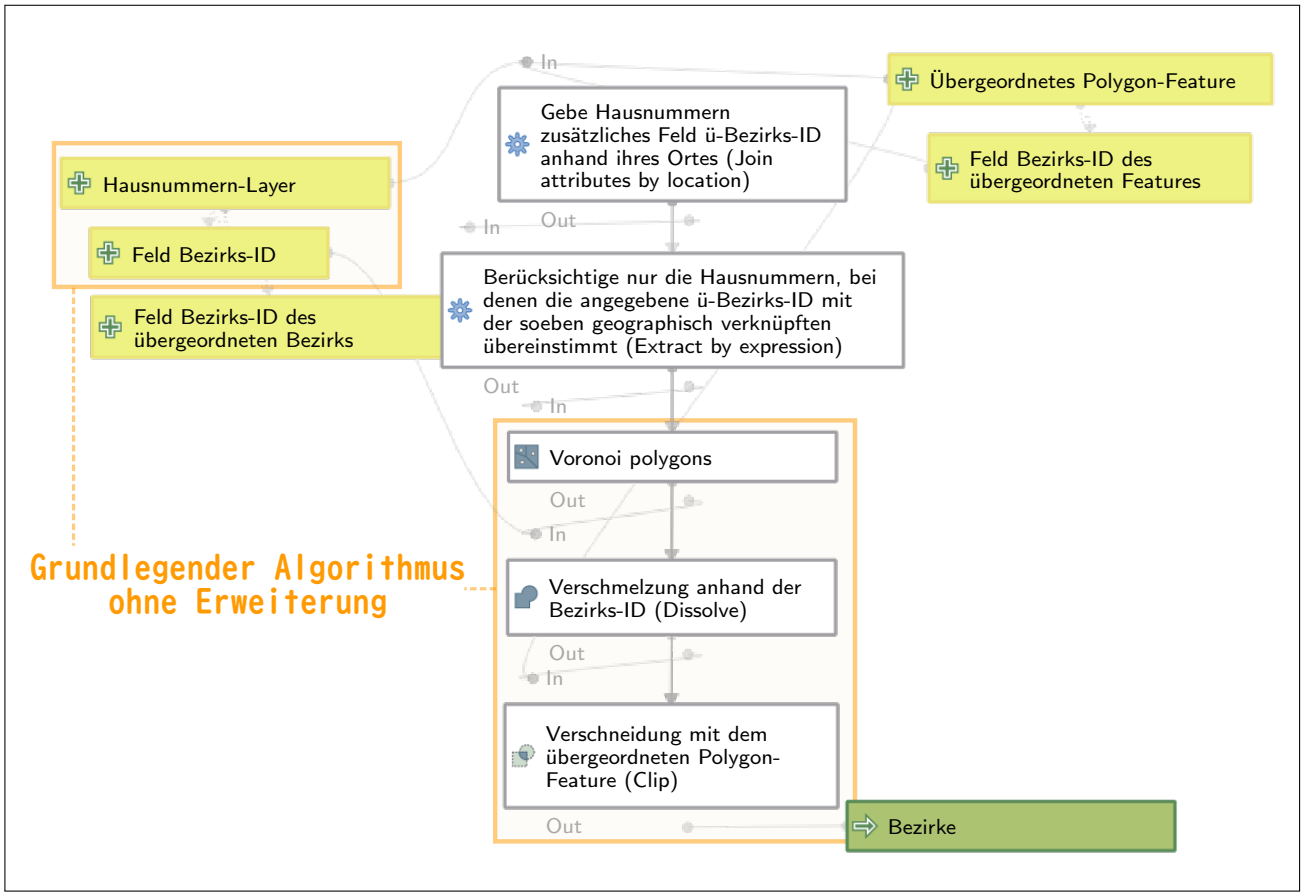

ABBILDUNG 5.1: Exportierte Visualisierung des verwendeten erweiterten Algorithmus. Die Darstellung entspricht der Bearbeitungsoberfläche des Modellerstellungswerkzeugs.

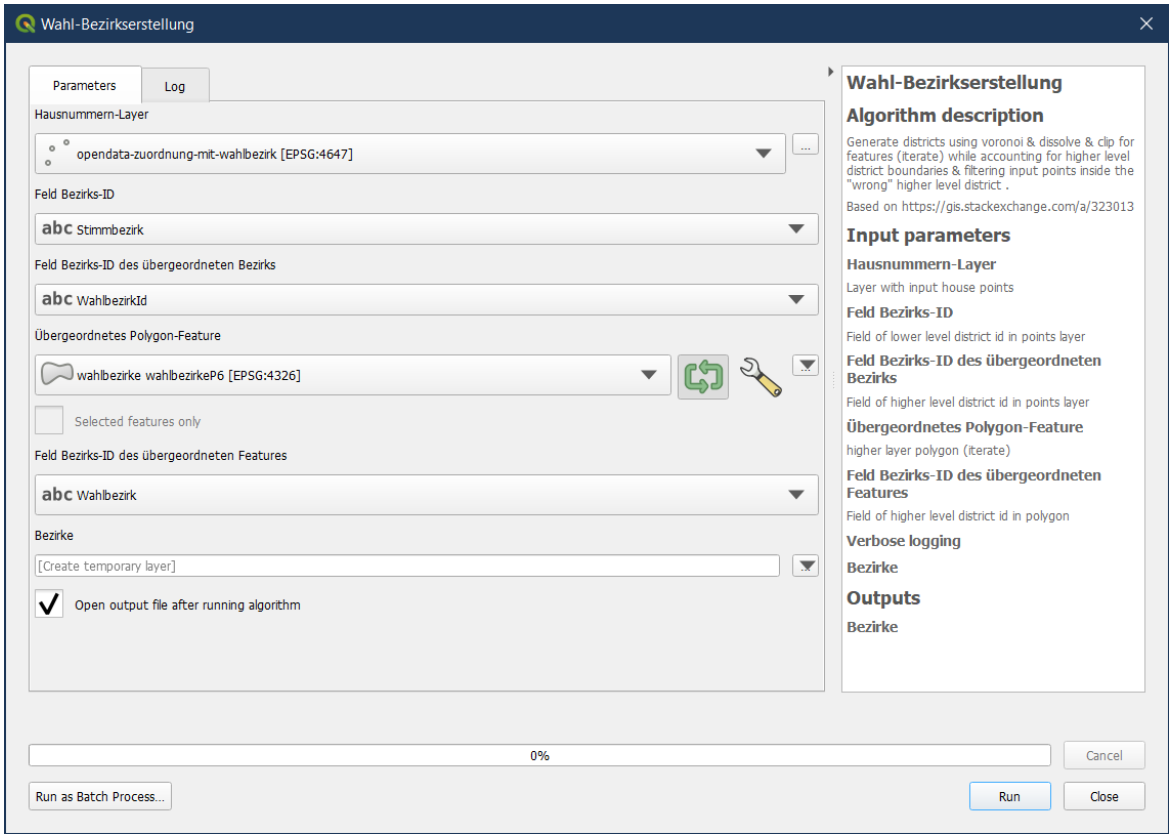

ABBILDUNG 5.2: Aus der angelegten Modelldefinition und Dokumentation generierte graphische Oberfläche zur Ausführung des Algorithmus.
Wichtig ist, dass das Wiederholungssymbol am übergeordneten Polygon-Layer aktiviert wird, damit die Ausführung pro Objekt und nicht für den Input-Layer im Ganzen erfolgt.

Die Ausgabe des Algorithmus sind mehrere Polygon-Layer. Es kommen so viele heraus, wie es Flächen in dem zugrundeliegenden übergeordneten Polygon-Layer gab. Jeder der resultierenden Layer enthält die Flächen der generierten Bezirke, die dem jeweiligen übergeordneten Bezirk zugehörig sind. Diese Layer überschneiden sich nicht geographisch (da die zugrundeliegenden übergeordneten Flächen sich nicht überschneiden sollten) und können in einem weiteren manuellen Schritt mit dem Tool „Merge vector layers“ in QGIS zu einem Layer zusammengeführt werden. Gleichzeitig kann eine Reprojektion erfolgen, um die schließliche Ausgabe direkt im für die weitere Verwendung gewünschten Koordinatensystem zu erhalten. Ein weiterer Schritt kann die Entfernung mancher Felder sein, da die Ergebnisflächen möglicherweise noch Attribute vom Hausnummernlayer besitzen, die nach der Verschmelzung nicht mehr sinnvoll anwendbar sind.

Beispieldarstellungen generierter Stimmbezirke der kreisfreien Stadt Hagen mit Stand Kommunalwahlen 2020 werden in Abbildungen 5.3 und 5.4 dargestellt und erläutert. In diesem Beispiel sind die zugrundeliegenden Wahlbezirksdaten durch Verwendung von Geodaten, die aus einer votemanager-GeoGrafik einer älteren Wahl [75] extrahiert wurden, und manueller Anpassung auf die aktuellen Gegebenheiten entstanden. Da dies ein sehr individueller Vorgang ist, mit manuellen Anpassungen in GIS, wurde darauf zuvor in diesem Kapitel nicht näher eingegangen. Die aktuellen Wahlbezirks-Grenzverläufe wurden zum Vergleich

aus einer manuell zu georeferenzierenden PDF-Datei [76] als Hintergrundlayer eingebunden. Dies diente als Basis der manuellen Anpassungen — stattdessen hätte auch allein damit ein manuelles Vektorisieren stattfinden können. Für die Hausnummern wurde ein einschränkungsfreier städtischer Datensatz verwendet [77], und für die Zuordnungen der klassische csv-Straßen-Datensatz aus votemanager [20].

Erkennbar sind die dargestellten Methoden der automatisierten Flächengenerierung mit GIS mit einem hohen Aufwand verbunden und das Resultat ist ohne weitere Nachbearbeitung nicht in einem verwendbar *aussehenden* Zustand.
Verbessern ließe sich das Ergebnis zunächst durch Aufräumen der Daten, beispielsweise Entfernung von Input-Hausnummern, die abseits stehende und irrelevante Industriegebäude darstellen, ohne die gegebenenfalls sauberere Grenzverläufe herauskommen könnten.
Als zusätzlicher GIS-Schritt könnte die Verwendung eines Generalisierungsalgorithmus zur Glättung vor dem „Clip“-Algorithmus erfolgen. Dabei muss darauf geachtet werden, dass dies unter Beachtung der Topologie erfolgt, beispielsweise indem der Algorithmus v.generalize von GRASS GIS verwendet wird (dies ist bei gemeinsamer Installation von QGIS/Processing aus möglich [78]).
Eine alternative Herangehensweise, die möglicherweise über die Standardfunktionalitäten der GIS oder der vorhandenen Algorithmen hinaus geht, ist die Generierung der Voronoi-Polygone unter Beachtung beliebiger Hindernisse. So könnten beispielsweise natürliche Elemente wie Flüsse oder andere zerschneidende Elemente wie Autobahnen bei der Generierung beachtet werden, dort, wo sie auch wirklich die erwartete Bezirksgrenze bilden. Dies scheint aufgrund der Komplexität allerdings nur dann eine sinnvolle Vorgehensweise zu sein, wenn beispielsweise eine Kommune ihre Bezirksgrenzen offiziell so generieren möchte, und andere Lösungsansätze, wie die wiederholte manuelle Digitalisierung, oder Definition über das Kataster, nicht umgesetzt werden können, so dass der zu erwartende hohe Aufwand für eine Implementierung dieser erweiterten Herangehensweise gerechtfertigt ist.

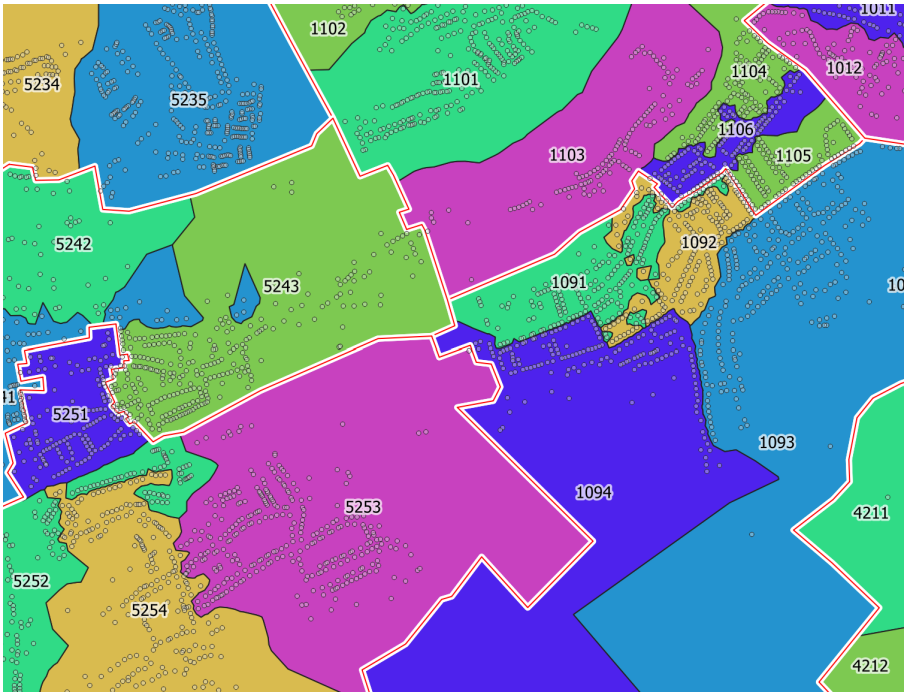

ABBILDUNG 5.3: Beispiel-Überblick über das Resulat im (sub-)urbanen Raum.
Die gegebenen Wahlbezirksgrenzen sind weiß-rot hervorgehoben. Im östlichen Teil ist zu erkennen, dass aufgrund rechtwinklig verlaufender Straßenzüge unterschiedlicher Stimmbezirke diverse Exklaven vorkommen.

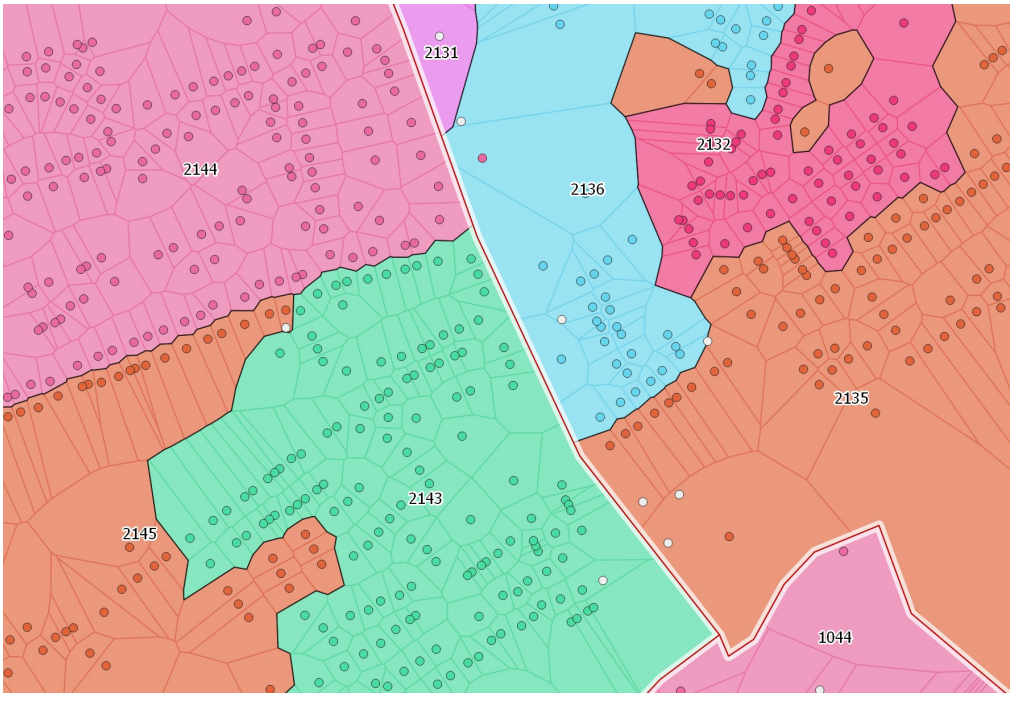

ABBILDUNG 5.4: Nahansicht generierter Stimmbezirke, zugrundeliegender Hausnummern und Voronoi-Polygone in einer passenden Einfärbung.
Hausnummern ohne Zuordnung werden automatisch ignoriert. Die pinke Hausnummer oberhalb der Bildmitte wird auch ignoriert, da diese einem Stimmbezirk im anderen Wahlbezirk zugeordnet ist, nicht aber in der vorgegebenen Fläche dieses Wahlbezirks liegt. Innerhalb eines vorgegebenen Wahlbezirks können Enklaven/Exklaven vorkommen, wie bei Stimmbezirk 2135.

6 Umsetzung

6.1 Technologien

Die zu entwickelnde Anwendung soll eine Web-Anwendung sein, zum Zugang von einem Web-Browser aus. Die Datenverarbeitung soll ebenfalls dort geschehen, ohne ein zur Anwendung gehörendes Backend auf einem Server.
Grundlage für die Anwendung ist JavaScript-Programmcode, der im JavaScript-Engine des Browsers ausgeführt werden soll. Damit moderne Sprachfeatures wie Module und Klassendeklarationen [79] bei der Programmierung eingesetzt werden können, und eine Ausführbarkeit in sämtlichen gängigen Browsern gegeben ist, kommen bei der Entwicklung verschiedene Tools zum Einsatz. Dazu gehört im Kern die Verwendung von Webpack [79, 80], um Code und Ressourcen aus verschiedenen Modulen und auch von Abhängigkeiten zusammenzufassen und daraus ermittelten, im Browser einbindbaren Code, bereitzustellen. Dabei können auch Umwandlungen und Optimierungen durchgeführt werden, wie das Transpiling von Code für die Kompatibilität mit einem älteren Sprachstandard und die Minifizierung von Code für kompaktere Dateien. Dazu werden durch Webpack weitere Tools wie Babel und Terser eingesetzt. In Zusammenhang mit Caching bietet Webpack auch die Möglichkeit, Hashes in Dateinamen zu inkludieren und dadurch dafür zu sorgen, dass (nur) bei Code-Änderungen keine veralteten Dateiversionen von Dateien von Clients aus ihren Caches geladen werden. Tiefergehend ist auch Code Splitting ein bemerkenswertes Feature von Webpack, womit Teile des Codes separiert werden können und auch ein Laden bei Bedarf ermöglicht wird [80].
Eine wichtige Datei im Projektverzeichnis ist package.json, die durch den verwendeten Node Package Manager unter anderem zur Verwaltung von Abhängigkeiten und ihrer Versionen verwendet wird [81]. In dieser werden auch ausführbare Kommandos aufgeführt, mit denen häufig verwendete Aufrufe abgekürzt werden können. Für das Bundling wird dabei `webpack --config webpack.config.prod.js` aufgerufen, für das Starten eines Entwicklungsservers `webpack serve --config webpack.config.dev.js`.

Im Rahmen der Entwicklung werden verschiedene optionale Tools eingesetzt.
Flow ermöglicht optionale Typ-Annotationen im Code, die nicht nur der Veranschaulichung dienen, sondern auch für das Finden von Fehlern und nicht vorgesehenen Verwendungen verwendet werden können [82]. Die Typ-Annotationen werden im Build-Prozess durch eine Babel-Erweiterung automatisiert entfernt. JSDoc [83] ermöglicht die Dokumentation vieler Code-Bestandteile durch besonders formatierte Kommentarblöcke. Für die zu entwickelnde Anwendung werden Kommentare in englischer Sprache formuliert.
Als Entwicklungsumgebung wird Visual Studio Code eingesetzt, welches um Plugins ergänzt werden kann, darunter beispielsweise die Einbindung vom Flow-Entwicklungsserver oder von ESLint-Fehlermeldungen. ESLint [84] wird eingesetzt, um Probleme in Bezug zum Code-Stil zu finden, anhand einer anpassbaren Konfiguration.

Eine wesentliche verwendete Bibliothek ist Leaflet, welches für die Kartendarstellung eingesetzt wird. Leaflet-Karten werden in Form einer interaktiven (zoom- und bewegbaren) und touchfähigen Ansicht auf Webseiten eingebunden. Es können sogenannte Controls in Ecken der Kartenansicht eingebunden werden, um Steuerungselemente oder Informationen bereitzustellen (vgl. 6.4.1) [64].
Für die vektorbasiert erfolgende Darstellung der Hintergrundkarte und der Bezirksflächen wird die Bibliothek Tangram verwendet, diese stellt ein Leaflet-Plugin bereit (vgl. 6.4.2.1).

Zu den weiteren nennenswerten Abhängigkeiten gehören LitElement und Weightless. LitElement wird eingesetzt, um eigene wiederverwendbare HTML-Elementklassen gemäß des Web Components Standards zu erstellen [85]. Weightless ist eine Bibliothek, die auf Basis von LitElement eine Menge an Elementen zur Einbindung in Oberflächen bereitstellt, darunter Auswahlfelder, Tableisten, und Dialoge [86].

In Zusammenhang mit der eingefärbten Kartendarstellung kommen chroma.js [87] für Farbberechnungen und geostats.js [88] für statistische Klasseneinteilung zum Einsatz. In Abschnitt 6.4.2.3 wird deren Verwendung näher beschrieben.

6.2 Anwendungsstruktur

Die Ausführung der Anwendung erfolgt über den Aufruf der Hauptseite index.html, die durch Webpack (genauer gesagt dessen html-webpack-plugin) anhand des Templates src/index-template.ejs generiert wird. Skripte und Stile werden automatisch im <**head**>-Element aufgeführt, und das <**body**>-Element enthält, wie im Template vorgegeben, nur ein Element: <**div** id="map"></**div**>.
Die folgenden Erläuterungen beziehen sich auf die grundlegenden Bestandteile der Anwendung, unabhängig vom Webpack-Bundling — Verzeichnisse und Dateien, die genannt werden, befinden sich im Verzeichnis src/ des Projektrepositoriums, darunter sämtlicher JavaScript-Code innerhalb des Verzeichnisses src/js/.

Im Einstiegspunkt app.js wird zunächst ein Leaflet-`Map`-Objekt erstellt, dabei wird die `id` des als Karten-Container zu verwendenden HTML-Elements angegeben, also `'map'`. Auf die Layerdefinition wird im Abschnitt 6.4.2.1 eingegangen.
Nach der grundlegenden Einrichtung der Leaflet-Karte wird in app.js ein neues Objekt der Klasse `WahlController` (aus wahl-controller.js) erstellt. Dieses Objekt verwaltet alle Objekte, die notwendige Daten vorhalten, und es ist die Grundlage für den Zustand der Web-Anwendung mit den zum jeweiligen Zeitpunkt auswählbaren und ausgewählten Objekten, wie in den folgenden Absätzen beschrieben wird.

Der Konstruktor von `WahlController` bekommt nicht nur das Karten- und Layerobjekt übergeben, sondern auch ein Konfigurationsobjekt `wahlenConfig` (aus config.js).
In dieser Konfiguration wird ein `Array` aus Wahltermine beschreibenden Objekten definiert. Zu jedem Wahltermin werden Name und Datum zur Darstellung, eine Basis-URL für die Datenladung, und Zoomlevel/Koordinaten für ein automatisches Einrichten des Kartenausschnitts angegeben.
Außerdem wird zu jeder Wahl des Wahltermins ein eine Wahl beschreibendes Objekt in

einem `Array` aufgeführt. Zu den Angaben jeder Wahl gehört ein Name zur Darstellung, der formelle `wahl-name`, und Pfadangaben zu sämtlichen zu ladenden csv-Dateien gemäß des Offenen Wahldatenstandards im aktuellen Zustand. Weitere benötigte Angaben werden im nächsten Abschnitt erläutert. In Listing 6.1 ist eine Wahlterminkonfiguration dargestellt.

```
let wahlTerminKrEuskirchenKWahl: WahlTerminConfigType = {
    name: "Kreis Euskirchen 2020",
    baseUrl: "./data/kreis-euskirchen-2020/",
    wahlDatumStr: "13.09.2020",
    defaultCenter: [50.565, 6.5],
    defaultZoom: 10,
    wahlen: [
        {
            displayName: "Kreistagswahl",
            name: "Kreistagswahl NRW",
            parameterPath:
    "05366000_20200913_Kreistagswahl-NRW_Wahlparameter_V0-2_20200913T111111.csv",
            gebietePath:
    "05366000_20200913_Kreistagswahl-NRW_Wahlgebietseinteilungen_V0-2_20200930T111111.csv",
            stimmzettelPath:
    "05366000_20200913_Kreistagswahl-NRW_Stimmzettel_V0-2_20200930T111111.csv",
            kandidatPath:
    "05366000_20200913_Kreistagswahl-NRW_Kandidaten_V0-2_20200930T111111.csv",
            ergebnisPath:
    "05366000_20200913_Kreistagswahl-NRW_Wahlergebnisse_V0-2_20200930T111111.csv",
            ergebnisType: ErgebnisKommunalwahlNRW,
            ebenen: new Map([
                ["Stimmbezirk", {
                    geoJson: undefined,
                    keyProp: undefined,
                    gsProp: undefined,
                    uniqueId: false
                }],
                ["Wahlbezirk", {
                    geoJson: "wahlbezirke.geojson",
                    keyProp: "strnr",
                    gsProp: "ags",
                    uniqueId: false
                }],
                ["Kreiswahlbezirk", {
                    geoJson: "kreiswahlbezirke.geojson",
                    keyProp: "padnr",
                    uniqueId: true
                }],
            ])
        },
        ...
    ]
}
```

LISTING 6.1: Wahltermin-Konfigurationsobjekt für die Wahlen auf Kreisebene im Kreis Euskirchen am 13. September 2020. An dieser Stelle ist als einzige Wahl die Kreistagswahl abgebildet.

Im Konstruktor des `WahlController` werden Steuerungselemente (Controls) zur Leaflet-Karte hinzugefügt. Weiteres dazu wird im Abschnitt 6.4.1 behandelt.

Dann erfolgt in der Methode `_initializeWahlTermin(wahlTerminConfig)` für den standardmäßig ausgewählten ersten definierten Wahltermin die Erstellung von Objekten der Klasse `Wahl` (aus wahl-lib/wahl.js), die bei der Verarbeitung von Wahldaten von zentraler Bedeutung ist. Im nächsten Abschnitt wird dieser grundlegende, von der Visualisierung unabhängige, Bestandteil der Anwendung behandelt.
Das Laden von Daten einer `Wahl` wird durch den `WahlController` dann ausgelöst, wenn dessen Eigenschaft `activeWahl` geändert wird. Standardmäßig wird eine `Wahl` vorausgewählt, wenn der Wahltermin nur eine einzige Wahl definiert, ansonsten geschieht die Auswahl und Änderung der aktiven Wahl durch eine entsprechende Aktion in der Anwendungsoberfläche.

Abbildung 6.1 stellt die Anwendungsstruktur dar. In den nächsten Abschnitten werden die erstellten Module und Klassen jeweils detaillierter beschrieben.

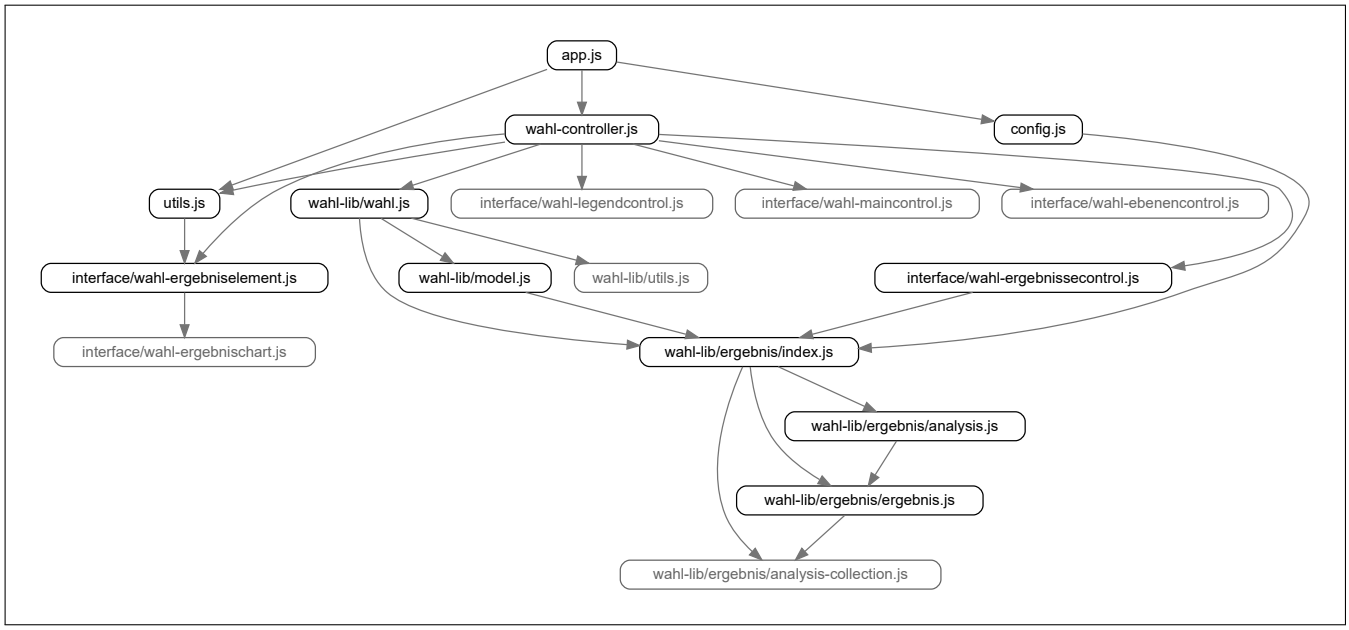

ABBILDUNG 6.1: Module und Abhängigkeiten innerhalb des Verzeichnisses src/js/

6.3 Verarbeitung von Wahldaten

Die `Wahl`-Klasse ist der Kern der Wahldatenverarbeitung. `Wahl`-Objekte halten sämtliche Daten verschiedener Klassen zu einer Wahl vor.
Im folgenden Abschnitt Modellklassen werden zunächst die Klassen beschrieben, die sich unmittelbar auf Datensätze der Dateitypen des Offenen Wahldatenstandards beziehen.
Im Abschnitt Wahl-Klassen wird die `Wahl`-Klasse mit ihren Zuständigkeiten beschrieben, sowie weitere Klassen, die notwendig sind, um mit Daten des Offenen Wahldatenstandards besser umgehen zu können.
In einem weiteren Abschnitt wird die Ergebnisverarbeitung besonders behandelt.

6.3.1 Modellklassen

Alle Modellklassen werden in wahl-lib/model.js definiert und sind von einer abstrakten Basisklasse `WahlModel` abgeleitet.
Die wesentliche Eigenschaft der Modellklassen ist das statische `Array` `params`, dessen Einträge vom Typ `ParamType` sind. Ein Eintrag kann also entweder ein `String` oder ein `Array` aus einer `String` und einem weiteren Wert, der als Standardwert verwendet wird, sein. Die in jedem Fall erfolgende `String`-Angabe ist die Bezeichnung einer Eigenschaft. Das `params`-`Array` jeder abgeleiteten Klasse enthält nur die jeweiligen zusätzlichen Eigenschaften, alle weiteren werden aufgrund ihrer Angaben in der Elternklasse[1] berücksichtigt (vgl. statische dynamische Eigenschaft `allParams`). Die vier grundsätzlich vorkommenden Eigenschaften `["version", "wahl-behoerde-gs", "wahl-datum", "wahl-name"]` werden bereits in `WahlModel` definiert.

In dem aus den abgeleiteten Modellklassen heraus verwendeten Konstruktor von `WahlModel` werden die in den `Array`s erfolgten Parameterdefinitionen verwendet und mit Werten aus einem dem Konstruktor übergebenen `Object` befüllt. Das dem Konstruktor übergebene `Object` enthält alle vorhandenen Schlüssel und Werte, die üblicherweise aus der Auslesung der OffeneWahlDaten-csv-Datei und Umwandlung in eine JSON-Abbildung in Form des übergebenen `Object` stammen.
Werden Eigenschaften angegeben, die in der Modellklasse nicht definiert sind, wird ein Fehler geworfen. Eine Ausnahme sind Felder, die in unterschiedlichen Varianten oder als Präfix vorkommen dürfen, wie es bei den Ergebnisdateien der Fall ist. Diese Feldbezeichnungen starten üblicherweise mit einem Großbuchstaben, dem gegebenenfalls weitere Angaben wie eine Zahl folgen. Die Angabe dieser erlaubten Ausnahmen von unbekannten Feldbezeichnungen erfolgt in Form von `String`-Objekten im statischen `Array` `allowAnyFieldStarts` einer Modellklasse.
Als letztes werden im Konstruktor weitere Überprüfungen durchgeführt, insbesondere bezüglich gültiger Werte bestimmter Eigenschaften. Die Überprüfungen beziehen sich dabei in jedem Fall auf das vorliegende Objekt. Überprüfungen, die sich auf eine Menge an Objekten in ihrem Zusammenspiel beziehen, erfolgen in der `Wahl`-Klasse.

Im Normalfall sieht das erstellte Modellobjekt ähnlich aus wie das ursprünglich angegebene `Object`, nur dass gegebenenfalls nicht in der Eingabe vorhandene Felder auch definiert sind, einige Fehler ausgeschlossen sind, und einige helfende dynamische Eigenschaften oder Methoden enthalten sind — aufgrund ihrer Definitionen in den jeweiligen Modellklassen.
Damit diese dynamischen Eigenschaften funktionieren können, muss dem Konstruktor der Modellobjekte das `Wahl`-Objekt mitgegeben werden. Hier ist der Grundgedanke, dass eine weitergehende Datenspeicherung (beispielsweise Referenzen auf andere erstellte Objekte) nicht in den Modellklassen erfolgen soll und die Ermittlung zugehöriger Objekte nur über dynamische Eigenschaften oder Methoden unter Verwendung des zugewiesenen `Wahl`-Objekts erfolgt.

[1] Dies funktioniert nicht nur mit der Elternklasse `WahlModel`, man kann auch eine davon vererbende Klasse vererben

Diese Lösung der Verwendung einer solchen Klassenstruktur ermöglicht eine Definition der Modellklassen mit niedrigem Aufwand. Die Erkennung grundlegender Fehler in den übergebenen Daten erweist sich im Umgang mit den bisher real bereitgestellten Datensätzen als sehr nützlich, da es häufig schwerwiegende Fehler in Feldbezeichnungen oder Werten geben kann, die nunmehr frühzeitig erkannt werden können.
Ein Nachteil dieser Lösung ist, dass die Eigenschaften nur dynamisch an den erstellten Objekten definiert werden und somit Tools wie Flow zum jetzigen Zeitpunkt erfahrungsgemäß nur schlecht damit umgehen können. Um dies in Zukunft zu verbessern, beispielsweise sobald der Wahldatenstandard in einem verbesserten und stabilen Zustand ist, kann es in Frage kommen, die Definitionen zukünftig mit weniger kompakten, unmittelbar angegebenen Klasseneigenschaften inklusive Typ-Angaben zu machen.

6.3.2 Wahl-Klassen

Der wichtigste Parameter des Konstruktors der `Wahl`-Klasse ist das Konfigurationsobjekt für die zu abzubildende Wahl.
Die dort angegebenen Pfade werden für das Laden der Daten verwendet, welches mit der asynchronen Methode `loadData` ausgelöst werden kann. Für das Laden verschiedener Dateiarten werden jeweils eigene asynchrone Funktionen aufgerufen, dies geschieht in einer bestimmten Reihenfolge synchron, da Überprüfungen stattfinden, die auf zuvor zu ladenden Daten basieren. Es wird die asynchrone Hilfsfunktion `fetchCsvToJson` verwendet, die die Datei über den angegebenen Pfad lädt. Dabei wird mit die Bibliothek `csvtojson` [89] verwendet, die für einen csv-Input ein `Array` mit `Object`-Elementen, je einer Zeile entsprechend, zurückgibt. Die Schlüssel in diesen Objekten entsprechen den jeweiligen Feldbezeichnungen aus dem Header der csv-Datei.

6.3.2.1 Parameter und Ebenen

Zunächst werden die Parameter-Daten geladen. Für das zugrundeliegende `Object` wird, wie bei allen weiteren geladenen Daten, ein Objekt der entsprechenden Modellklasse, in diesem Fall `WahlParameter`, erstellt. Daraufhin erfolgt die Erstellung von Objekten der Klasse `Ebene` anhand der grundlegenden Konfiguration aus der Parameterdatei.
Die Klasse `Ebene` von der eingebauten `Map`-Klasse abgeleitet. Ihre Eigenschaften sind die Nummer (gemäß des Offenen Wahldatenstandards sind das üblicherweise Werte von 1 bis 5), Bezeichnung (aus der Parameterdefinition), und ein eigenes Ebenenkonfigurationsobjekt, welches aus einer `Map` in der Wahlkonfiguration stammt. Dies ist zurzeit aus dem Grund notwendig, dass, wie im Abschnitt Annahmen zu Gebieten (4.3.3) beschrieben, nicht klar ist, wann eine Gebietsnummer (egal, auf welcher Ebene) alleine oder nur in Verbindung mit dem Gemeindeschlüssel eindeutig ist. Dies wird, abhängig von der manuell zu supplementierenden Angabe `uniqueId` in der Konfiguration, unterschiedlich behandelt. Weil auch diese Angaben unabhängig vom Offenen Wahldatenstandard gemacht werden müssen, werden auch der Pfad zu einer GeoJSON-Datei der jeweiligen Ebene und die Bezeichnungen von ID- und ggf. Gemeindeschlüsselfeldern der Flächenobjekte in der Ebenenkonfiguration angegeben.

Aufgrund der internen unterschiedlichen Behandlung im Falle `uniqueId` sollten zum Speichern oder Auslesen nicht die Standardmethoden einer `Map` verwendet werden, sondern die eigens definierten Methoden wie `getGebiet`, `addGebiet/addWahlGebiet` (vgl. 6.3.2.2), oder `flat`. Letztere ist als dynamische Eigenschaft, also als Getter, ohne Parameter implementiert und gibt unabhängig von der zugrundeliegenden Speicherungsart immer ein flaches `Array` aller Gebiete der Ebene zurück.

6.3.2.2 Gebiete

Bei der nach der Parameterdatei und damit nach der Ebenendefinition stattfindenden Ladung der Wahlgebietsdaten wird je Gebiet (Modellklasse `WahlGebiet`) die Methode `addWahlGebiet` des `Wahl`-Objekts aufgerufen. Anhand der Eigenschaften des `WahlGebiet`-Objekts werden für die aggregierenden Ebenen `Gebiet`-Objekte in dem jeweiligen `Ebene`-Objekt mit der Methode `addGebiet` erstellt (bei dem ersten Antreffen der jeweiligen Gebietsnummer). Die Klasse `Gebiet` ist von `Map` abgeleitet, ihre Objekte stellen „virtuelle“ Gebiete einer höheren Ebene als Aggregation von Gebieten niedrigster Ebene, also `WahlGebiet`-Objekten, dar. Die `WahlGebiet`-Objekte werden mit ihrer ID als Schlüssel in der `Map`, also im `Gebiet`, gespeichert.
Weiterhin im Schleifendurchlauf für die aggregierenden Ebenen, wird dem erstellten oder zuvor vorhandenen `Gebiet`-Objekt das `WahlGebiet` hinzugefügt (mit der Standardmethode `set` von `Map`). Für Gebiete niedrigster Ebene, also der Ebene mit Nummer 1, werden die einfachen `WahlGebiet`-Objekte an sich direkt verwendet und mit der Methode `addWahlGebiet` der `Ebene` hinzugefügt.
Da es für die unmittelbar abgebildeten Gebiete niedrigster Ebene und für aggregierende Gebiete unterschiedliche Klassen gibt, implementieren sie beide das Interface `GebietInterface` (als Funktionalität von Flow hat dies keinen formellen Charakter). So kann es an Stellen in Anwendungen, an denen es irrelevant ist, zu einer Gleichbehandlung unabhängig von der zugrundeliegenden Klasse kommen.

6.3.2.3 Stimmzettel und Kandidaturen

Daten zu Stimmzetteleinträgen (`WahlStimmzettelPartei`) und kandidierenden Personen (`WahlKandidat`) werden ähnlich zueinander behandelt und zunächst gemeinsam erläutert: Die jeweilige `"stimmzettel-gebiet-nr"` bzw. `"kandidat-gebiet-nr"` wird als Grundlage verwendet, um Objekte zu erstellen, die alle Stimmzetteleinträge/Kandidaturen eines jeweiligen „Gebiets“ vorhalten — sie entsprechen den in den Grundlagenkapiteln erwähnten Stimmzettelgebieten (Klasse `Stimmzettel`) beziehungsweise Kandidaturgebieten (Klasse `GebietKandidaturen`). Beide Klassen sind von `Map` abgeleitet. Die Objekte dieser Klassen werden bei dem ersten Antreffen der jeweiligen Gebietsnummer erstellt.
Wie bereits unter Annahmen zu Gebieten beschrieben, haben diese hier beschriebenen Gebiete und ihre Nummern oder Bezeichnungen keinen Bezug zu den tatsächlichen `WahlGebiet`- oder den aggregierenden `Gebiet`-Objekten, obwohl sie oft übereinstimmen. Ist es gewünscht, alle relevanten Stimmzettel- oder Kandidaturgebiete eines `Gebiet`-Objekts zu ermitteln, kann die Hilfseigenschaft `stimmzettelNrs` oder `gebietKandidaturenNrs` des

`GebietInterface` verwendet werden, um `Sets` mit allen zu beachtenden `"...-gebiet-nr"`-Angaben zu erhalten.

Nach der gemeinsamen konzeptuellen Erläuterung folgen Besonderheiten von Stimmzetteln und Kandidaturen separat:

Es soll mindestens 1 `Stimmzettel`-Objekt pro `Wahl` geben. Ist es nur eines, kann die ID dieses Stimmzettels (gemeint ist die `"stimmzettel-gebiet-nr"`) auch **`undefined`** sein, da dies eindeutig ist und keine weitere Unterscheidung erfolgen muss.
Mit der Methode `addPartei` wird ein `WahlStimmzettelPartei`-Objekt, also ein Stimmzetteleintrag, dem `Stimmzettel` hinzugefügt — es wird mit der numerischen Angabe der Stimmzettelposition als Schlüssel in der zugrundeliegenden `Map` hinzugefügt.

In Zusammenhang mit der Klasse `Stimmzettel` ist die Klasse `Partei` relevant. Sie basiert nicht auf dem Modell des Offenen Wahldatenstandards und kommt als unter Datenmodell (Abschnitt 4.3.4) erwähnte Problemumgehung zum Einsatz, um tatsächlich mit „Parteien“ arbeiten zu können — nicht allein mit den Stimmzetteleinträgen — besonders wichtig beispielsweise bei Bezirksvertretungswahlen aufgrund mehrerer `Stimmzettel` und mehrfach vorkommender Parteiangaben.
`Partei`-Objekte können über die dynamische Eigenschaft `Wahl.parteien` ermittelt werden. Sie werden nicht als stetige Datenspeicherung verwendet, da sie nur aus den bereits gespeicherten `WahlStimmzettelPartei`-Objekten in allen `Stimmzettel`-Objekten ermittelt werden und keine redundante Datenspeicherung erfolgen sollte. Die Klasse `Partei` ist von `Map` abgeleitet, es werden dabei als Schlüssel `Stimmzettel` und als Wert das jeweilige `WahlStimmzettelPartei`-Objekt verwendet — dies ist aber bei der Nutzung der Objekte weniger relevant als bei der initialen Herleitung des jeweiligen `Partei`-Objektes. Die Methode `Map.set` wird so überschrieben, dass weitere Stimmzetteleinträge einer „Partei“ nur hinzugefügt werden können, wenn ihre Angaben (Kurz-/Langname, Farbe, Typ) mit den bisher in dieser `Partei`-Map vorhandenen übereinstimmen. Uneinheitlichkeiten bei den Eigenschaften ein und derselben „Partei“ fallen so auf und werden nicht zugelassen.
Ist die `Partei`-Map vollständig befüllt worden, lassen sich die Eigenschaften nunmehr über sie holen und nicht mehr anhand der redundanten „Partei“/Stimmzetteleinträge. Es sind also quasi „wahlweit“ verwendbare „Partei“-Objekte, die bislang formell im Offenen Wahldatenstandard fehlen.

Liegen Daten zu Kandidaturen vor, wird das `Wahl`-Objekt mindestens 1 `GebietKandidaturen`-Objekt haben. Wenn es keine unterschiedlichen Kandidaturgebiete gibt, hat das eine Objekt als Kandidaturgebiets-ID (gemeint ist `GebietKandidaturen.nr`, `"kandidat-gebiet-nr"`) den Wert **`undefined`**. Dies ist beispielsweise bei Stadtoberhauptswahlen der Fall, wo keine Unterscheidung nach Gebiet oder Listenplatz stattfindet. Es kann auch bei Wahlen mit Listen sein, dass nur ein Gebiet existiert, wie bei der Wahl der Verbandsversammlung des Regionalverbands Ruhr, bei der überall die gleichen Listen wählbar sind. Bezirksvertretungswahlen sind daran zu erkennen, dass es unterschiedliche Kandidaturgebiete gibt, innerhalb derer mehrere Personen von ein und derselben „Partei“ zur Wahl stehen — mit unterschiedlichen Listenplätzen. Bei Ratswahlen ist es je Gebiet und „Partei“ nur eine Person. Diese kann aber auch einen Listenplatz haben. Personen, die nur auf einer Liste kandidieren und

nicht in einem Gebiet aufgestellt sind, werden in einem GebietKandidaturen-Objekt mit Gebiets-ID undefined berücksichtigt.
Mit der Methode GebietKandidaturen.addKandidat wird ein WahlKandidat-Objekt hinzugefügt. In der zugrundeliegenden Map werden die Kandidaturen nicht unmittelbar abgespeichert, sondern unterschieden nach deren „Partei“ in Objekten der von Map abgeleiteten Klasse GebietKandidaturenParteiMap. Darin werden die Kandidierenden, die einen Listenplatz haben, mit diesem numerischen Wert als Schlüssel gespeichert. Die Kandidierenden der jeweiligen „Partei“, die keinen Listenplatz haben, werden in einem Set als Eigenschaft noListKandidaten gespeichert.
Die Erstellung der GebietKandidaturenParteiMap erfolgt ausschließlich intern von einem GebietKandidaturen-Objekt aus. Zur dort stattfindenden Speicherung eines WahlKandidat-Objekts wird die Methode GebietKandidaturenParteiMap.addKandidat verwendet, die die Unterscheidung nach Listenplatz berücksichtigt. Um unabhängig von dieser vom Vorhandensein der Listenplatzangabe abhängigen Speicherung ein Set aller WahlKandidat-Objekte dieser GebietKandidaturenParteiMap zu erhalten, kann die dynamische Eigenschaft allKandidaten verwendet werden.

Da die soeben beschriebene Speicherung grundsätzlich über Gebiete, und darin nach „Partei“, strukturiert wird, muss eine zusätzliche Lösung geschaffen werden, um alle Kandidierenden einer „Partei“ ermitteln zu können, beispielsweise um vollständige Listen darstellen zu können, unabhängig von der Situation in einem bestimmten Gebiet. Dafür kann die Methode kandidaten der in Zusammenhang mit Stimmzetteln beschriebenen Hilfsklasse Partei verwendet werden, die ein Set aller WahlKandidat-Objekte der jeweiligen Partei zurückgibt. Um den Umfang beispielsweise bei einer Bezirksvertretungswahl auf einen Stadtbezirk einzuschränken, kann für den Parameter gebietKandidaturenNrs ein Set aus zu berücksichtigenden Kandidaturgebiets-IDs angegeben werden.

Abschließend ist zu Kandidaturdaten festzuhalten, dass sie wie beschrieben verarbeitbar sind, in Zusammenhang mit der zu entwickelnden geographischen Darstellung aber eine untergeordnete Rolle spielen. Zurzeit erfolgt die über das bis hier beschriebene hinausgehende Datenverarbeitung zur Darstellung der Kandidaturen im Wesentlichen interfaceseitig. Dort werden gegebenenfalls relevante Filterungen aus der Anwendung berücksichtigt. Außerdem findet dort eine wahlweite Erkennung der Tatsache, ob es sich um eine Wahl handelt, die wie eine Bezirksvertretungswahl strukturiert ist, statt, da dies Auswirkungen auf die Darstellung haben kann.Diese Situation ist im Rahmen der Weiterentwicklung nach der Bachelorarbeit zu verbessern, so dass die Funktionalitäten, auch unter Berücksichtigung von Filtermöglichkeiten und weiteren Parametern, innerhalb des Moduls wahl-lib bereitgestellt werden.

6.3.2.4 Ergebnisdaten

Ergebnisdaten (Modellklasse WahlErgebnis) werden in einer Map gespeichert, die jedem definierten WahlGebiet ein optionales anhand des WahlErgebnis-Objekts erstelltes Ergebnis-Objekt zuordnet. Optional deswegen, weil es auch möglich ist, dass für ein WahlGebiet

(noch) kein Ergebnis vorliegt. Die abstrakte Klasse `Ergebnis` wird im folgenden Unterabschnitt Ergebnisverarbeitung beschrieben. Welche der davon abgeleiteten Klassen bei dieser `Wahl` für alle Ergebnisse verwendet wird, wird über die Angabe `ergebnisType` in der Wahlkonfiguration definiert.

6.3.2.5 Abschließendes

Bei der beschriebenen Hinzufügung verschiedener Modellobjekte finden verschiedene Plausibilitätsüberprüfungen statt, beispielsweise ob die bei einer Kandidatur angegebene Partei zuvor in einem Stimmzetteleintrag definiert wurde, oder, ob der Bezirk eines Wahlergebnisses zuvor definiert wurde. In den zusätzlichen Klassen neben der `Wahl`-Klasse erfolgen weitere Überprüfungen, um beispielsweise Duplikate zu erkennen.

Nach einer erfolgten oder fehlgeschlagenen Datenladung werden entsprechende Callback-Funktionen aufgerufen, wenn sie dem `Wahl`-Konstruktor übergeben wurden. So gibt es im Untermodul wahl-lib/ keine oberflächenbezogenen Abhängigkeiten, da entsprechende Dialoge o. ä. von außen definiert in den Callbackfunktionen gehandhabt werden.

6.3.3 Ergebnisverarbeitung

Die Ergebnisverarbeitung ist aufgrund des Umfangs notwendiger Berechnungen und Aggregierungen auf mehrere Dateien aufgeteilt und wird in einem gemeinsamen Untermodul im Verzeichnis wahl-lib/ergebnis/ implementiert.
Folgende Klassen sind von wesentlicher Bedeutung:

Ergebnis (in ergebnis.js)
: Subklassen dieser abstrakten Klasse definieren mögliche Ergebnisfelder.
Objekte verwenden je 1 `WahlErgebnis`, stellen auslesbare Eigenschaften bereit.

ErgebnisAnalysis (in analysis.js)
: Zusammenfassung mehrerer `Ergebnis`-Objekte für aggregierte Werte, beispielsweise für Gesamt- oder Gebietsergebnisse.

ErgebnisAnalysisCollection (in analysis-collection.js)
: Sammlung mehrerer `ErgebnisAnalysis`-Objekte, die jeweils ein Gebiet mit dessen Gebietsergebnis abbilden, zum Vergleich der Ergebnisse verschiedener Gebiete.

6.3.3.1 Ergebnis

Objekte von Klassen, die von der abstrakten Klasse `Ergebnis` abgeleitet sind, stellen exakt 1 `WahlErgebnis`, also ein Ergebnis in einem Bezirk niedrigster Ebene (`WahlGebiet`), dar.
In den `Ergebnis`-Subklassen wird grundlegend definiert, welche Datenfelder im `Wahl` `Ergebnis` erwartet werden, und wie sie verwendet werden, um beispielsweise Stimmenzahlen je nach Wahlvorschlag über Eigenschaften zu verarbeiten.
Grundlegend werden in der statischen Eigenschaft `constantProperties` die Felder der zu

erwartenden `WahlErgebnis`-Objekte angegeben (mit dem zu verwendenden Eigenschaftsnamen und mit dem Eigenschaftsnamen aus dem `WahlErgebnis`-Objekt, also schließlich aus der csv-Datei). Diese Eigenschaften werden durch den `Ergebnis`-Konstruktor in den Objekten der `Ergebnis`-Subklasse definiert und ermöglichen die Verwendung der einfachen Angaben wie der Anzahl von Wahlberechtigten oder Wählenden, ohne die ursprünglichen Feldbezeichnungen wie `"A"` oder `"B"` verwenden zu müssen.
In der statischen Eigenschaft `properties` werden Felder weitergehend definiert, anhand von Klassen, die das Interface `FieldDescription` implementieren. In diesen Klassen wird nicht nur die zu verwendende Eigenschaftsbezeichnung `propName` angegeben, sondern auch in Oberflächen verwendbare Angaben wie den Namen, oder ob es sich um eine Summe oder eine Proportion handelt.
Die einfachste dieser `FieldDescription`-Klassen ist `ConstantFieldDescription`, die dazu verwendet wird, eine der bereits erwähnten „einfachen“ Angaben zu beschreiben. Erst dadurch, dass eine einfache Eigenschaft auch als Objekt dieser Klasse in `properties`, und nicht nur in `constantProperties`, definiert wird, wird es beispielsweise für Oberflächen als „verwendbar“ bereitgestellt.
`CollectedFieldDescription` wird verwendet, um die komplexen Felder für abgegebene Stimmen abzubilden. Es ist anzugeben, welche Ursprungsfelder die ungültigen (z. B. `"C"`) und die gültigen Stimmen je nach Stimmzetteleintrag (z. B. `"D1"` usw., angegeben wird nur die Basis `"D"`) enthalten. Zur Überprüfung wird die Bezeichnung der Eigenschaft angegeben, die die Gesamtanzahl abgegebener Stimmen enthält. Besonders sind bei `CollectedFieldDescription` die anzugebenden Objekte des Typs `DataTypeType`, über die Funktionen angegeben werden, die weitere Berechnungen durchführen, worauf erst bei Ergebnis-AnalysisCollection eingegangen wird.
`CalculatedFieldDescription` beschreibt Eigenschaften, die dynamisch anhand der Werte anderer einfacher Eigenschaften berechnet werden, beispielsweise die Wahlbeteiligung als Verhältnis von Wählenden zu Wahlberechtigten.

Nachdem im `Ergebnis`-Konstruktor anhand der zuvor beschriebenen Angaben die Eigenschaften (mit Gettern) am neuen Objekt der `Ergebnis`-Subklasse definiert wurden, wird die Methode `update` aufgerufen, die den intern verwendeten „privaten“ Eigenschaften Werte zuweist. Daraufhin können statisch definierte Überprüfungen stattfinden, beispielsweise ob Summen mit ihren zugrundeliegenden Werten übereinstimmen. Für als `CollectedFieldDescription` definierte Eigenschaften findet dies auch grundsätzlich statt. In Listing 6.2 wird beispielhaft die Definition einer `Ergebnis`-Subklasse dargestellt.

```
export class ErgebnisKommunalwahlNRW extends Ergebnis {
    static constantProperties: { [key: string]: string } = {
        wahlberechtigteOhneWahlschein: "A1",
        wahlberechtigteMitWahlschein: "A2",
        wahlberechtigteNichtImWaehlerverzeichnis: "A3",
        wahlberechtigteGesamt: "A",
        waehlendeGesamt: "B",
        waehlendeMitWahlschein: "B2",
    }
```

```
    static properties: Array<FieldDescription> = [
        new CollectedFieldDescription({
            name: "Stimmen",
            base: "D",
            invalid: "C",
            votesumProp: "waehlendeGesamt",
            propName: "stimmen",
            ergebnisType: this,
            displayInTooltip: true,
            defaultDataTypeAndArgsIfInitial: { type: collectedDataTypes[1], args: [1] },
            dataTypes: collectedDataTypes,
        }),
        new CalculatedFieldDescription({
            name: "Wahlbeteiligung",
            isSum: () => false,
            fn: (_1, _2) => _1 / _2,
            args: ["waehlendeGesamt", "wahlberechtigteGesamt"],
            propName: "wahlbeteiligung",
            ergebnisType: this,
            displayInTooltip: true,
        }),
        new ConstantFieldDescription({
            name: "Wahlberechtigte",
            propName: "wahlberechtigteGesamt",
            ergebnisType: this,
        }),
    ];

    static checks: { [key: string]: (o: ErgebnisKommunalwahlNRW) => boolean } = {
        "checkWahlberechtigte": (o) => (
            o.wahlberechtigteOhneWahlschein
            + o.wahlberechtigteMitWahlschein
            + o.wahlberechtigteNichtImWaehlerverzeichnis
        ) === o.wahlberechtigteGesamt,
    }
}
```

LISTING 6.2: Gekürzte Implementierung einer `Ergebnis`-Subklasse für die zu den Kommunalwahlen NRW 2020 dokumentierten Ergebnisdaten

6.3.3.2 ErgebnisAnalysis

Objekte der Klasse `ErgebnisAnalysis` dienen als Zusammenfassung mehrerer beliebiger `Ergebnis`-Objekte, um sie gleichberechtigt in ihrer Gesamtheit zu analysieren. So können beispielsweise alle vorliegenden Ergebnisse als Gesamtergebnis verwendet werden, oder mehrere Ergebnisse, die ein Gebiet höherer Ebene ausmachen, für ein aggregiertes Gebietsergebnis betrachtet werden.
Dem Konstruktor wird ein `Array` aus `Ergebnis`-Objekten übergeben. Das `Array` kann auch **`undefined`**-Werte enthalten, um auf noch fehlende Ergebnisse hinzuweisen. Im Konstruktor werden Getter für Eigenschaften definiert, anhand der ebenfalls übergebenen Angabe der `Ergebnis`-Subklasse, von der die Objekte im `Array` sind. Bei Zugriff auf die Eigenschaften

anhand ihres `propName` wird je nach Art der Eigenschaft (eine der `FieldDescription`-Klassen) eine Methode aufgerufen, die ein über alle `Ergebnis`-Objekte aggregiertes Resultat zurückgibt. Da nicht nur ein Zahlenwert (`value`), sondern auch Angaben wie die Anzahl zugrundeliegender `Ergebnis`-Objekte (`count`, `possibleCount`) zurückgegeben werden, ist die Rückgabe ein Objekt der Klasse `ResultDescription`.

Da bei `CollectedFieldDescription` nicht nur ein Wert relevant ist, wird in der von `ResultDescription` abgeleiteten Klasse `CollectedResultDescription` zusätzlich zur `value` (mit der Anzahl gültiger Stimmen) auch eine `Map` als Eigenschaft `results` bereitgestellt, mit der Stimmenzahl je Partei. Der Schlüssel in der Map ist jeweils der `anyName` der Partei: ein `String`, dem Kurznamen entsprechend, und wenn dieser nicht gegeben ist, dem Langnamen entsprechend.
`Map`-Einträge sind sortiert, dies kann zum Vorteil genutzt werden, um die Stimmenzahlen in der Reihenfolge auszugeben, wie sie auf dem Stimmzettel stehen. Da durchaus `Ergebnis`-Objekte unterschiedlicher `Stimmzettel` die Grundlage einer `ErgebnisAnalysis` sein können, muss hier vor den `Map`-Einfügungen eine besondere Gewichtung durchgeführt werden (vgl. Methode `ErgebnisAnalysis.collectedProp`). Die resultierende Sortierung stimmt anhand von Stichproben bei verschiedenen Bezirksvertretungswahlen mit den Darstellungen auf den votemanager-Seiten überein, obwohl der dort verwendete Algorithmus nicht bekannt/zugänglich ist.

In Listing 6.3 sind beispielhafte Resultate der Verwendung von `ErgebnisAnalysis` dargestellt.

```
// Methode ergebnisAnalysis erstellt eine ErgebnisAnalysis anhand der
// zugrundeliegenden Gebiete niedrigster Ebene
let eA = wahlController.activeWahl.ebenen.get(2).get("12").ergebnisAnalysis();

eA.wahlberechtigteGesamt
> ResultDescription {value: 6078, count: 7, possibleCount: 7, ...}

// enthält nicht nur einen Wert (gültige Stimmen),
// sondern auch eine Map mit Stimmen je Partei (Angabe anyName)
eA.stimmen
> CollectedResultDescription {value: 2560, results: Map(11), count: 7, possibleCount: 7, ...}

eA.wahlbeteiligung
> ResultDescription {value: 0.43024021059559064, count: 7, possibleCount: 7, ...}

eA.stimmen.results
> Map(11) {"SPD" => 777, "CDU" => 792, "GRÜNE" => 264, "HAGEN AKTIV" => 152, ...}

// Hilfsfunktion zur Ermittlung der erstplatzierten Partei(en)
eA.collectedPropPlace(eA.stimmen.fieldDesc, 1)
> {keys: ["CDU"], totalValid: 2560, votes: 792}
```

LISTING 6.3: Ermittlung einer `ErgebnisAnalysis` für einen Wahlbezirk bei der Ratswahl in Hagen, und gekürzte beispielhafte Verwendung der Eigenschaften in der Konsole.

6.3.3.3 ErgebnisAnalysisCollection

`ErgebnisAnalysisCollection` wird als Zusammenfassung mehrerer Gebiets-`ErgebnisAnalysis` verwendet, zugeordnet zu ihrem entsprechenden `Gebiet`/`WahlGebiet`-Objekt. Der Umfang dieser Sammlung kann beispielsweise eine vollständige `Ebene` mit allen ihren Gebieten sein (oder eine Untermenge der Gebiete, im Falle einer Filterung). Es werden Eigenschaften definiert, die es ermöglichen, eine Menge an Werten beispielsweise zur Einfärbung jedes einzelnen Gebiets auf einer Karte zu verwenden — insbesondere unter statistischer Berücksichtigung, da jeweils eine Liste aller Werte einer bestimmten Eigenschaft/Berechnung ausgewertet werden kann.
Die Klasse ist von `Map` abgeleitet, gespeichert werden die `ErgebnisAnalysis`-Objekte zugeordnet zu einem Schlüssel vom Typ `GebietInterface`. Übrigens muss bei `Gebiet`-Objekten (also auf höheren Ebenen) der Umfang der `ErgebnisAnalysis` nicht vollständig allen vorhandenen `WahlErgebnis`-Objekten dieses Gebiets entsprechen — aufgrund von Filterungen kann es sein, dass nicht alle Gebiete niedrigster Ebene berücksichtigt werden sollten[2]. Es wäre falsch, die Filterung in dieser Situation zu ignorieren und in der jeweiligen `ErgebnisAnalysis` alle Ergebnisse vom `Gebiet` zu berücksichtigen.

Bei Verwendung einer Eigenschaft wird die Methode `propFn` der `ErgebnisAnalysisCollection` verwendet, die ein `Object` zurückgibt, dessen Eigenschaftsnamen den Bezeichnungen von `DataTypeType`-Objekten der abgebildeten `FieldDescription` entsprechen. Dies ist nur bei `CollectedFieldDescription` relevant.
Bei *sämtlichen* Eigenschaften wird zunächst (mit dem Eigenschaftsnamen `undefined` im zurückzugebenden `Object`) die `calculate`-Methode der `ErgebnisAnalysisCollection` verwendet, um alle Hauptwerte (`value` aus `ResultDescription`) zu sammeln und in einem Objekt des Typs `CollectedResultType` zurückgeben. Das Wort „Collected“ bezieht sich dabei nicht auf die `CollectedFieldDescription`, sondern auf die Tatsache, dass Ergebnisse mehrerer gleichberechtigter Gebiete gesammelt werden. Es werden drei `Array`s mit übereinstimmender Sortierung zurückgegeben, jeweils mit den `GebietInterface`-Objekten, Datenwerten, und Datenklassen inklusive ihrer Farben (bei Verwendung der `calculate`-Methode nicht belegt). Die Verwendung von `Array`s ermöglicht die direkte Verwendung dieser Rückgaben in Bibliotheken, die eine solche Auflistung von Werten als Eingabe erwarten. Bei anderen Verwendungen muss zur Zuordnung eine Schleife verwendet werden, die je Durchlauf in den verschiedenen `Array`s den gleichen Indexwert verwendet.

Bei Eigenschaften mit definierten `dataTypes` (also wenn es sich um `CollectedFieldDescription` handelt) wird daraufhin etwas ähnliches gemacht, allerdings nicht mit der

[2] Beispiel: Bei einer Bezirksvertretungswahl wird eine `ErgebnisAnalysisCollection` für Ebene 2 erstellt, es ist aber der Filter nach einem bestimmten Stadtbezirk (`Stimmzettel`) aktiv. Eines der Ebene-2-Gebiete enthält nach der Filterung nicht etwa gar keine zu berücksichtigenden Gebiete niedrigster Ebene mehr, und auch nicht alle davon — sondern beispielsweise nur ein Gebiet niedrigster Ebene, welches als einziges nach der Filterung übrig bleibt.

`calculate`-Methode. Es sind in `ErgebnisAnalysisCollection` drei Methoden implementiert, die erweiterte Berechnungen durchführen, die nicht nur auf `ResultDescription.value` basieren, und auch parametrisiert sind. Der Rückgabetyp parametrisierter Ergebnisse entspricht nicht `CollectedResultType` mit den `Array`s, sondern `DataTypeFnResultType`. Objekte dieser Art enthalten ein `Array` erlaubter Argumente der ebenfalls mitgegebenen Funktion des Rückgabetyps `CollectedResultType`.
Implementiert sind folgende verwendbare parametrisierte Methoden:

calculateCollectedProportions
: Die zurückgegebene Funktion gibt in den `Array`s die berechneten **Anteile** einer bestimmten Partei (mitgegeben anhand ihres `anyName`) zurück. Die Datenklassen sind einheitlich.

calculateCollectedPlacements (vgl. Beispiel in Listing 6.4)
: Die zurückgegebene Funktion gibt in den `Array`s die berechneten **Anteile** der sich auf einem jeweiligen **Platz** (mitgegeben anhand einer Zahl) befindlichen Partei zurück. Die Datenklassen entsprechen dem `anyName` der jeweiligen Partei, für Gleichstände werden separate Datenklassen mit weißer Klassenfarbe definiert.

calculateCollectedDistances
: Die zurückgegebene Funktion gibt in den `Array`s den berechneten **Abstand** zwischen den **Anteilen** zweier Parteien (mitgegeben anhand ihres `anyName`) zurück. Anstelle einer zweiten Partei kann über den Wert `"1./2."` der Abstand zur jeweils Erstplatzierten bzw. zur Zweitplatzierten (bei eigener Erstplatzierung) berechnet werden. Die Datenklassen beschreiben, ob es sich bei dem Abstandswert um einen Vorsprung oder Rückstand (oder Gleichstand) handelt.

```
{
  classColors: Map(3) {"CDU" => "#010101", "SPD" => "#d60029", "BfHo" => "#9fe1d6"}
  classes: (26) ["CDU", "CDU", "SPD", "CDU", "CDU", "CDU", "SPD", "SPD", "SPD", "SPD", "CDU", ...]
  data: (26) [26.77, 24.74, 24.24, 27.46, 35.99, 34.58, 33.56, 25.93, 25.87, 25.65, 37.35, ...]
  gebiete: (26) [Map(5), Map(6), Map(5), ...] // Objekte einer GebietInterface implementierenden Klasse,
                                              // hier Objekte der von Map abgeleiteten Gebiet-Klasse
}
```

LISTING 6.4: Gekürzte Rückgabe (`Object` entspricht Typ `CollectedResultType`) eines Aufrufs der von `calculateCollectedPlacements` zurückgegebenen Funktion mit der Platzierung `1` als Argument.

Die weitere Verwendung dieser Rückgaben wird in Zusammenhang mit dem Interface und mit der Kartendarstellung beschrieben.

6.4 Interface und Darstellung

Kern der Inhalte dieses Abschnitts ist die bereits unter Anwendungsstruktur eingeführte Klasse `WahlController`. Sie definiert verschiedene Eigenschaften zur Zustandsverwaltung. Diese erfolgt also im `WahlController`-Objekt der Anwendung und nicht auf Ebene eines `Wahl`-Objekts oder über die Steuerungselemente.

wahlTerminConfig

Der aktuell ausgewählte Wahltermin aus der Konfiguration. Bei Änderung werden andere Eigenschaften zurückgesetzt und `Wahl`-Objekte für die im neu ausgewählten Wahltermine erstellt.

activeWahl

Die aktive `Wahl`, die die Grundlage für die auswählbaren und darstellbaren Ebenen und Eigenschaften bietet. Bei Änderung wird die asynchrone Methode `loadActiveWahl` aufgerufen, die einen Dialog dargestellt und das Laden der Daten der `Wahl` auslöst (Methode `Wahl.loadData`).

activeEbene, activeEAC, stimmzettelGebietFilter

Die aktive `Ebene` und dazugehörige `ErgebnisAnalysisCollection` (ggf. unter Berücksichtigung der Filterung nach Stimmzettelgebiet in Eigenschaft `stimmzettelGebietFilter`). Bei Änderung der `activeEbene` werden die Geodaten aus der Ebenenkonfiguration geladen. Bei der ersten Zuweisung der `activeEAC` wird die erste der in der `Ergebnis`-Subklasse definierten Eigenschaften (`FieldDescription`) automatisch als `activeProp` gesetzt.

activeProp, activeDataType

Aktuell zur Darstellung ausgewählte `FieldDescription`, bei einer `CollectedFieldDescription` wird in `activeDataType` ggf. zusätzlich der ausgewählte `DataType` mit den aktuell ausgewählten Parameterwerten gespeichert. Nach Änderung dieser Eigenschaften (vgl. 6.4.1.1) muss die Methode `WahlController.applyData` separat aufgerufen werden, um die Kartendarstellung zu aktualisieren (vgl. 6.4.2.3).

6.4.1 Steuerungselemente

Wie unter Anwendungsstruktur erwähnt, werden der Leaflet-Karte verschiedene Steuerungselemente hinzugefügt. Über diese erfolgt die Änderung der zuvor beschriebenen Zustandseigenschaften des `WahlController`-Objekts.

Zur Hinzufügung der Elemente in einer Ecke des Leaflet-Kartencontainers wird die Standardklasse `L.Control` mit der Methode `extend` erweitert[3], bei Hinzufügung wird ein beliebiges, in Methode `onAdd` zu ermittelndes HTML-Element verwendet [91].
Innerhalb dieser konkret für diese Anwendung (im Unterverzeichnis `interface`) definierten `L.Control`-Subklassen werden eigens mithilfe von LitElement definierte Custom Elements verwendet, die folgend einzeln beschrieben werden. In den meisten Fällen erfolgt dabei keine wesentliche Daten- oder Zustandsspeicherung in diesen Elementen, sie sind dann abhängig von den im `WahlController` verwalteten Eigenschaften. Da die Eigenschaften bei Änderung nicht unmittelbar an den Elementen geändert werden, erfolgen vom `WahlController` aus explizite Aktualisierungsveranlassungen.

[3] Leaflet arbeitet nicht mit dem Konzept moderner JavaScript-Klassen, deswegen ist für Subklassen diese Methode zu verwenden [90].

<**main-control**> (MainControlElement)

Stellt den Namen des aktuellen Wahltermins als Überschrift dar, und eine Tableiste (<**wl-tab-group**> aus Weightless) ermöglicht die Auswahl einer `Wahl` dieses Wahltermins. Ein Knopf ermöglicht den Wechsel des Wahltermins über einen Dialog, der die Methode `WahlController.wahlterminDialog` aufruft.

<**ebenen-control**> (EbenenControlElement)

Sobald eine `Wahl` ausgewählt ist schließt das Element sich visuell an die MainControl an, und setzt bei Auswahl einer Ebene der enthaltenen Tableiste die entsprechende `Ebene` als `activeEbene` des `WahlController`.

<**ergebnisse-control**> (ErgebnisseControlElement)

Sobald eine `Ebene` und die damit verbundene `activeEAC` des `WahlController` gesetzt ist, werden hier Auswahlen zur Ergebnisdarstellung bereitgestellt. Definiert die `Wahl` mehr als 1 `Stimmzettel`, wird es ermöglicht, über `WahlController.gebietFilterDialog` die Eigenschaft `stimmzettelGebietFilter` des `WahlController` zu ändern. Außerdem wird ein Button zur Gesamtergebnisdarstellung bereitgestellt, die Darstellung erfolgt über die Methode `WahlController.gesamtErgebnisDialog`. Je `FieldDescription` wird ein <**ergebnis-prop**> dargestellt (folgt in 6.4.1.1).

<**legend-control**> (LegendControlElement)

Diesem Element werden Daten über die aktuell vorliegende Klassifizierung (u. a. Wertebereiche, Farben) mitgegeben, auf diese wird in Zusammenhang mit der Kartendarstellung näher eingegangen. Eine Callback-Funktion kann gesetzt werden, wodurch Hover-Events eine filternde Auswirkung auf die Kartendarstellung haben.

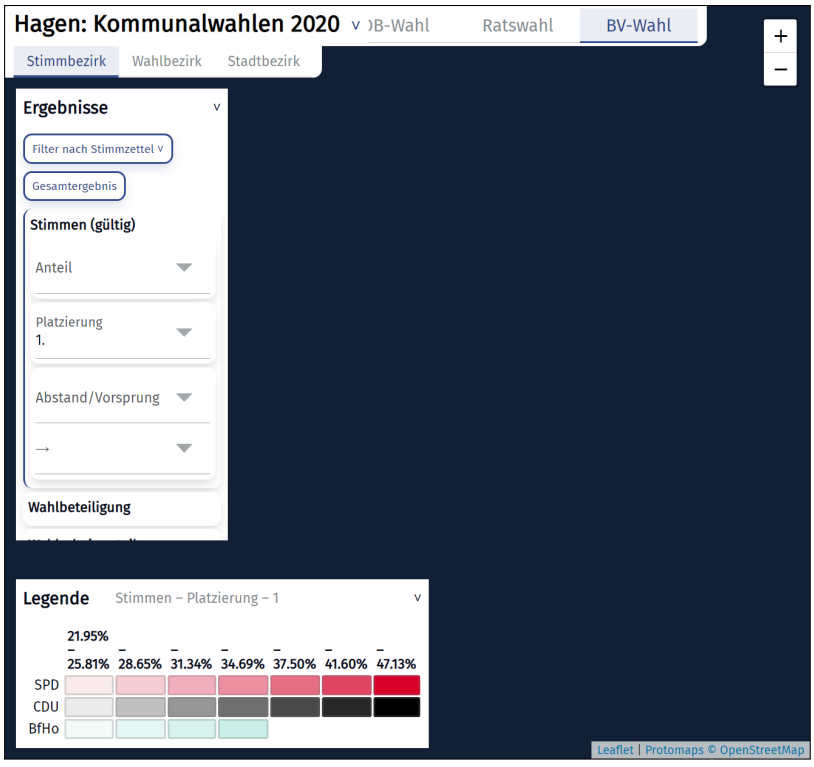

ABBILDUNG 6.2: Benutzungsoberfläche der Webanwendung. Dargestellt sind die Control-Elemente von oben nach unten in der zuvor textlich verwendeten Reihenfolge. Auf der rechten Seite befinden sich eingebaute Controls von Leaflet.

6.4.1.1 Auswahl von Eigenschaften

Elemente der Klasse `ErgebnisPropElement` (<**ergebnis-prop**>) werden im `ErgebnisseControlElement` verwendet, um die verfügbaren `FieldDescription` auswählbar zu machen. Die jeweilige `FieldDescription` eines einzelnen `ErgebnisPropElement` wird in der Eigenschaft `prop` angegeben.
Je Element wird die Bezeichnung (`FieldDescription.name`) dargestellt, bei Klick wird die Methode `setActive` des Elements aufgerufen, die für das Setzen der `activeProp`-Eigenschaft des `WahlController` zuständig ist.

Handelt es sich um eine `CollectedFieldDescription`, wird das Element deutlich umfangreicher. Je definiertem `DataTypeType` wird ein Auswahlelement dargestellt, in dem <**option**>-Elemente zur Auswahl enthalten sind. Das eigens definierte `CustomSelectElement` (<**wl-c-select**>) verwendet das <**wl-select**>-Element aus Weightless. Die eigene Klasse wird verwendet, damit das Auslösen von `'change'`-Events in allen benötigten Fällen veranlasst wird, insbesondere auch dann, wenn die zugrundeliegenden <**option**>-Elemente sich ändern und das bisher ausgewählte entfallen ist, wodurch der Wert zurückgesetzt wird und ein `'change'`-Event auch ausgelöst werden sollte. Für den erstellten Abstands-`DataTypeType` wurde ein separates Element erstellt, `AbstandSelectElement` (<**abstand-select**>), welches intern zwei <**wl-select**> verwendet, damit die Auswahl zweier Parameter für einen `DataTypeType` ermöglicht wird. Nach außen hin wird eine einheitliche `value`-Eigenschaft bereitgestellt, die die Werte in einem `Array` zurückgibt. Ist einer der Werte nicht definiert, werden beide Werte in dieser Rückgabe nicht belegt.
Bei Änderung einer der für `DataTypeType` verwendeten Elemente, also zum `'change'`-Event, wird die Methode `applyArgs` des `ErgebnisPropElement` aufgerufen. In dieser wird der Wert eines ggf. zuvor ausgewählten anderen Auswahlelements auswirkungsfrei zurückgesetzt, so dass jeweils nur eines der Auswahlelemente gleichzeitig Werte enthalten kann. Schließlich wird die Eigenschaft `activeDataType` des `WahlController` geändert, in der das jeweilige `DataTypeType`-Objekt und das `Array` aus Parameterwerten vorgehalten wird.

Nach Klick auf eine Überschrift oder Änderung eines Auswahlwertes wird als letztes `WahlController.applyData()` aufgerufen, wodurch anhand der zuvor gesetzten `active*`-Eigenschaften sämtliche Datenverarbeitung zur Visualisierung durchgeführt wird (vgl. 6.4.2.3).

6.4.2 Kartendarstellung

In diesem Abschnitt werden zunächst die angewendeten technischen Grundlagen der Kartendarstellung erläutert. Daraufhin werden die kartographischen Grundlagen beschrieben, die für die Datenverarbeitung und Klassifizierung von wesentlicher Bedeutung sind. Zuletzt werden die spezifisch für diese Anwendung implementierten Funktionalitäten der Interaktivität beschrieben.

6.4.2.1 Technische Grundlagen

Während mit Leaflet standardmäßig Hintergrundkartenkacheln über <**img**>-Elemente und Vektordaten über <**svg**>-Elemente eingebunden werden [64], verwendet Tangram, das insbesondere auf Vektordaten ausgelegt ist, WebGL zur Darstellung [92].
Über Tangram können Vektorkacheln zur Darstellung beispielsweise der Hintergrundkarte eingebunden werden. Ein wesentlicher Vorteil der Verwendung von Vektordaten gegenüber Rasterdaten ist die Möglichkeit der Verwendung von Objekteigenschaften und Interaktivität. Kartenstile können clientseitig konfigurierbar gemacht werden. Es wird ermöglicht, kartographische Methoden wie Generalisierung (Vereinfachung) oder Platzierung von Textelementen unter Berücksichtigung anderer Elemente dynamisch durchzuführen [93].

Die Einbindung von Daten im Tangram-Layer, zunächst für einen Hintergrundlayer, erfolgt über ein Konfigurationsobjekt. In der Eigenschaft `scene.import` kann ein `Array` an zu importierenden YAML-Dateien übergeben werden, in denen umfangreiche Definitionen von Datenquellen und Stilen erfolgen können [94]. Für die entwickelte Anwendung wird ein Stil (static/dark_simple.yml) benutzt, der auf einem Kartenstil von *Protomaps* basiert. Protomaps ist ein kostenloser Dienst, der unter anderem Vektorkacheln, basierend auf OpenStreetMap-Daten, bereitstellt [95]. Im `sources`-Objekt wurde die dazugehörige Datenquellendefinition beibehalten. Sie besteht unter anderem aus der Angabe des Typs (in diesem Fall *MVT*, für Vektorkacheln gemäß des Standards Mapbox Vector Tiles) und einer parametrisierten URL zur Protomaps-API.
Das `layers`-Objekt macht einen Großteil der Kartenstildefinition aus. Es können beliebige Layer unter Angabe der zuvor definierten Datenquelle und ggf. einer dortigen Layerbezeichnung definiert werden. Innerhalb dieser Layer-Konfigurationen werden Angaben zur Filterung und zur Darstellung der geographischen Objekte und ggf. dazugehörigen Labels gemacht [96].
Der für diese Anwendung verwendete Kartenstil ist im Gegensatz zur Grundlage mit dunklen Farben definiert und um diverse Layer gekürzt. Dies ist so, weil der Fokus bei der Anwendung auf den darzustellenden wahlbezogenen Flächendaten und ihren Einfärbungen liegen sollte.

Für die thematischen Daten, also die wahlbezogenen Flächen, ist es grundsätzlich möglich, die GeoJSON-Daten direkt in Leaflet zu verwenden [97]. Mit Tangram ist dies auch möglich, was mit verschiedenen Vorteilen verbunden ist. Es kommt nicht zu sichtbaren Artefakten bei der Bewegung der Karte, die sonst erfahrungsgemäß auftreten, vermutlich aufgrund der verschiedenen und voneinander unabhängigen Rendering-Verfahren. Darüber hinaus können alle Rendering-Funktionalitäten von Tangram auch für die thematischen Daten eingesetzt werden, darunter beispielsweise verschiedene Modi der Farbmischung, der Einsatz von Shadern, oder die zukünftig denkbare Nutzung von 3D-Funktionalitäten [96]. Durch die einheitliche Nutzung von Tangram für alle Karteninhalte wird es auch vereinfacht, verschiedene Daten auf verschiedenen Ebenen übereinander darzustellen. Bei der Verwendung eines Rasterkachelhintergrunds in Leaflet müssten Schriftzüge beispielsweise weggelassen werden, und separat über den thematischen Daten angezeigt werden. In Tangram kann die

Definition hingegen so erfolgen, dass bestimmte Renderreihenfolgen und weitere Regelungen angegeben werden, um eine erwünschte zusammenhängende Kartendarstellung zu erhalten [96].

Die Definition des thematischen Kartenlayers (genannt `"districts"`) erfolgt direkt bei der Erstellung des Tangram-Layers in `app.js`, zusätzlich zum importierten Grundkartenstil. Dabei wird die Quelle zunächst bewusst nicht definiert, da diese erst im `WahlController` nach dem Setzen einer `activeEbene` und dem Laden der jeweiligen GeoJSON-Daten erfolgt (über die Methode `setDataSource` des `scene`-Objekts des Tangram-Layers). Eine wesentliche Eigenschaft bei den darstellungsbezogenen Angaben ist der `Boolean`-Wert `interactive`, wodurch Tangram die Objekte der jeweiligen Art bei Zeigerklick/-bewegung berücksichtigen kann (vgl. 6.4.2.4) [98]. Die Farbe der darzustellenden Polygone wird anhand ihrer Eigenschaft `color` ermittelt. Wie die dortigen Werte ermittelt werden, wird unter Datenverarbeitung, Klassifizierung beschrieben. Bezeichnungen (beispielsweise Gebietsnamen) dieser Daten werden nicht in der Karte dargestellt, dies erfolgt aber in Zusammenhang mit der Interaktivität. Im Hintergrundkartenstil ist hingegen definiert, dass Stadt- und Ortsteilnamen über den Gebieten dargestellt werden. Diese auf den Protomaps/OpenStreetMap-Daten basierenden Angaben dienen, zusätzlich zu den ebenfalls dargestellten Gemeindegrenzen, als „Ankerpunkte“ bei der Kartenansicht. Außerdem werden bei einem hohen Zoomlevel Straßennamen angezeigt, die auf einer hyperlokalen Ebene als Referenz dienen. Auch diese Labels wurden so definiert, dass sie *über* den Gebietsflächen dargestellt werden (vgl. spätere Abbildung 6.5).

Zunächst ist das Konzept von `transform` und `extra_data` aus Tangram zu erläutern, wovon bei der noch zu beschreibenden Einfärbung und Interaktivität oft Gebrauch gemacht wird. Beides sind Angaben, die bei der Definition einer `source` gemacht werden können, um vor jedem Rendering bereits geladene Daten zu manipulieren [65].
Bei `extra_data` handelt es sich um ein beliebiges `Object`. Relevant wird es dadurch, dass es in der `transform`-Funktion verfügbar ist. Der erste Parameter der Funktion (`data`) wird für die grundlegenden Daten verwendet, der zweite Parameter der Funktion (`extra_data`) entspricht dem `extra_data`-Objekt der Quelle, wenn es definiert ist. In dieser Funktion kann dann beispielsweise eine Schleife über alle Objekte ausgeführt werden, um ihre Eigenschaften, ggf. unter Berücksichtigung von Daten aus `extra_data`, zu ändern oder zu erweitern. Wichtig ist, dass diese `transform`-Funktion das `data`-Objekt schließlich zurückgeben sollte.
In der Anwendung wird von `transform` und `extra_data` Gebrauch gemacht, um Informationen unter anderem zur Einfärbung, Klassifizierung, und Tooltip-Inhalten aus dem `Wahl Controller` heraus in den Eigenschaften der Bezirksdaten vor der Darstellung berücksichtigen zu lassen.

6.4.2.2 Kartographische Grundlagen

Bei der Darstellung von Werten in Flächen anhand von Einfärbungen handelt es sich um eine Choroplethenkarte. Aus ihr wird die räumliche Verteilung von Statistiken ersichtlich. Daten werden in bestimmten definierten Flächen wie beispielsweise Ortsgrenzen, und sie

können aggregiert werden, um auf höheren Ebenen dargestellt zu werden [7]. In Zusammenhang mit Wahlen erfolgt die Darstellung meistens insbesondere anhand der wahlbezogenen Bezirke. Choroplethenkarten sollten nicht eingesetzt werden, um absolute Werte darzustellen, da Datenwerte oft von der Bevölkerungszahl (oder z. B. der Zahl an Wahlberechtigten) abhängig sind und bei *proportionalen* Werten eine Vergleichbarkeit eher gegeben ist.
Die Darstellung erfolgt durch unterschiedliche Einfärbung von Gebieten, meistens von einer hellen zu einer dunkleren oder einer weniger zu einer eher gesättigten Farbe. Bei bimodalen (divergierenden) Datenreihen werden an den Enden der Farbskala unterschiedliche Farbtöne verwendet, mit einem neutralen Farbton in der Mitte [7].
Bei einer wie beschrieben eingefärbten Darstellung anhand von Werten handelt es sich um eine *quantitative* Darstellung. Von einer *qualitativen* Karte wird gesprochen, wenn unterschiedliche Einfärbungen gleichmäßig nebeneinander stehen und unterschiedliche *Arten* von Daten bezeichnen sollen [7]. Zur Darstellung der Platzierung von Wahlvorschlägen in der Web-Anwendung kann gesagt werden, dass sie einen qualitativen Aspekt hat, da einerseits wie sonst auch der Stimmanteil über die Einfärbung ersichtlich wird, andererseits aber auch der Farbton, abhängig von dem Wahlvorschlag, dessen Wert dargestellt wird, sich unterscheidet (vgl. spätere Abbildung 6.5). Im weiteren Verlauf dieses Kapitels wird dabei von unterschiedlichen *Klassen* gesprochen.

Für die Darstellung von Werten in einer Choropleth-Karte erfolgt oft eine Klassifizierung. Dabei werden Daten so gruppiert, dass die Lesbarkeit erhöht wird und geographische Muster erkennbarer werden [7]. Um nicht mit den zuvor beschriebenen Klassen von Daten verwechselt zu werden, wird im weiteren Verlauf dieses Kapitels für die Gruppen das englische Wort *Bin(s)* verwendet.
Es gibt verschiedene Methoden der Klassifizierung, die zu unterschiedlichen Resultaten führen können. Zu den mathematischen Methoden zählen die Verwendung regelmäßiger Intervalle (Equal Intervals), die Einteilung in eine bestimmte Anzahl gleich großer Bins (z. B. Quartiles für vier), oder die Verwendung der Standardabweichung bei einer Normalverteilung [7].
Die statistische Methode, die für die Anwendung verwendet wird, wird auch als „Natural Breaks“ bezeichnet, unter der Verwendung des Jenks-Algorithmus. Dabei kommt es nicht etwa zu einer Klassifizierung anhand regelmäßiger Abstände oder Anzahlen, sondern es wird eine Einteilung erstellt, die sich aus „natürlichen“ Grenzen in den Daten statistisch signifikant ergibt ([99] mit Bezug auf [100]).
Es ist notwendig, die Anzahl zu erstellender Bins vor Ausführung des Algorithmus anzugeben. Dies kann Auswirkungen auf das Resultat haben. Es gibt theoretische Vorschläge für Voranalysen, die eine strukturierte Lösung dieses Problems ermöglichen, ohne dass die Anzahl explizit mit Kenntnis über die Daten angegeben werden muss (vgl. [99]). Es handelt sich nicht um einen Ersatz des Jenks-Algorithmus, da es nicht darum geht, *wo* die Grenzen bei der Klassifizierung liegen, sondern *wie viele* es geben sollte [99].
Grundsätzlich ist zur Anzahl an Bins bei Choroplethenkarten zu sagen, dass eine Anzahl von 4 bis 9 [7, 99] für eine optimierte Unterscheidbarkeit und Lesbarkeit empfohlen wird.

Choroplethenkarten sind nicht immer eine optimale Methode, statistische Daten zu Gebieten auf einer Karte darzustellen. Eine Alternative für die Darstellung absoluter Werte sind

Punktkarten. Die Werte in Gebieten werden dabei nicht durch die Helligkeit oder den Farbton der Einfärbung, sondern durch die Punktdichte ersichtlich [7].
Die Idee klassischer Choroplethenkarten kann erweitert werden um die Darstellung der zugrundeliegenden Faktoren wie der Anzahl von Einwohnenden oder Wahlberechtigten durch Verzerrung der Flächen [7]. Der Bezug zu Hintergrundkarten geht bei solchen Kartogrammen schnell verloren, es entsteht aber eine akkurate Abbildung der Bedeutung einzelner Gebiete, so dass beispielsweise Gebiete mit einer geringen Bevölkerungsdichte visuell nicht überrepräsentiert werden [7]. Als bekanntes Beispiel dafür können Karten zu US-Wahlen genannt werden, bei diesen kommt es in klassischen Darstellungen zu einer starken visuellen Überrepräsentation einer Partei, die weniger Stimmen als eine andere hat.
Eine andere Möglichkeit wäre die ausschließliche Verwendung von Mittelpunkten von Gebieten, auf denen Kreise dargestellt werden, die abhängig von der Bevölkerungszahl oder dem Stimmenvorsprung in ihrem Durchmesser variieren [101].

6.4.2.3 Datenverarbeitung, Klassifizierung

Wie zuvor mehrfach angedeutet, wird die Methode `applyData` von `WahlController` verwendet, um die Daten für die Kartendarstellung zu ermitteln. Die zuvor beschriebenen Zustandseigenschaften werden dabei verwendet, um die notwendigen Daten zu erhalten. Die Methode hat keine Parameter und kann so jederzeit aufgerufen werden, ohne dass beispielsweise die Angabe der darzustellenden Daten bei dem Methodenaufruf zusätzlich erfolgen muss. Es wird also vorausgesetzt, dass die Eigenschaften `activeEAC` und `activeProp` gesetzt sind. Andernfalls wird die Methode `resetData` aufgerufen, die jegliche in `applyData` gesetzte Eigenschaften sowie die Kartendarstellung zurücksetzt.
Zunächst wird das Aggregationsergebnis der aktuellen `ErgebnisAnalysisCollection` für die aktuelle Eigenschaft geholt, und darin das Ergebnis anhand des aktuellen `DataType`␣`Type` (bzw. **`undefined`**). Hat das erhaltene `Object` die Eigenschaft `fn`, wenn es sich also um ein `Object` des Typs `DataTypeFnResultType` handelt, wird diese Funktion mit den in `activeDataType` vorgehaltenen Argumenten ausgeführt. Schließlich liegt in jedem Fall ein `Object` des Typs `CollectedResultType` vor. Ist an dieser Stelle bereits klar, dass keine Daten vorhanden sind, wird `resetData` aufgerufen und die Ausführung von `applyData` beendet. Andernfalls beginnt dann die weitere Datenverarbeitung und Klassifizierung.

Zu Beginn der Klassifizierung werden zunächst verschiedene (theoretisch konfigurierbare) Parameter definiert. Die Anzahl an Bins wird zurzeit grundlegend auf 7 gesetzt, nur im Falle eines geringen Datenumfangs wird diese reduziert, weil es sonst erfahrungsgemäß dazu kommt, dass Bins mit **`NaN`**-Grenzen erstellt werden, um auf die angegebene Anzahl zu kommen. Eine dynamische, datenabhängige Bestimmung der *optimalen* Anzahl an Bins erfolgt zurzeit nicht.
Es wird eine „Standardfarbe“ definiert, die verwendet wird, wenn die Daten keine Klassen haben. Darüber hinaus wird ein Basis-Farbwert, Weiß, angegeben, der bei den resultierenden Farbbereichen als niedrigster Wert verwendet werden soll, ggf. zu einem bestimmten Anteil (15 %) gemischt mit der Zielfarbe des Farbbereichs, damit die niedrigsten Bins bei Vorhandensein mehrerer Klassen nicht alle im gleichen Weiß dargestellt werden.

Es wird ein geostats-Objekt erstellt, welches dann in der Eigenschaft serie ein Array mit allen einzelnen Datenwerten, also einem Wert je Gebiet, gesetzt bekommt. Mit der Methode getClassJenks von geostats wird veranlasst, dass anhand der gesetzten Datenserie eine Klassifizierung mit der Jenks-Optimierung durchgeführt wird [88].
Bevor anhand der nun erstellten Klassifizierung die weitere Datenverarbeitung erfolgt, werden abschließend alle tatsächlichen Farbenbereiche mit einzelnen Farbwerten ermittelt. Je Klasse (bzw. ohne Klassen nur für die Standardfarbe) wird ein Array mit so vielen Farbwert-Einträgen wie es Bins gibt ermittelt. Dies erfolgt unter Einsatz der Bibliothek chroma.js [87]. Mit der scale-Funktion wird ein Farbbereich zwischen der (wie zuvor beschrieben ggf. leicht gemischten) Basisfarbe und der Zielfarbe der Klasse bzw. der Standardfarbe definiert. Als Wertebereich wird der Wertebereich der Daten definiert, wobei davon im weiteren Verlauf kein Gebrauch gemacht wird, da die Klassifizierung über geostats verwaltet wird. Mit einem Aufruf der Methode correctLightness wird sichergestellt, dass die Farbhelligkeit im Farbbereich gleichmäßig verteilt ist, was für die Interpretierbarkeit der in der Karte dargestellten Farben von Bedeutung ist. Mit der Methode colors wird schließlich eine gewünschte Anzahl an Farbwerten aus dem Farbbereich gezogen, so dass das Farbwert-Array für die Bins verwendbar vorliegt.
Die Speicherung dieser Farbwerte erfolgt zugeordnet zur jeweiligen Klasse in einer Map. Die Einfügung in die Map erfolgt in einer der Häufigkeit an Daten der jeweiligen Klasse entsprechenden Reihenfolge, so dass die Klassen beispielsweise in einer Legende auch so nach Häufigkeit sortiert dargestellt werden können. Zusätzlich wird gezählt, wie viele Datenpunkte *je Bin* jeder Klasse existieren. So kann bei einer Legende dafür gesorgt werden, dass Bins, die in der Kartendarstellung gar nicht verwendet werden, nicht dargestellt werden, wodurch auch der Wertebereich einzelner Klassen erkennbar wird.
Eine beispielhafte Darstellung der Klassifizierung und der Farbwerte kann der Legende in der zuvor eingefügten Abbildung 6.2 oder einigen noch folgenden Abbildungen entnommen werden.

Zu diesem Zeitpunkt in der Datenverarbeitung innerhalb von applyData liegen also sämtliche Informationen zu Farben und Klassifizierung vor. Die erstellten Map-Objekte werden in den Eigenschaften des geostats ergänzt, welches wiederum als Eigenschaft _active_Geostats des WahlController festgehalten wird. So kann auf diese Informationen jederzeit auch nach deren Erstellung zurückgegriffen werden, wie beispielsweise in der Methode WahlController.updateLegend. Diese wird auch von applyData aus aufgerufen, sie sorgt dafür, dass die Eigenschaften des LegendControlElement aktualisiert werden (wodurch auch ein neues Rendering des Elements veranlasst wird). Die Funktionalitäten der Legende werden in Zusammenhang mit Interaktivität weitergehend erläutert.

Nach der Klassifizierung muss schließlich die gebietsbezogene Verarbeitung folgen, damit jedes Gebiet unter anderem einen Farbwert zugewiesen bekommt. Es erfolgt eine Schleife über die Map.entries der aktiven ErgebnisAnalysisCollection, es ist also in jedem Schleifendurchlauf das GebietInterface implementierende Objekt und die dazugehörige ErgebnisAnalysis verfügbar. Diese wird aber zunächst nicht für die Einfärbung verwendet, da dies anhand des ursprünglichen CollectedResultType-Object und des geostats-Objekts erfolgt. Dessen Methode getClass wird verwendet, um für einen Datenwert die

Bin, in dem dieser liegt, zu erhalten [88]. Anhand dessen wird im jeweiligen Farbrange der Klasse dieses Datenpunkts der korrekte Farbwert für das vorliegende Gebiet anhand der Nummer der Bin ermittelt. Liegen zu einem Gebiet keine Daten vor, wird ein dafür vorgesehener Farbwert verwendet. Ist dieser Wert `undefined`, wird das Gebiet nicht sichtbar und auch nicht auswählbar sein. Die Angabe eines Wertes, beispielsweise einer durchscheinenden Farbe, sorgt für den Erhalt des Gebietes bei dem Rendering. Die Tatsache, dass zum Gebiet keine Daten vorliegen, wird in einem weiteren Attribut festgehalten und bei der Darstellung durch eine Schraffierung verdeutlicht (vgl. Abbildung 6.4).

Die für das Gebiet im Schleifendurchlauf ermittelten Angaben müssen, damit die Darstellung anhand dieser erfolgen kann, zugreifbar gespeichert werden. An dieser Stelle kommt das `extra_data`-Objekt ins Spiel, in diesem werden mehrere `Object`s angelegt, die einem Gebiet anhand der Nummer je einen Wert zuordnen. Dies geschieht also mit den zuvor beschriebenen Angaben wie dem Farbwert, dem Klassennamen, und der Nummer der Bin aus der Klassifizierung. Grundsätzlich sind letztere Angaben nicht notwendig, sie werden aber in Zusammenhang mit der Legende und Interaktivität zum Einsatz kommen.
Weiterhin im Schleifendurchlauf wird anhand der `ErgebnisAnalysis` des aktuellen Gebiets ein HTML-Inhalt für ein Tooltip definiert und schließlich auch in `extra_data` zur Speicherung als Objekteigenschaft für die spätere interaktive Verwendung festgehalten.

Damit die zuvor ermittelten Daten für ein frisches Kartenrendering berücksichtigt werden, erfolgt abschließend ein Aufruf von `scene.rebuild`, woraufhin alle in `extra_data` vorhandenen Angaben im `transform`-Aufruf den geographischen Objekten in ihren Eigenschaften ergänzt oder geändert werden.

In den folgenden vier Abbildungen werden Resultate der Klassifizierung anhand unterschiedlicher Beispiele dargestellt.

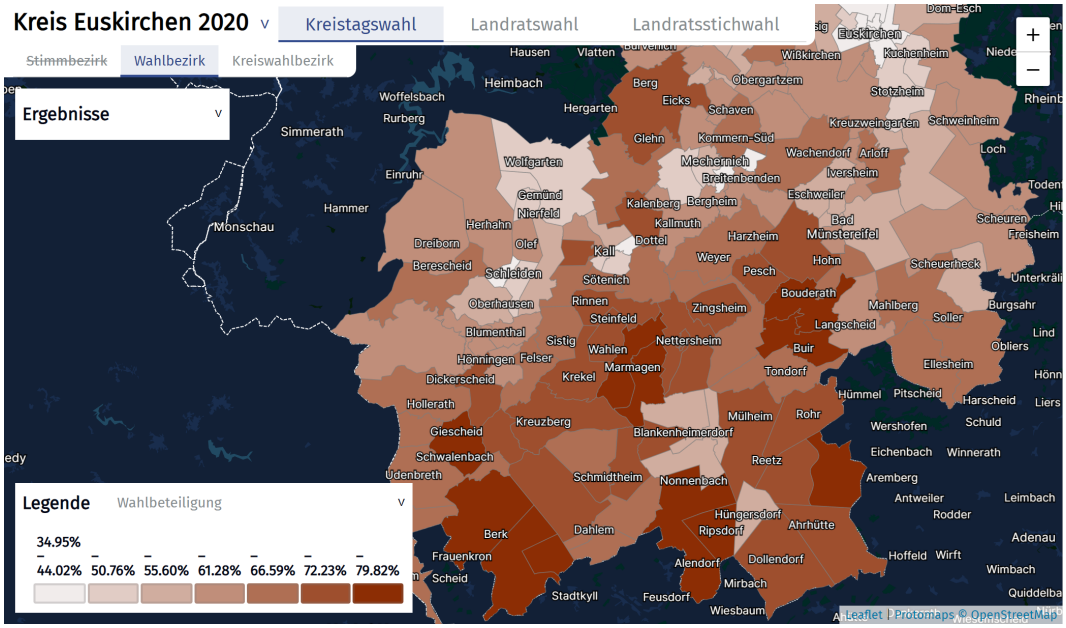

ABBILDUNG 6.3: Darstellung der Wahlbeteiligung. Da keine unterschiedlichen Datenklassen bestehen, wird die definierte Standardfarbe verwendet.

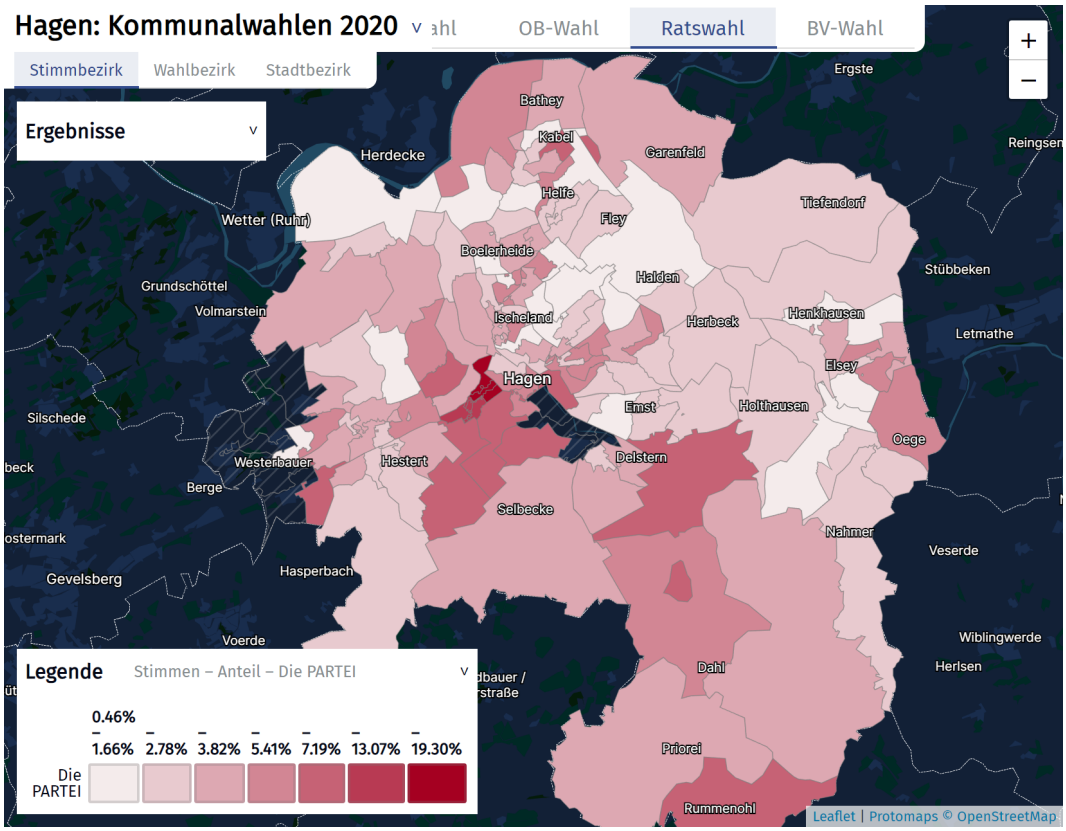

ABBILDUNG 6.4: Darstellung des Anteils einer bestimmten Partei unter Berücksichtigung der jeweiligen Klassenfarbe. Da die Partei nicht in allen Bezirken gewählt werden konnte, werden diese Bereiche ohne Daten schraffiert dargestellt.

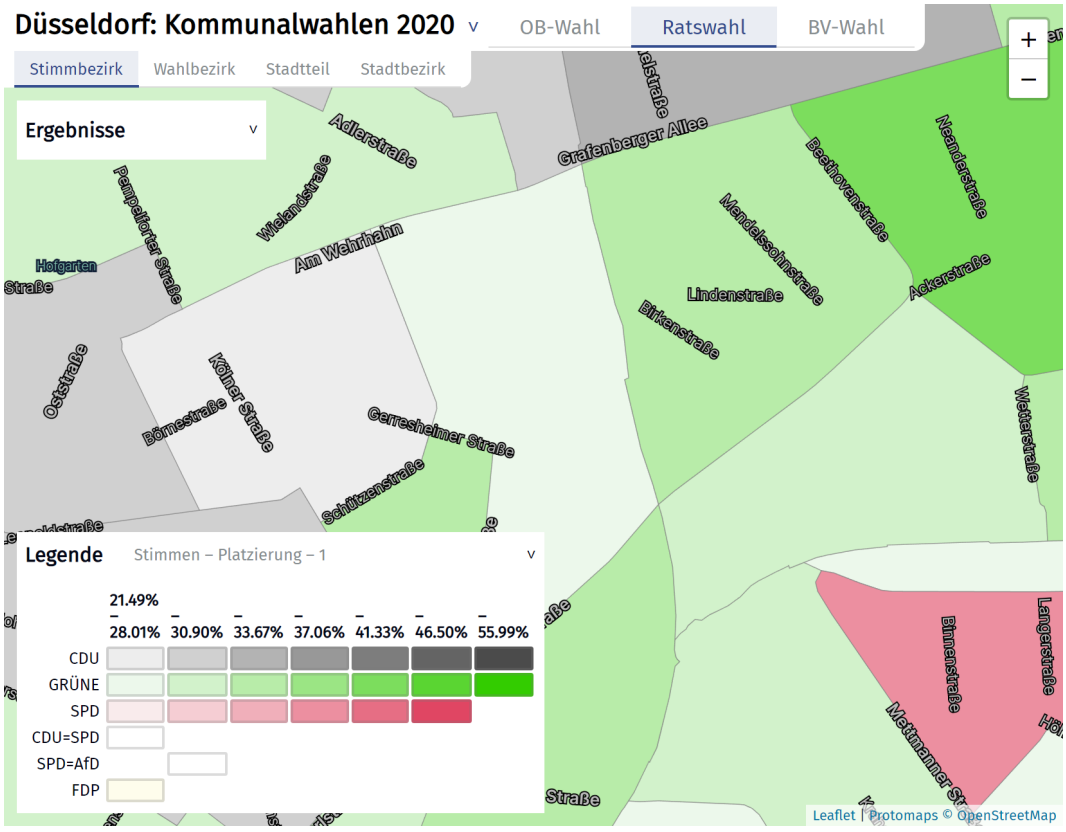

ABBILDUNG 6.5: Darstellung der erstplatzierten Partei je Bezirk mit der entsprechenden Klassenfarbe und wertabhängigen Farbstärke. Auf dem in diesem Beispiel verwendeten Zoomlevel werden für den lokalen Kontext Straßennamen dargestellt.

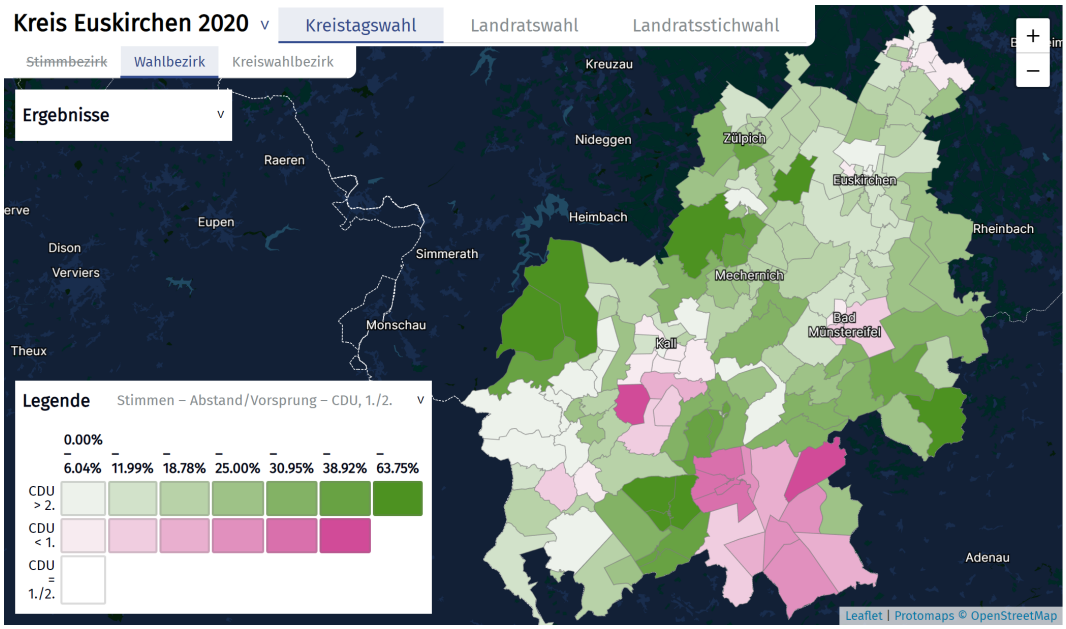

ABBILDUNG 6.6: Darstellung des Abstands der Anteile zwischen einer ausgewählten Partei und der erstplatzierten, bzw. im Falle der eigenen Erstplatzierung mit dem Vorsprung zur zweitplatzierten Partei. Da letzteres häufiger eingetreten ist, wird diese Klasse in der ersten Inhaltszeile der Legende dargestellt.

6.4.2.4 Interaktivität

In diesem Abschnitt geht es um die Elemente der Interaktivität, die über die sich ergebende Interaktivität aufgrund der Nutzung von Leaflet (verschiebbare Kartenansicht usw.) oder der bereits beschriebenen erstellten Darstellungswechselmöglichkeiten hinaus geht.

Tangram ermöglicht die Angabe von Handler-Funktionen für Zeigerbewegungen (*hover*) und Klicks (*click*). Die Funktionen können bei der Definition des `Tangram.leafletLayer` im Konfigurationsobjekt `events` oder über die Methode `setSelectionEvents` übergeben werden. Die angegebenen Funktionen bekommen als Argument ein Objekt, welches Angaben zum Event und zu „ausgewählten“ Features enthält. Damit Features eines Tangram-Layers berücksichtigt werden können, ist die Angabe der Eigenschaft `interactive` bei der Layer/Draw-Konfiguration notwendig [102, 103].

Die Funktion `clickGebiet` wird zum Handling der Klicks verwendet und greift auf das `WahlController`-Objekt der Anwendung zu, welches auch als globale Variable zugänglich ist. Für das `GebietInterface` implementierende Objekt, dessen geographische Abbildung angeklickt wurde, wird eine `ErgebnisAnalysis` ermittelt. Es wird ein neuer Dialog mit Weightless erstellt, diesem wird als Inhalt ein <**`ergebnis-element`**> (`ErgebnisElement`) unter Angabe der aktiven `Wahl` und der `ErgebnisAnalysis` als Eigenschaften übergeben. Die Klasse `ErgebnisElement` wird auch bei Klick auf den „Gesamtergebnis“-Button in dem `ErgebnisseControlElement` verwendet, sie wird aber erst folgend beschrieben. Ein `ErgebnisElement` enthält eine Grafik, die üblichen nicht-geographischen Wahlergebnisdarstellungen wie in votemanager ähnelt. Außerdem werden die Ergebnisse tabellarisch dargestellt. Die Datengrundlage ist eine beliebige `ErgebnisAnalysis`.

Für die Darstellung der Grafik (vgl. Abbildung 6.7) wird ein eigenes Element <**`ergebnis-chart`**> (`ErgebnisChartElement`) verwendet, welches anhand einer `Map` mit `Partei`-Objekten und einer `ErgebnisAnalysis` für eine bestimmte `CollectedFieldDescription` eine Grafik unter Einsatz der Bibliothek Chart.js [104] darstellt. Es werden dafür `Array`s mit den Bezeichnungen, Anteilwerten, und Farben jeder Partei verwendet. Tooltip- und Achsenbezeichnungen werden über Callback-Funktionen angepasst.

Für die tabellarische Darstellung (vgl. Abbildung 6.8) wird ein weiteres eigenes Element, <**`ergebnis-table`**> (`ErgebnisTableElement`) verwendet. In den Spalten werden Partei- und Kandidaturangaben sowie die zugehörigen proportionalen und absoluten Stimmenzahlen dargestellt. Ein Großteil der Datenverarbeitung bezieht sich auf die angemessene Darstellung der Kandidaturdaten, da es wie unter 2.1, 4.2 und 6.3.2.3 beschrieben verschiedene Rahmenbedingungen hinsichtlich der Arten von Kandidaturen geben kann.

Die grafische und tabellarische Darstellung erfolgt jeweils für jede Stimmen-Eigenschaft, also jede definierte `CollectedFieldDescription`. Bei den bisher verwendeten Wahlarten ist dies also ein Mal für die abgegebenen Stimmen der Fall. Bei (z. B.) Bundestagswahlen gäbe es für Erst-/Zweitstimmen jeweils separate Darstellungen. Am Ende des `ErgebnisElement` erfolgt für alle weiteren `FieldDescription`-Arten eine textliche Darstellung.

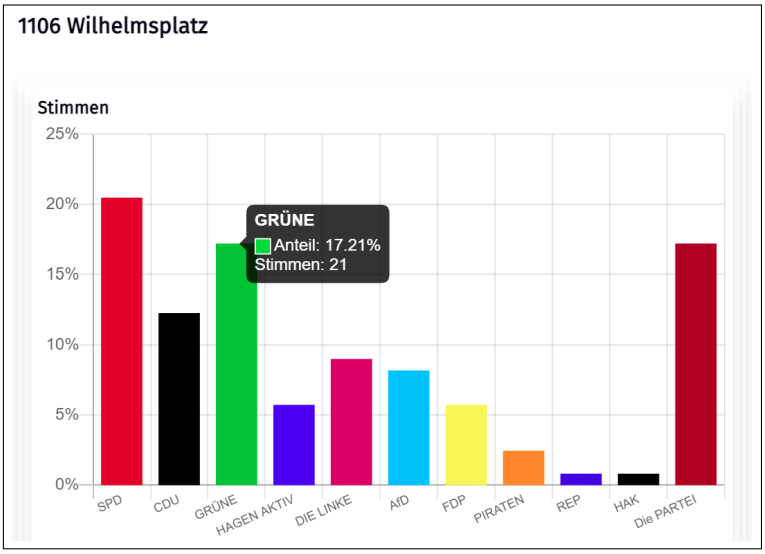

ABBILDUNG 6.7: Ergebnisgrafik in einem Stimmbezirk der Ratswahl. Bei Hover über die Balken werden Tooltips mit Details dargestellt.

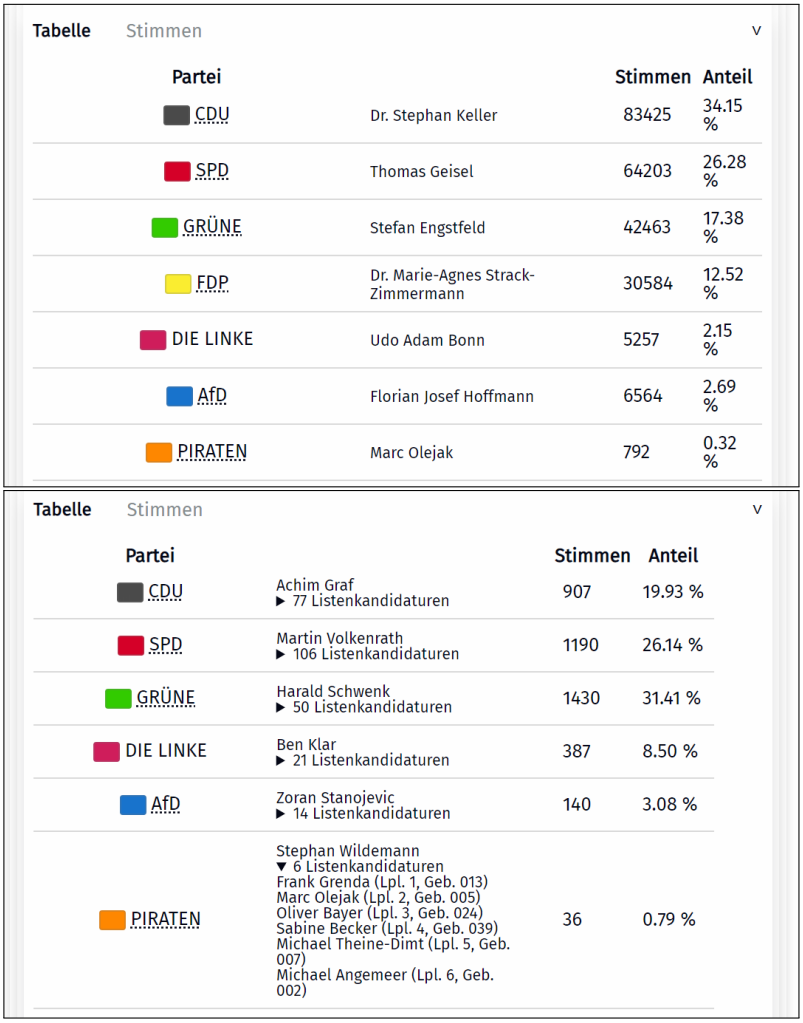

Tabelle Stimmen

Partei		Stimmen	Anteil
CDU	Dr. Stephan Keller	83425	34.15 %
SPD	Thomas Geisel	64203	26.28 %
GRÜNE	Stefan Engstfeld	42463	17.38 %
FDP	Dr. Marie-Agnes Strack-Zimmermann	30584	12.52 %
DIE LINKE	Udo Adam Bonn	5257	2.15 %
AfD	Florian Josef Hoffmann	6564	2.69 %
PIRATEN	Marc Olejak	792	0.32 %

Tabelle Stimmen

Partei		Stimmen	Anteil
CDU	Achim Graf ▶ 77 Listenkandidaturen	907	19.93 %
SPD	Martin Volkenrath ▶ 106 Listenkandidaturen	1190	26.14 %
GRÜNE	Harald Schwenk ▶ 50 Listenkandidaturen	1430	31.41 %
DIE LINKE	Ben Klar ▶ 21 Listenkandidaturen	387	8.50 %
AfD	Zoran Stanojevic ▶ 14 Listenkandidaturen	140	3.08 %
PIRATEN	Stephan Wildemann ▼ 6 Listenkandidaturen Frank Grenda (Lpl. 1, Geb. 013) Marc Olejak (Lpl. 2, Geb. 005) Oliver Bayer (Lpl. 3, Geb. 024) Sabine Becker (Lpl. 4, Geb. 039) Michael Theine-Dimt (Lpl. 5, Geb. 007) Michael Angemeer (Lpl. 6, Geb. 002)	36	0.79 %

ABBILDUNG 6.8: Tabellendarstellung für das Gesamtergebnis einer Oberbürgermeister∗innenwahl (oben) und ein Wahlbezirksergebnis einer Ratswahl, mit Wahlbezirks- und Listenkandidierenden (unten).

Hover-Events, also Zeigerbewegungen in der Kartenansicht, werden in der Funktion `hover_Gebiet` behandelt. Die wesentliche Funktionalität darin ist die Darstellung eines Tooltips (vgl. Abb. 6.9). Die HTML-Inhalte der Tooltips werden nicht bei der Verarbeitung des Events erstellt, sondern wurden bereits in `applyData` zur Berücksichtigung in den Objekteigenschaften über `extra_data` definiert. Dies wird verwendet, um mit der Standardfunktionalität von Leaflet ein `L.Tooltip` zu erstellen oder die Position zu aktualisieren [64].

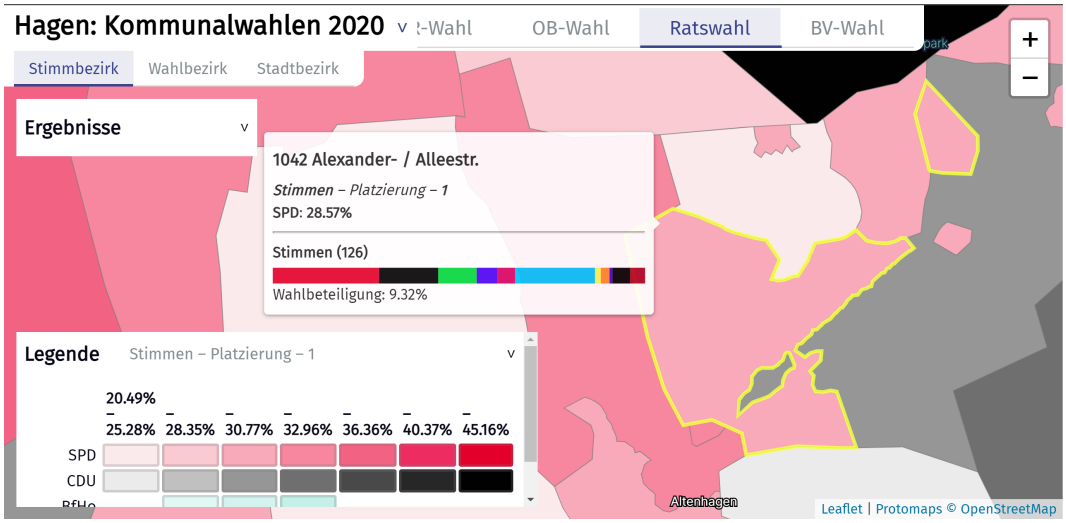

ABBILDUNG 6.9: Darstellung eines Tooltips und Hervorhebung des betroffenen Gebietes. Dargestellt wird, für die zuvor ausgewählte Eigenschaft (Platzierung), die jeweilige Datenklasse und der Wert.
Für die Stimmeneigenschaften wird *grundsätzlich* eine proportionale Balkengrafik dargestellt. Auch die Wahlbeteiligung ist sichtbar, aufgrund der Angabe von `Field_Description.displayInTooltip` bei der Eigenschaftsdefinition.

Außerdem wird die jeweilige Fläche durch eine Umrandung besonders hervorgehoben. Dies erfolgt über die Änderung einer Eigenschaft in `extra_data` und die Anforderung eines neuen Renderings, so dass die Hervorhebung wie im Layerstil definiert angewendet werden kann. Es hat sich hinsichtlich der Nutzungserfahrung als problematisch herausgestellt, wie diese Hervorhebung umgesetzt wurde. Es ist eine Art „Stolpern“ erkennbar, weil bei einer Zeigerbewegung auf ein anderes Gebiet für einen Augenblick in manchen Fällen wieder eine unzutreffende Hervorhebung und Tooltipdarstellung erfolgt. Dass dieser Anwendungsfall nicht optimal vorgesehen ist, ist auch den Tangram-Entwickelnden bekannt, es wurde vor mehreren Jahren geschrieben, dass dies zukünftig zu verbessern sei [105, 106].

Zusätzlich zur Verarbeitung der Interaktion mit den Karteninhalten gibt es die Möglichkeit, die Karteninhalte über die Legende (zuvor beschriebenes `LegendControlElement`) dynamisch zu filtern. Die Bins der Klassifizierung werden spaltenweise aufgelistet, mit einer Zeile je Klasse. Die Spaltenüberschriften stellen die Wertebereiche der Bins dar. Dargestellt werden nur Bins, die laut der mitgegebenen Anzahl von Daten überhaupt bei den dargestellten Flächen verwendet werden (vgl. Abbildung 6.10).

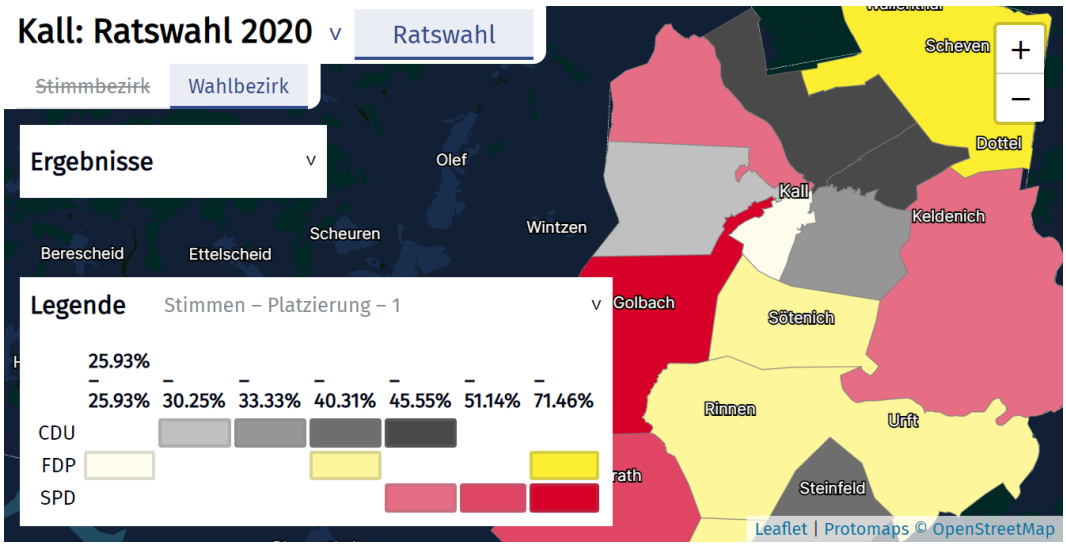

ABBILDUNG 6.10: Dadurch, dass nur die verwendeten Bins dargestellt werden, ist erkennbar, welche Wertebereiche je Datenklasse überhaupt zutreffend sind.

Über alle Tabellenzellen inklusive der Spalten/Zeilenbezeichnungen kann der Zeiger bewegt werden, um einen Aufruf der mitgegebenen Funktion `hoverCallback` mit Klassenangabe und/oder Bin-Angabe zu veranlassen. Bei der von `WahlController` aus mitgegebenen Callback-Funktion handelt es sich um dessen Methode `markClass`, die eine Anpassung von `extra_data` durchführt und ein Neurendering veranlasst, wobei aufgrund geänderter Eigenschaften nur die aktuell durch das Zellenhovering zutreffenden Gebiete dargestellt werden (vgl. Abbildung 6.11). Es fällt dabei auf, dass es keine „Stolperer“ wie bei der Gebietsumrandung gibt. Das kann darauf hindeuten, dass die Problematik bei der Hervorhebung auf die Verarbeitung der Zeigerevents zurückzuführen ist, und nicht etwa auf die Dichte der aufeinanderfolgenden Rendering-Veranlassungen, die bei der Legende kein Problem sind.

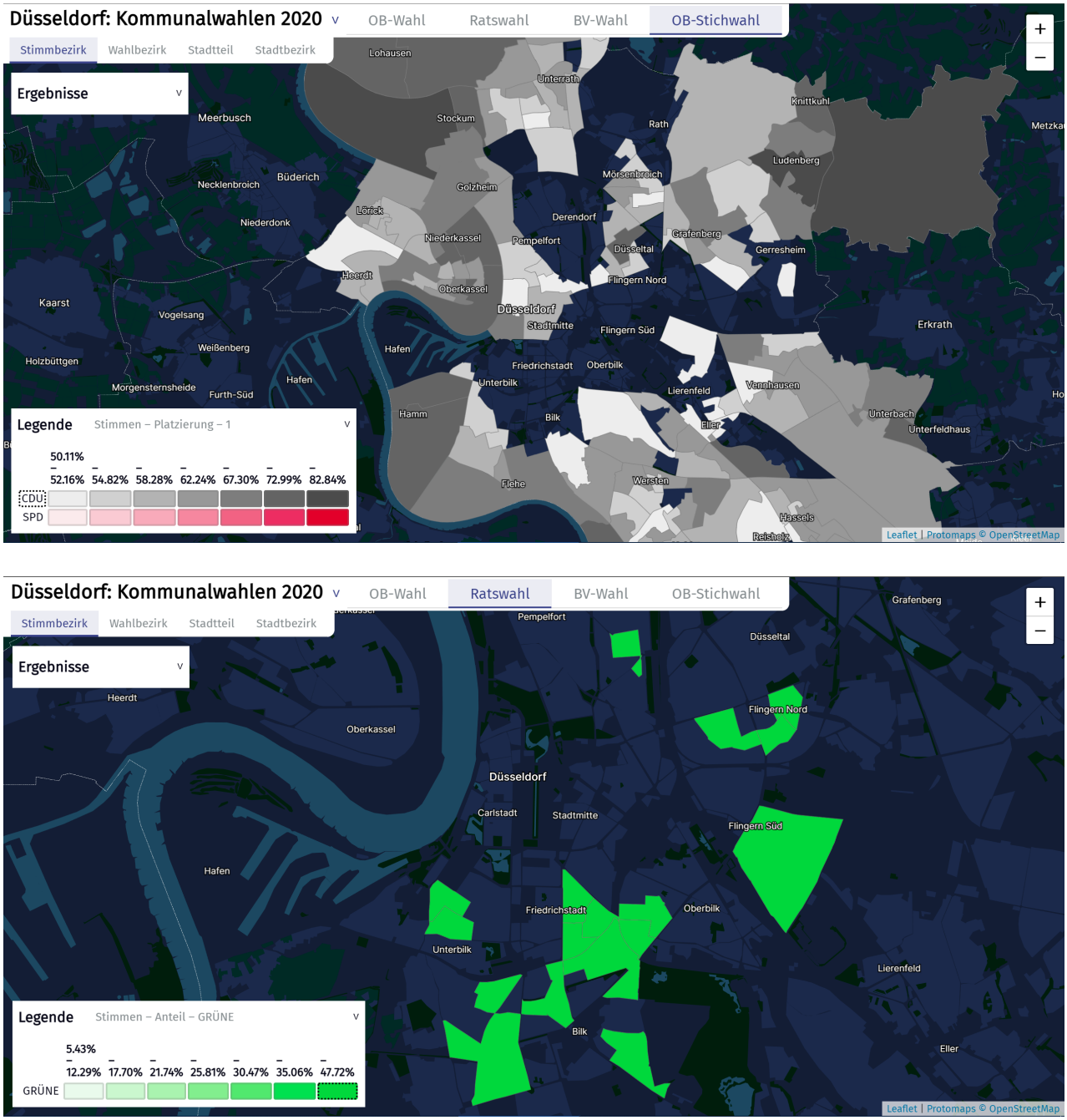

ABBILDUNG 6.11: Dynamische Filterung der Daten durch Hovering über Bezeichnungs- oder Inhaltszellen.
Im ersten Bild werden die Gebiete dargestellt, bei der die Datenklasse der ausgewählten Partei entspricht. Da die Platzierung-Analyse ausgewählt wurde, bedeutet dies, dass nur die Gebiete dargestellt werden, in denen diese Partei die erstplatzierte war.
Im zweiten Bild wird die Darstellung des Anteils einer einzelnen Partei verwendet. Alle Gebiete der ausgewählten (höchsten) Bin werden dargestellt. Daraus werden die Hochburgen der Partei ersichtlich.

7 Zusammenfassung

Verfügbarkeit von Daten

Der Standard für Offene Wahldaten ermöglicht erstmals eine umfassende maschinelle Verarbeitung von Wahldaten. Dadurch, dass auch Datensätze zu Gebietsdefinitionen, Stimmzetteleinträgen oder Kandidaturen enthalten sind, reduziert sich der benötigte Umfang an manueller Datenverknüpfung verglichen mit der sonst typischen ausschließlichen Bereitstellung von Ergebnisdaten.
Zur zukünftigen Verfügbarkeit dieser Wahldaten ist zum jetzigen Zeitpunkt nicht viel bekannt. Es ist zu erwarten, dass die Datensätze dieser Art auch in votemanager-Portalen bereitgestellt werden, dies würde für eine ebenfalls hohe Verbreitung in der Zukunft sorgen. Bei einer Weiterentwicklung des Datenstandards ist in Aussicht gestellt, dass auch eine dynamische Programmierschnittstelle bereitgestellt werden könnte. Die hohe Verfügbarkeit der bisherigen offenen Wahldaten in sämtlichen votemanager-Instanzen ist vermutlich darauf zurückzuführen, dass dies herstellerseitig als Standardfunktionalität implementiert wurde, nicht etwa darauf, dass dieser Zustand bewusst seitens jeder Kommune herbeigeführt wurde.
Bei wahlbezogenen Geodaten ist die bisherige Nichtverfügbarkeit von Open Data besonders auffällig. Selbst wenn eine Kommune nachweislich maschinenlesbare Geodaten vorliegend hat, sie diese also nicht erst erstellen müsste, bedeutet es nicht, dass man einfach an diese Daten kommt. In solchen Situationen müssen die Daten selber hergeleitet werden, was mit einem hohen Aufwand und unschönen Resultaten verbunden ist. Dadurch entstehen möglicherweise genau die Fehldarstellungen, die Kommunalverwaltungen eigentlich als Gegenargument der Veröffentlichung offener Daten anführen.
Über Open Data wird insgesamt viel geredet, doch praktisch gibt es trotz einzelner herausragender Beispiele kaum die Ausbreitung in der Fläche, die dazu nötig wäre, eine Anwendung wie die im Rahmen dieser Bachelorarbeit entwickelte ohne eigene Datenherleitung überall verwenden zu können.

Verarbeitung von Wahldaten

Die für die Anwendung implementierte Wahldatenverarbeitung erfolgte unter den Rahmenbedingungen des aktuellen Zustands des Standards für Offene Wahldaten. Sie kann in weiten Teilen auch für andere Zwecke der Wahldatenverarbeitung verwendet werden und ist abspaltbar von der restlichen Funktionalität der erstellten Web-Anwendung.

Bei der Implementierung musste mit Annahmen und Problemumgehungen gearbeitet werden, weil die Dokumentation Fragen offen lässt und die Maschinenverarbeitbarkeit des Datenmodells optimiert werden kann. Grundlegende Aussagen zu Relationen zwischen Datensätzen sind unvollständig. An manchen Stellen musste so vorgegangen werden, wie es sich anhand der Datensätze als korrekt erweist (quasi „reverse engineered“), da die Dokumentation unzureichend war.

Aufgrund der Beschaffenheit des Datenmodells in manchen Aspekten mussten bei der Datenverarbeitung zusätzliche Schritte erfolgen, um den Umgang mit Daten einfacher zu machen oder überhaupt zu ermöglichen. Gleichzeitig ist davon auszugehen, dass auch bei einfacheren Datenverwendungen, beispielsweise in Tabellenkalkulationssoftware, das Datenmodell nicht ohne weiteres einfach zu verwenden ist, beispielsweise wegen der bedeutungsvollen Relationen verschiedener Datensätze, ohne die eine Datennutzung schwer vorstellbar ist. Der Standard sei aber mit dem Ziel erstellt worden, sowohl Bedürfnissen der maschinellen Verarbeitung als auch der einfacheren Datenverwendung gerecht zu werden.
Als Fazit aus den Erfahrungen mit dem Umgang mit diesen Daten ist eher festzuhalten, dass es sinnvoller erscheint, das Format hinsichtlich der maschinellen Verwendbarkeit weiterzuentwickeln und zu dokumentieren. Gleichzeitig sollten, für die einfachen Verwendungen ohne erforderlichen Einsatz von Programmcode, separate Dateien abgeleitet werden können. Diese wären dann besonders darauf zugeschnitten, und sollten nicht als *primäre* Form der Wahldatenbereitstellung dienen.

Zusammenfassend ist festzuhalten, dass zunächst mindestens die Dokumentation des Datenstandards verbessert werden muss. Auch ohne große Veränderungen des Datenmodells wäre durch klare und verbindliche Informationen die Verarbeitbarkeit erhöhbar.
Gerade bei der Implementierung eigener Datenverarbeitungen und Anwendungen wäre es zusätzlich besonders hilfreich, wenn die Software votemanager quelloffen verfügbar wäre. Datennutzende hätten dadurch einen Einblick in die dortige Datenverarbeitung als „Referenzimplementierung“. Auch aus anderen gesellschaftlichen und sicherheitsbezogenen Gründen ist es wichtig, die Implementierung der im Rahmen von Wahlen verwendeten Software nicht geheim zu halten (vgl. [107]). Angesichts des mittlerweile im Jahr 2020 bestehenden umfangreichen Marktanteils sämtlicher Software, die unter Kontrolle der vote iT GmbH (mit ausschließlich kommunal strukturierten Gesellschaftern) steht, erscheint die Realisierung dieser Forderung, mit möglicherweise weniger Gegenargumenten, angebracht.

Geographische Wahlergebnisdarstellung

Zur interaktiven geographischen Darstellung von Wahlergebnissen wurde eine Web-Anwendung erstellt. Diese stellt auf Basis von Leaflet und vektorbasiertem Rendering mit Tangram unterschiedliche Einfärbungen von Gebieten in Form einer Choroplethenkarte dar. Es kann zwischen verschiedenen Darstellungen gewechselt werden, außerdem werden Tooltips und Popups bei Interaktion mit Kartenobjekten dargestellt. Die Anwendung kann im jetzigen Zustand als eigene Website verwendet werden, auch die Verwendung in Form eines Embeds ist denkbar. Je nach örtlichen Bedürfnissen ist es möglich, dass für zusätzlichen Kontext beliebige weitere Kartenlayer und Funktionen in der Leaflet-Karte ergänzt werden.

Die Inbetriebnahme der Anwendung ist besonders aufwandsarm. Die Datenverarbeitung erfolgt clientseitig, es ist kein zusätzliches Backend erforderlich. Liegen die Daten im benötigten Umfang vor, ist das Aufsetzen einer neuen Instanz oder das Hinzufügen eines weiteren Wahltermins mit wenig Aufwand verbunden, da größtenteils nur auf die Datenquellen verwiesen werden muss und geringfügige Angaben zur Darstellung von Bezeichnungen gemacht werden müssen.

Vor einem produktiven Betrieb sind gegebenenfalls kurzfristig weitere Anpassungen durchzuführen. Dazu kann die Erweiterung um Hilfe- und Einstellungsmenüs (um beispielsweise die Hervorhebung oder Kartenlabels auszuschalten) sowie die Berücksichtigung von Datensatz-Lizenzangaben gehören.
Darüber hinaus kommen verschiedene Ergänzungen in Frage, beispielsweise die Möglichkeit der Erstellung von Permalinks zu bestimmten Analysen oder Flächen. Inhaltlich kann der nicht-geographische Teil der Anwendung erweitert werden, um sich den Funktionalitäten klassischer Wahlportale anzunähern.

Außerdem besteht Potenzial hinsichtlich der Weiterentwicklung der Anwendung zu einer eigenen zusätzlichen Datenquelle. Neben dem Export der Visualisierungsergebnisse wäre es denkbar, weiterverarbeitete Wahldaten in einfach verwendbaren csv-Dateien oder Geodatenformaten bereitzustellen. So könnte die Anwendung als Anlaufstelle für Daten zur Verwendung in Tabellenkalkulationssoftware und Geoinformationssystemen o. ä. dienen, so dass der bisherige Anspruch, auch diesen Nutzungsarten gerecht zu werden, umgesetzt werden kann. Währenddessen würden die Daten gemäß des Standards für Offene Wahldaten auf die Maschinenverarbeitbarkeit unter Einsatz von Programmcode oder Datenbanken hin optimiert sein können.

Generell wird es in Zukunft notwendig sein, die Anwendung und die zugrundeliegende Datenverarbeitung den aktualisierten Gegebenheiten anzupassen, insbesondere zur Berücksichtigung potenzieller Weiterentwicklungen des Standards für Offene Wahldaten.

Literaturverzeichnis

[1] J. W. Falter und H. Schoen, Eds., *Handbuch Wahlforschung*. Wiesbaden: Springer Fachmedien Wiesbaden, 2014. https://doi.org/10.1007/978-3-658-05164-8

[2] J. W. Falter und J. R. Winkler, "Wahlgeographie und Politische Ökologie", in *Handbuch Wahlforschung*, J. W. Falter und H. Schoen, Eds. Wiesbaden: Springer Fachmedien Wiesbaden, 2014, S. 135–167. https://doi.org/10.1007/978-3-658-05164-8_5

[3] L. Matzat, "Datenjournalismus", Okt. 2011. https://www.bpb.de/gesellschaft/digitales/opendata/64069/datenjournalismus *[Archivlink]*

[4] A. B. Howard, *The Art and Science of Data-Driven Journalism*. Tow Center for Digital Journalism, 2014. https://doi.org/10.7916/D8Q531V1

[5] D. Bochsler, "Wie deckt die Statistik Wahlfälschung auf?", Dez. 2020. https://www.republik.ch/2020/12/14/wie-deckt-die-statistik-wahlfaelschung-auf *[Archivlink]*

[6] M.-L. Timcke, A. Pätzold, D. Wendler, und M. Klack, "Europawahl 2019 in Berlin", 2019. https://interaktiv.morgenpost.de/europawahl-berlin/ *[Archivlink]*

[7] G. Darkes und M. Spence, *Cartography: An Introduction*, 2. ed. British Cartographic Society, 2017, 978-0-904482-25-6.

[8] "Why Datawrapper?". https://www.datawrapper.de/why-datawrapper/ *[Archivlink]*

[9] A. Flohe, "Wahldatenstandard", Jul. 2019. https://oknrw.de/wahldatenstandard/ *[Archivlink]*

[10] Bundesministerium des Innern, für Bau und Heimat, "Bundestagswahlrecht". https://www.bmi.bund.de/DE/themen/verfassung/wahlrecht/bundestagswahlrecht/bundestagswahlrecht-node.html *[Archivlink]*

[11] "Bekanntmachung der Neufassung des Kommunalwahlgesetzes (Kommunalwahlgesetz) Nordrhein-Westfalen". https://recht.nrw.de/lmi/owa/br_text_anzeigen?v_id=4520040121111440485 *[Archivlink]*

[12] W. Zicht, "Übersicht über die Wahlsysteme bei Kommunalwahlen". https://www.wahlrecht.de/kommunal/index.htm *[Archivlink]*

[13] A. Vetter, "Kumulieren, Panaschieren und die Beteiligung der Bürger an kommunalen Wahlen", in *Zivile Bürgergesellschaft und Demokratie: Aktuelle Ergebnisse der empirischen Politikforschung*, S. I. Keil und S. I. Thaidigsmann, Eds. Wiesbaden: Springer Fachmedien Wiesbaden, 2013, S. 237–256. https://doi.org/10.1007/978-3-658-00875-8_13

[14] vote iT, "Votemanager". https://vote-it.de/?page_id=146 *[Archivlink]*

[15] Anstalt für Kommunale Datenverarbeitung in Bayern (AKDB), "OK.VOTE: neues Angebot für Wahlauswertungen", Feb. 2018. https://www.akdb.de/loesungen/okbuergerservice/aktuelles-aus-dem-bereich-okbuergerservice/news/detail/okvote-neues-angebot-fuer-wahlauswertungen/ *[Archivlink]*

[16] "Kauf von IVU.elect", Mai 2020. https://www.kommune21.de/meldung_34012_Kauf+von+IVU.elect.html *[Archivlink]*

[17] "vote iT GmbH baut Kooperation aus", 2020. https://vote-it.de/?p=863 *[Archivlink]*

[18] "Von PC-Wahl zum votemanager", Jan. 2020. https://www.kommune21.de/meldung_33153_Von+PC-Wahl+zum+votemanager.html *[Archivlink]*

[19] "Ausbau der Kooperation von ekom21 und vote iT", 2020. https://vote-it.de/?p=854 *[Archivlink]*

[20] Stadt Hagen, Der Oberbürgermeister, "RVR-Wahl / Kommunalwahlen / Integrationsratswahl 2020 in der Stadt Hagen - OpenData CSV-Dateien". http://wahlergebnisse.stadt-hagen.de/prod/KW2020/05914000/html5/OpenDataInfo.html *[Archivlink]*

[21] "Kommunalwahlordnung (KWahlO) Nordrhein-Westfalen". https://recht.nrw.de/lmi/owa/br_bes_text?anw_nr=2&gld_nr=1&ugl_nr=1112&bes_id=3356&aufgehoben=N&menu=1&sg=0 *[Archivlink]*

[22] B. Krabina, "Ein Leitfaden für offene Daten", 2020. https://www.bertelsmann-stiftung.de/en/publications/publication/did/ein-leitfaden-fuer-offene-daten *[Archivlink]*

[23] B. Krabina und B. Lutz, "Open-Government-Vorgehensmodell", 2016. https://www.kdz.eu/de/open-government-vorgehensmodell *[Archivlink]*

[24] B. Krabina, "Open Government und Open Data als Modernisierungskonzepte: Chancen und Herausforderungen offener Verwaltungen", in *Handbuch E-Government*, J. Stember, W. Eixelsberger, A. Neuroni, A. Spichiger, F. Habbel, und M. Wundara, Eds. Wiesbaden: Springer Gabler, 2019. https://doi.org/10.1007/978-3-658-21596-5_41-1

[25] A. Stern, “Open Data Policy and Freedom of Information Law”, Okt. 2018. https://sunlightfoundation.com/wp-content/uploads/2018/10/alena-white-paper-PDF.pdf *[Archivlink]*

[26] Kompetenzzentrum Open Data, “Open Data Handbuch”. https://www.bva.bund.de/SharedDocs/Downloads/DE/Behoerden/Beratung/Methoden/open_data_handbuch.pdf?__blob=publicationFile&v=8 *[Archivlink]*

[27] Kompetenzzentrum Open Data (Bundesverwaltungsamt), “Anforderung an die Daten – Leitfaden –”, Aug. 2020. https://www.bva.bund.de/SharedDocs/Downloads/DE/Behoerden/Beratung/Methoden/open_data_anforderungen_daten.pdf?__blob=publicationFile&v=2 *[Archivlink]*

[28] S. Kaufmann, “Das pwc-Gutachten fuer NRW zur Datenlizenz Deutschland”, Nov. 2019. https://stefan.bloggt.es/2019/11/das-pwc-gutachten-fuer-nrw-zur-datenlizenz-deutschland/ *[Archivlink]*

[29] OpenAIRE, “Creative Commons 4.0: it’s here!”. https://www.openaire.eu/launch-of-creative-commons-40-license *[Archivlink]*

[30] S. Kaufmann, “Die Datenlizenz Deutschland gehoert auf den Muell. Jetzt.”, Nov. 2019. https://stefan.bloggt.es/2019/11/die-datenlizenz-deutschland-gehoert-auf-den-muell-jetzt/ *[Archivlink]*

[31] A. Wiebe, “Open Data in Deutschland und Europa – Vorschlag zur Weiterentwicklung des rechtlichen Rahmens einer Informationsordnung (Open Data – Public-Service-Information Richtlinie)”, 2020. https://www.kas.de/en/single-title/-/content/open-data-in-deutschland-und-europa-1 *[Archivlink]*

[32] Bundeskanzleramt, “Zweiter Nationaler Aktionsplan 2019–2021”, 2019. https://www.open-government-deutschland.de/opengov-de/aktionsplaene-und-berichte/zweiter-nationaler-aktionsplan-1591034 *[Archivlink]*

[33] Bitkom e.V., “Landkarte Open Data Akteure in Deutschland”, 2020. https://www.bitkom.org/sites/default/files/2020-06/200602_grafik_open-data-akteure_titel.jpg *[Archivlink]*

[34] B. Fleischer und Y. Rother, “Germany: The path to open data leadership”, in *Digital Government: Leveraging Innovation to Improve Public Sector Performance and Outcomes for Citizens*, S. Falk, A. Römmele, und M. Silverman, Eds. Cham: Springer International Publishing, 2017, S. 169–189. https://doi.org/10.1007/978-3-319-38795-6_9

[35] Geschäftsstelle Open.NRW, “Open Government Pakt”. https://open.nrw/open-government/open-government-pakt *[Archivlink]*

[36] ——, “Pilotprojekt Kommunales Open Government”. https://open.nrw/Open-Government-in-NRW/Pilotprojekt-Kommunales-Open-Government *[Archivlink]*

[37] “Gesetz zur Förderung der elektronischen Verwaltung in Nordrhein-Westfalen (E-Government-Gesetz Nordrhein-Westfalen - EGovG NRW)”. https://recht.nrw.de/lmi/owa/br_bes_text?sg=0&menu=1&bes_id=34925&aufgehoben=N&anw_nr=2 *[Archivlink]*

[38] Geschäftsstelle Open.NRW, “Unabhängig kommunale Verwaltungsdaten veröffentlichen”, Jun. 2020. https://open.nrw/unabhaengig-kommunale-verwaltungsdaten-veroeffentlichen *[Archivlink]*

[39] A. Caffier, C. Elsner, C. Rath, F. Robens, J. Seidel, und K. Will, “Offene Geobasisdaten für NRW”, *zfv – Zeitschrift für Geodäsie, Geoinformation und Landmanagement*, 2017. https://doi.org/10.12902/zfv-0166-2017

[40] “Zweite Verordnung zur Änderung der Verordnung zur Durchführung des Gesetzes über die Landesvermessung und das Liegenschaftskataster”, Dez. 2019. https://recht.nrw.de/lmi/owa/br_vbl_detail_text?anw_nr=6&vd_id=18165&ver=8&val=18165&sg=0&menu=1&vd_back=N *[Archivlink]*

[41] “Geonetzwerk Metropole Ruhr”. https://www.geonetzwerk.ruhr/ *[Archivlink]*

[42] Regionalverband Ruhr, “Digitale Metropole Ruhr – Digitale Entwicklung für die Region aktiv gestalten”. https://www.rvr.ruhr/daten-digitales/digitale-metropole-ruhr/ *[Archivlink]*

[43] T. Bürger, A. Hoch, und Bertelsmann Stiftung, “Open Data in Kommunen”, 2020. https://doi.org/10.11586/2020068

[44] U. Thalheim und B. Hekele, “Warum wir über Wahldaten reden müssen”, Jan. 2018. https://codefor.de/blog/warum-wir-ueber-wahldaten-reden-muessen/ *[Archivlink]*

[45] E. Ruge, “Datenverarbeitung ohne Schnittstelle – wie geht das?”, Jan. 2015. https://openruhr.de/2015/01/04/datenverarbeitung-ohne-schnittstelle-wie-geht-das/ *[Archivlink]*

[46] Stadt Hagen, Der Oberbürgermeister, “Beschlussvorlage 0960/2018: Verzicht auf die Anbindung des Ratsinformationssystems ALLRIS an das Portal „Politik bei uns“”, Okt. 2018. https://www.hagen.de/buergerinfo/vo020.asp?VOLFDNR=18343&no_mobile=1 *[Archivlink]*

[47] Stadt Hagen, Der Oberbürgermeister (Einreichende: F. Schmidt, T. Kiszkenow), "Anfrage 0740/2015: Antwort auf eine Einwohnerfrage von 18.06.2015", Aug. 2015. https://www.hagen.de/buergerinfo/vo020.asp?VOLFDNR=14504&no_mobile=1 *[Archivlink]*

[48] Offene Wahlen Österreich, "Unsere Forderungen". http://offenewahlen.at/forderungen-v1 *[Archivlink]*

[49] Zweckverband KDN – Dachverband kommunaler IT-Dienstleister, "Umsetzungsprojekt Wahlen (in: OZG NRW kommunal – Onlinedienste zum Onlinezugangsgesetz)". https://ozg.kdn.de/umsetzungsprojekte/details/wahlen *[Archivlink]*

[50] kdvz Rhein-Erft-Rur, "Warum ein Standard für Offene Wahldaten?". https://offenewahldaten.de/warum-ein-standard-fuer-offene-wahldaten/ *[Archivlink]*

[51] S. Schmitz, "Zweites Zusammenfinden der Projektgruppe", Aug. 2019. https://offenewahldaten.de/2019/08/21/zweites-zusammenfinden/ *[Archivlink]*

[52] Projektgruppe, "Konzept: Ein Standard für Offene Wahldaten!", Aug. 2018. https://docs.google.com/document/d/1yG9sK3Hd7x8GzzeZFHeMNgTZ6it39h5Cyw9NAQKZ_Cc/edit *[Archivlink]*

[53] kdvz Rhein-Erft-Rur, "Fein-Spezifikation der Dateiformate: Allgemein". https://offenewahldaten.de/dateiformate/allgemein/ *[Archivlink]*

[54] ——, "Fein-Spezifikation der Dateiformate". https://offenewahldaten.de/dateiformate/ *[Archivlink]*

[55] ——, "Datensatzbeschreibung Kandidaten V0.3". https://offenewahldaten.de/wp-content/uploads/2020/11/Datensatzbeschreibung_Kandidaten_V0-3.pdf *[Archivlink]*

[56] Geodatenzentrum, Stadt Wuppertal, vertreten durch den Oberbürgermeister, "Gebietsgliederungen". https://www.wuppertal.de/microsite/geoportal/gebietsgliederungen/index.php *[Archivlink]*

[57] kdvz Rhein-Erft-Rur, "Fein-Spezifikation der Dateiformate: Straßen". https://offenewahldaten.de/dateiformate/strassen/ *[Archivlink]*

[58] O. Hartig und J. Pérez, "An initial analysis of Facebook's GraphQL language", 2017. http://repositorio.uchile.cl/handle/2250/169110 *[Archivlink]*

[59] M. Mauky, "GraphQL als Alternative zu REST", 2017. https://www.doag.org/formes/pubfiles/9716835/Manuel-Mauky-GraphQL-als-Alternative-zu-REST.pdf *[Archivlink]*

[60] "Code using GraphQL". https://graphql.org/code/ *[Archivlink]*

[61] H. Butler, M. Daly, A. Doyle, S. Gillies, T. Schaub, und T. Schaub, "The GeoJSON Format", RFC 7946, Aug. 2016. https://rfc-editor.org/rfc/rfc7946.txt *[Archivlink]*

[62] OpenGIS Consortium, Inc., "OpenGIS Simple Features Specification For SQL Revision 1.1", OpenGIS Consortium, Inc., OGC 99-049, Mai 1999. https://portal.ogc.org/files/?artifact_id=829 *[Archivlink]*

[63] S. E. Battersby, M. P. Finn, E. L. Usery, und K. H. Yamamoto, "Implications of web mercator and its use in online mapping", *Cartographica: The International Journal for Geographic Information and Geovisualization*, Band 49, Nr. 2, S. 85–101, 2014. https://doi.org/10.3138/carto.49.2.2313

[64] V. Agafonkin, *Leaflet API reference*. https://leafletjs.com/reference-1.7.1.html *[Archivlink]*

[65] *Tangram Syntax Reference: Sources*. https://tangrams.readthedocs.io/en/latest/Syntax-Reference/sources *[Archivlink]*

[66] M. Bostock und C. Metcalf, "The TopoJSON Format Specification", 2013. https://github.com/topojson/topojson-specification *[Archivlink]*

[67] Kreis Euskirchen, Der Landrat, "Wahlen: Weitere Bekanntmachungen zur Kommunalwahl 2020", 2020. https://www.kreis-euskirchen.de/politik/wahlen/index_31373.php *[Archivlink]*

[68] Wikipedia contributors, "Geospatial PDF — Wikipedia, The Free Encyclopedia", 2020. https://en.wikipedia.org/w/index.php?title=Geospatial_PDF&oldid=957410524

[69] Stadt Hagen, Der Oberbürgermeister (Einreichender: J. Bücker), "Vorschlag einer Fraktion 0423/2019: Bereitstellung der Stadt-, Wahl- und Stimmbezirke als Open Data", Mai 2019. https://www.hagen.de/buergerinfo/vo020.asp?VOLFDNR=19095&no_mobile=1 *[Archivlink]*

[70] Arbeitsgruppe „AG Geokom.NRW" der Kommunalen Spitzenverbände in NRW und des Landes NRW, "Aufbau einer europäischen Geodateninfrastruktur (INSPIRE) Umsetzung in NRW: Handlungsempfehlung für die Kommunen (Version 2.1)", Mär. 2016. https://www.geoportal.nrw/sites/default/files/Kommunale_Betroffenheit-2015_V_2-1.pdf *[Archivlink]*

[71] W. Pokojski und P. Pokojska, "Voronoi diagrams – inventor, method, applications", *Polish Cartographical Review*, Band 50, Nr. 3, S. 141 – 150, 01 Sep. 2018. https://doi.org/10.2478/pcr-2018-0009

[72] JoshC, "(Antwort auf) Constrained Voronoi polygons QGIS", Mai 2019. https://gis.stackexchange.com/a/323013 *[Archivlink]*

[73] QGIS Development Team, *QGIS Geographic Information System*, QGIS Association, 2020. https://www.qgis.org

[74] A. Graser und V. Olaya, "Processing: A Python Framework for the Seamless Integration of Geoprocessing Tools in QGIS", *ISPRS International Journal of Geo-Information*, Band 4, Nr. 4, S. 2219–2245, Okt. 2015. http://dx.doi.org/10.3390/ijgi4042219

[75] Stadt Hagen, Der Oberbürgermeister, "Europawahl – Wahlbezirk – GeoGrafik". http://wahlergebnisse.stadt-hagen.de/prod/EW2019/05914000/html5/geografik_145_6_.html *[Archivlink]*

[76] ——, "Wahlen und Statistik – Kommunalwahlbezirke". https://www.hagen.de/web/de/fachbereiche/fb_sp/fb_sp_01/fb_sp_0111/wahlen_und_statistik.html *[Archivlink]*

[77] ——, "Das tagesaktuelle Straßenverzeichnis und die Hauskoordinaten der Stadt Hagen". https://www.hagen.de/web/de/fachbereiche/fb_sp/fb_sp_01/fb_sp_0110/strassenverzeichnis.html *[Archivlink]*

[78] *QGIS User Guide — GRASS GIS Integration*. https://docs.qgis.org/3.16/en/docs/user_manual/grass_integration/grass_integration.html

[79] D. Grover und H. P. Kunduru, *ES6 for Humans*. Apress, 2017. https://doi.org/10.1007/978-1-4842-2623-0

[80] "Webpack". https://webpack.js.org/

[81] "What is the file 'package.json'?", Aug. 2011. https://nodejs.org/en/knowledge/getting-started/npm/what-is-the-file-package-json/ *[Archivlink]*

[82] A. Chaudhuri, P. Vekris, S. Goldman, M. Roch, und G. Levi, "Fast and Precise Type Checking for JavaScript", *Proc. ACM Program. Lang.*, Band 1, Nr. OOPSLA, Okt. 2017. https://doi.org/10.1145/3133872

[83] M. Mathews und Beitragende, "JSDoc". https://github.com/jsdoc/jsdoc

[84] JS Foundation, "ESLint — Find and fix problems in your JavaScript code". https://eslint.org/

[85] The Polymer Project, "LitElement — A simple base class for creating fast, lightweight web components". https://lit-element.polymer-project.org/

[86] A. Mehlsen, "Weightless — High quality web components with a small footprint.". https://weightless.dev/

[87] G. Aisch, "chroma.js". https://gka.github.io/chroma.js/

[88] S. Georget, "geostats". https://github.com/simogeo/geostats

[89] K. Xiang, "node-csvtojson". https://github.com/Keyang/node-csvtojson/tree/4fcda1029bb8f6190f04779a847ed2b94a8cbbf5

[90] "Extending Leaflet: Class Theory". https://leafletjs.com/examples/extending/extending-1-classes.html *[Archivlink]*

[91] "Extending Leaflet: Handlers and Controls". https://leafletjs.com/examples/extending/extending-3-controls.html *[Archivlink]*

[92] G. Qiu und J. Chen, "Web-based 3D map visualization using WebGL", in *2018 13th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, 2018, S. 759–763. https://doi.org/10.1109/ICIEA.2018.8397815

[93] J. Gaffuri, "Toward Web Mapping with Vector Data", in *Geographic Information Science*, N. Xiao, M.-P. Kwan, M. F. Goodchild, und S. Shekhar, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, S. 87–101. https://doi.org/10.1007/978-3-642-33024-7_7

[94] *Tangram Syntax Reference: Import*. https://tangrams.readthedocs.io/en/master/Syntax-Reference/import/ *[Archivlink]*

[95] B. Liu, "Protomaps". https://protomaps.com/ *[Archivlink]*

[96] *Tangram Overviews: Styles Overview*. https://tangrams.readthedocs.io/en/master/Overviews/Styles-Overview/ *[Archivlink]*

[97] T. Horbiński und D. Lorek, "The use of Leaflet and GeoJSON files for creating the interactive web map of the preindustrial state of the natural environment", *Journal of Spatial Science*, Band 0, Nr. 0, S. 1–17, 2020. https://doi.org/10.1080/14498596.2020.1713237

[98] *Tangram Syntax Reference: Draw*. https://tangrams.readthedocs.io/en/master/Syntax-Reference/draw/#interactive *[Archivlink]*

[99] M. A. North, "A Method for Implementing a Statistically Significant Number of Data Classes in the Jenks Algorithm", in *Proceedings of the 2009 Sixth International Conference on Fuzzy Systems and Knowledge Discovery - Volume 01*, ser. FSKD '09. USA: IEEE Computer Society, 2009, S. 35–38. https://doi.org/10.1109/FSKD.2009.319

[100] G. F. Jenks, "Optimal data classification for choropleth maps", Lawrence, University of Kansas, 1977.

[101] Points Unknown, "08 - Introduction to Web Mapping with Election Data", 2019. https://pointsunknown.nyc/qgis/web%20mapping/mapbox/2019/11/07/09_WebmappingElectionData.html#creating-a-graduated-point-interactive-map *[Archivlink]*

[102] *Tangram API Reference: Javascript API*. https://tangrams.readthedocs.io/en/master/API-Reference/Javascript-API/#events *[Archivlink]*

[103] R. Friberg, “Interactive Mapping with Tangram”, Dez. 2016. https://www.mapzen.com/blog/tangram-interactivity/ *[Archivlink]*

[104] Chart.js Contributors, “Chart.js — Simple yet flexible JavaScript charting for designers & developers”. https://www.chartjs.org/

[105] B. Camper, *Antwort auf Issue #410 ’is it possible to „select“ building on click?!’*, Sep. 2016. https://github.com/tangrams/tangram/issues/410#issuecomment-248407040 *[Archivlink]*

[106] ——, *Chat-Nachricht zur Hervorhebung von Objekten*, Mär. 2017. https://gitter.im/tangrams/tangram-chat?at=58d92619f22385553ddf9c13 *[Archivlink]*

[107] T. Schröder, L. Neumann, und M. Tschirsich, “Analyse einer Wahlsoftware”, 2017. https://www.ccc.de/system/uploads/230/original/PC-Wahl_Bericht_CCC.pdf *[Archivlink]*

Erklärung

Ich erkläre hiermit, dass ich die Bachelorarbeit selbstständig angefertigt und keine anderen als die angegebenen und bei Zitaten kenntlich gemachten Quellen und Hilfsmittel benutzt habe.

Hagen, den 21. Dezember 2020

Unterschrift