Technische Universität Dresden
Fakultät Informatik

Bachelorarbeit
im Studiengang Medieninformatik

Visualisierung räumlich-zeitlicher Inhalte aus Wikidata

Visualization of spacio-temporal data from Wikidata

Vorname, Name: Georg Wild
Anschrift: Bergstraße 78, 01069 Dresden
E-Mail: georg.wild@mailbox.tu-dresden.de

Studiengang: B.Sc. Medieninformatik
Matrikelnummer: 3705395

Verantwortlicher Hochschullehrer: Prof. Dr. rer. nat. Sebastian Rudolph
Betreuer: Dr. rer. pol. Markus Krötzsch
Eingereicht am 4. September 2014

Zusammenfassung

Das Projekt Wikidata bietet neue Möglichkeiten zur algorithmischen Interpretation komplexer relationaler Wissensstrukturen. Der immer weiter wachsende Umfang dieses Datenbestands verlangt nach Möglichkeiten, Teilmengen der Daten in einem menschenfreundlichen Format zu präsentieren, das größere Zusammenhänge erkennbar macht. Mit dem Ziel, die zeitliche und räumliche Verortung umfangreicher Datenmengen intuitiv begreifbar zu machen, werden wir im Verlauf dieser Arbeit einen Entwurf für eine geeignete Datenvisualisierung erarbeiten und diesen anschließend in Form einer interaktiven Webanwendung umsetzen.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Diese Arbeit beschäftigt sich damit, wie die großen Mengen semantischer Daten aus Wikidata mittels Datenvisualisierung in Formen gebracht werden können, die größere Zusammenhänge erkennbar machen. Anhand eines Fallbeispiels wird ein Konzept für eine Visualisierung erarbeitet und davon ausgehend eine interaktive Webanwendung implementiert. Dabei stehen die Faktoren Performanz und Verarbeitbarkeit großer Datenmengen im Vordergrund.

1.1 Die Befreiung der freien Daten

Wikipedia ist heute für viele kaum mehr wegzudenken. Die freie online Enzyklopädie wird von Alexa auf Platz 6 der meistbesuchten Websites eingestuft und von ihren Besuchern durchschnittlich mehr als dreimal täglich aufgerufen [2]. Die Informationen in Wikipedia sind in einem textbasierten Format gespeichert. Für den Menschen ist dies eine sehr naheliegende Methode zur Konservierung von Wissen, für eine Maschine repräsentiert es dagegen eher eine semantikfreie Aneinanderreihung von codierten Symbolen. Ein direkter Zugriff auf die enorme Menge strukturierter Daten, welche die mittlerweile über 32 Millionen Artikel [34] beherbergen ist nicht möglich [31]. Zwar gibt es erfolgreiche Ansätze, diese Daten zu extrahieren und nutzbar zu machen, jedoch sind diese technisch bedingt mit Aufwand verbunden und fehleranfällig.

Aus diesem Grund wurde das Projekt Wikidata ins Leben gerufen. Dabei handelt es sich um eine freie Wissensdatenbank, die über ihre Website[1] von jedem Besucher bearbeitet werden kann. Die Informationen werden intern in einem semantischen Datenmodell gespeichert, das einen direkten und robusten Zugriff auf spezifische Teilmengen der Daten erlaubt. Die Anwendungsmöglichkeiten dieser Daten sind vielfältig und das Potenzial ist längst nicht ausgeschöpft.

[1] http://www.wikidata.org/

Obwohl das Projekt noch am Anfang steht, umfasst Wikidata bereits heute über 43 Millionen Aussagen zu mehr als 15 Millionen Dingen[2]. Aufgrund des wachsenden Zuspruchs aus der Community ist vorstellbar, dass der Datenbestand in der Zukunft noch auf ein Vielfaches anwachsen wird. Mit dem erklärten Ziel, ein allgemeines Repositorium für alle Wikimedia Projekte und Dritte zu sein [33], sind der Menge nahezu keine Grenzen gesetzt. Die Vorstellung einer erschöpfenden Erfassung aller künstlerischen Werke, aller geschichtlichen Begebenheiten, politischen Entscheidungen, aller bekannten Molekülstrukturen, aller Krankheiten und deren Erreger etc. führt vor Augen, welches Ausmaß ein solcher Datenbestand annehmen könnte. Allein die Musikdatenbank Discogs umfasst beispielsweise mehr als 120 Millionen Titel [12].

Dieser wachsende Umfang stellt eine große Herausforderung dar. Kaum ein Mensch kann aus dem Betrachten einer gigantischen Menge Rohdaten vernünftige Schlüsse ziehen. Das Problem offenbart sich bereits bei der Betrachtung von Einträgen mit vielen Aussagen. Diese erscheinen momentan als eine viele Seiten lange Aneinanderreihung von Fakten ohne klar erkennbare Sortierung. Ein unbedarfter Nutzer wird sich bei der Betrachtung vielleicht fragen, wofür man diesen ganzen Aufwand überhaupt betreibt, wenn dieselben Informationen doch an anderen Orten – zum Beispiel auf Wikipedia – schon viel übersichtlicher verfügbar sind. Auch beispielsweise eine tabellarische Auflistung aller Katzen in Wikidata[3] [22] könnte diese Zweifel wahrscheinlich nicht ganz aus der Welt schaffen.

Aus diesem Grund müssen Lösungen für eine menschenfreundliche Präsentation des angesammelten Wissens erarbeitet werden. Die Stärke von Wikidata liegt in den maschinenlesbaren Daten und dies müssen wir uns zunutze machen. Während sich einzelne Daten gezielt extrahieren lassen, können größere Teilmengen durch Programme analysiert oder für spezifische Anwendungsfälle verarbeitet werden.

Aufschlussreicher als unsere Katzenliste wäre zum Beispiel eine Anwendung, die daraus extrapoliert, ob die starke Verbreitung von Katzenbildern im Internet zu einem Anstieg in der Katzenpopulation geführt hat. Die Aussagekraft einer solchen Berechnung ist natürlich sehr fragwürdig, wenn der verwendete Datensatz derart klein[4] ist. Angaben zu Raum und Zeit sind dagegen schon in viel größerem Umfang vorhanden und interessant, da sie konkrete Punkte der Weltgeschichte markieren. Wir möchten herausfinden, welche Aussagekraft diese Daten bekommen, wenn man sie zusammenführt und so präsentiert, dass das große Ganze sichtbar wird.

Ziele dieser Arbeit sind daher Entwurf und Umsetzung einer intuitiven interaktiven Visualisierung, die geeignet ist, zeitliche und räumliche Verortung von Teilmengen des Datenbestands aus Wikidata sichtbar zu machen. Dadurch soll die Möglichkeit geschaffen werden, Trends und Bewegungen im Lauf der Zeit zu erkennen und verschiedene Datensätze miteinander zu vergleichen.

[2] Stand 19. August 2014, berechnet mit Wikidata Toolkit [21] *Dump Processing Example*

[3] http://tools.wmflabs.org/wikidata-todo/autolist.html?q=CLAIM[31:146]

[4] Stand 23. August 2014, 33 Katzen

1.2 Gliederung

Im folgenden Kapitel wird zunächst ein einführender Überblick zu Wikidata und den damit zusammenhängenden Technologien gegeben. Danach leiten wir in Kapitel 3 aus der Aufgabenstellung konkrete Anforderungen für unsere Anwendung ab. Aus diesen Anforderungen werden wir ein Konzept für eine mögliche Visualisierung erarbeiten (Kapitel 4) und im darauf folgenden Schritt implementieren (Kapitel 5). In Kapitel 6 beurteilen wir die erreichten Ergebnisse und geben einen Ausblick auf mögliche Weiterentwicklungen.

Die folgenden Beiträge leisten wir im Verlauf dieser Arbeit:

- Ausarbeitung einer Visualisierung für große dreidimensionale Punktmengen
- Entwicklung einer effizienten Datenstruktur für die darzustellenden Daten
- Extraktion der erforderlichen Daten aus Wikidata
- Umsetzung der Visualisierung als interaktive Webanwendung

Kapitel 2

Wikidata

Dieses Kapitel gibt eine kurze Einführung in Wikidata und die zugrunde liegenden semantischen Technologien. Wer mit Wikidata vertraut ist, kann dieses Kapitel ohne spätere Verständnisprobleme überspringen.

Wikidata ist für Daten das, was Wikimedia Commons für Medien ist – eine Wissensdatenbank, die einen zentralen Zugriff auf Daten ermöglicht, welche an unterschiedlichen Stellen Verwendung finden können. Dinge sind in Wikidata *Items* und Aussagen über Items werden als *Statements* bezeichnet. Jedes Item besitzt eine IRI, über die es eindeutig referenziert werden kann. Im Gegensatz zu einem Label ermöglicht dies, insbesondere bei maschineller Kommunikation über eine Sache, Missverständnisse auszuschließen. Zudem werden durch die Referenz auf ein einzelnes, zentral gewartetes Datum widersprüchliche Instanzen an unterschiedlichen Stellen vermieden. Weiterhin ermöglicht die Zusammenführung an einem Ort leichten Zugriff auf die angesammelten Daten, maschinenverständlich und fertig zur Analyse und Weiterverarbeitung.

2.1 Semantische Daten

Der Begriff Semantic Web beschreibt ein Konzept von maschinenverständlichen Daten, wobei der in diesem Zusammenhang gebräuchliche Begriff der Semantik etwas unscharf ist. In erster Linie geht es darum, Datenstrukturen zu schaffen, mit denen Beziehungen zwischen Daten auf Bedeutungsebene zum Ausdruck gebracht werden können. [18]

Die Wissensrepräsentation semantischer Technologien basiert auf Tripeln, die eine Struktur aus Subjekt, Prädikat und Objekt verwenden. Die Bedeutung der Aussage „Garfield ist eine fiktive Katze“ ist für Menschen leicht zu erschließen, während durch den kognitiven Prozess automatisch Informationen ins Bewusstsein gerufen werden, die mit den verwendeten Begriffen vernetzt sind. Ein Computer kann das nicht. Wenn man die Aussage aber etwas formalisiert, könnte sie sinngemäß lauten: `Q767120 P31 Q15831457`

(in Worten: *„Garfield“*[1] *„ist eine“*[2] *„fiktive Katze“*[3]). Die Zeichenfolgen sind nun lediglich Referenzen auf die jeweiligen Begriffe. Die Referenzen verweisen auf Knoten, unter denen wiederum alle zu den jeweiligen Begriffen verfügbaren Informationen in gleicher Struktur gespeichert sind. Damit lässt sich beispielsweise ableiten, dass Garfield ein fiktives Tier ist, denn `Q15831457 P279 Q3542731` (*„fiktive Katze“ „ist Unterklasse von“*[4] *„fiktives Tier“*[5]). Diese Erkenntnis mag trivial erscheinen, ist für den Computer aber nur möglich, weil die Daten miteinander verknüpft sind. Eine solche Wissensrepräsentation stammt aus dem Forschungsgebiet der künstlichen Intelligenz. Man könnte diese Art der Datenmodellierung also auch als den Versuch bezeichnen, assoziative Eigenschaften menschlicher Kognition in einer Datenstruktur nachzubilden.

2.2 Das Datenmodell von Wikidata

Wikibase, die Wikidata zugrunde liegende Software, verwendet ein elaboriertes Datenmodell, das über einfache Tripel hinausgeht und erlaubt, komplexe Wissensstrukturen zu repräsentieren [13]. In einem kurzen Abriss werden hier die für das allgemeine Verständnis nötigen Merkmale dieses Datenmodells vorgestellt.

Alle atomaren Daten sind *Values*. Ein Value kann vom Typ *DataValue* oder *Entity* sein. DataValues repräsentieren einen Wert von einem bestimmten *Datatype*[6]. Von besonderer Bedeutung für diese Arbeit sind die Datatypes *Time* (Zeitangaben) und *Globe coordinate* (Geographische Koordinaten). Alle Entities sind durch eine IRI referenzierbar. Items und *Properties* sind Entities. Properties stellen die Prädikate dar, die in einer Aussage über Dinge verwendet werden können, während eben jene Dinge durch Items repräsentiert werden. In unserem „Garfield ist eine fiktive Katze“ Beispiel verweisen also `Q767120` und `Q15831457` auf Items und `P31` auf eine Property. Die Aussagen selbst werden in Statements organisiert. Statements speichern ihr Subjekt, Quellenangaben und die eigentliche Behauptung (*Claim*) in sogenannten *Snaks*. Ein Snak ist ein Property-Value Paar, welches eine Aussage über ein Subjekt repräsentiert. Jeder Claim besitzt einen *MainSnak* und eine beliebige Anzahl weiterer Snaks, die auch als *Qualifier* bezeichnet werden. Der MainSnak bezieht sich auf das Subjekt des Statements, wohingegen sich die Qualifier auf die Aussage beziehen, die der MainSnak über das Subjekt trifft. Einfacher gesagt konkretisieren sie den Claim. Die Aussage *„Garfield“ (die Figur) „kommt vor in“*[7] *„Garfield“*[8] *(der Comic)* liese sich beispielsweise mit dem Qualifier *„als“*[9] *„Protagonist“*[10] genauer eingrenzen.

[1] https://www.wikidata.org/entity/Q767120 *„Garfield“ (character)*
[2] https://www.wikidata.org/entity/P31 *„instance of“*
[3] https://www.wikidata.org/entity/Q15831457 *„fictional cats and other felines“*
[4] https://www.wikidata.org/entity/P279 *„subclass of“*
[5] https://www.wikidata.org/entity/Q3542731 *„fictional animal character“*
[6] https://www.wikidata.org/wiki/Special:ListDatatypes
[7] https://www.wikidata.org/entity/P1441 *„present in work“*
[8] https://www.wikidata.org/entity/Q72533 *„Garfield“ (Comic)*
[9] https://www.wikidata.org/entity/P794 *„as“*
[10] https://www.wikidata.org/entity/Q215972 *„protagonist“*

2.3 Eine kurze Geschichte

Zum Zeitpunkt dieser Arbeit nähert sich Wikidata dem zweiten Geburtstag. Gemäß dem Paradigma „release early, release often“ wird das Projekt in enger Zusammenarbeit mit der Community fortwährend weiterentwickelt. Zum Launch im Oktober 2012 war der Funktionsumfang noch gering, doch bereits wenige Monate später konnte der erste signifikante Erfolg verbucht werden: die Zentralisierung der Sprachlinks auf Wikipedia, die zu anderssprachigen Versionen eines Artikels führen. Diese waren zuvor noch redundant auf jeder Seite jeder Sprache einzeln enthalten, wodurch ein hoher Speicher- und Wartungsaufwand entstand. Während Wikidata zu diesem Zeitpunkt schon mehr als 3 Millionen Items enthielt, ist die Zahl mittlerweile auf 15.388.369 [28] angewachsen. Trotz großer Erfolge steht das Projekt noch an seinem Anfang. Das Potenzial für eine wichtige und einflussreiche Plattform ist vorhanden, aber der Erfolg hängt von der Akzeptanz der diversen Wiki-Communities ab.

2.4 Wikidata Toolkit

Wikidata Toolkit [21] ist eines der vielen Projekte, die sich um Wikidata gebildet haben. Da es für diese Arbeit von Bedeutung ist, soll es hier noch besonders erwähnt werden.

Wie in Abbildung 2.1 dargestellt, gibt es verschiedene Möglichkeiten auf die Inhalte von Wikidata zuzugreifen, welche je nach Anwendungsfall unterschiedliche Vor- und Nachteile haben [19]. Für gezielte Datenzugriffe eignet sich die Verwendung einer Web API. Für komplexere Berechnungen empfiehlt sich die Verwendung der regelmäßig von Wikidata bereitgestellten Dumps des gesamten Datenbestands. Wikidata Toolkit ist eine Java-Bibliothek zur Vereinfachung dieser Aufgabe. Sie übernimmt das Herunterladen und Parsen der Dumps, bildet die Wikibase Datenstruktur in Java ab und ermöglicht damit die Verarbeitung für verschiedenste Anwendungszwecke.

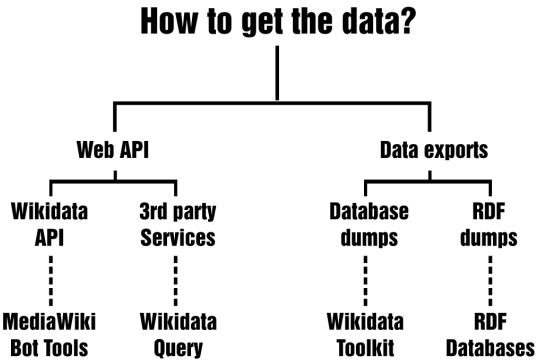

Abbildung 2.1: Flowchart zum Datenexport [20]

Kapitel 3

Anforderungsanalyse

3.1 Aufgabenstellung

Ziel dieser Arbeit ist es, eine Anwendung zu entwickeln, die erlaubt, aus den reichhaltigen Daten in Wikidata interessante Teilmengen auszuwählen und deren räumliche und/oder zeitliche Verortung miteinander zu vergleichen. Diese Aufgabe umfasst verschiedene Aspekte:

(1) Entwurf einer Visualisierung von Ort und Zeit von Datenpunkten. Die Punkte sollen einzeln oder aggregiert repräsentiert werden. Eine gleichzeitige Visualisierung mehrerer Ergebnismengen soll möglich sein.

(2) Vorverarbeitung der Daten mit Wikidata Toolkit. Orts- und Zeitangaben, die über andere Items indirekt mit einem Datenpunkt verknüft sind, sollen explizit gemacht werden. Das Ergebnis der Vorverarbeitung kann in einem reduzierten Datenformat gespeichert werden, das nur noch Orte, Zeiten und Item-IDs enthält.

(3) Zuordnung von Item-Mengen zu Klassen oder anderen einfachen Abfragekriterien. Es soll durch die Vorverarbeitung ein Index erstellt werden, durch den alle Daten eines bestimmten Typs ohne komplexe Datenbankabfragen direkt abgerufen werden können.

(4) Implementierung einer Webanwendung zur Visualisierung der vorberechneten Daten. Nutzer sollen verschiedene Datenmengen und verschiedene Orts- und Zeitquellen auswählen können.

Das implementierte System soll mindestens die Visualisierung vorberechneter Daten mit vorgegebener Auswahl für Ort und Zeit erlauben. Wünschenswerte Erweiterungen sind:

- Freiere Filterung der anzuzeigenden Datenmengen
- Freiere Auswahl der verwendeten Orts- und Zeitangaben, unabhängig voneinander
- Vergleichende Darstellung mehrerer Suchanfragen

3.2 Abgeleitete Anforderungen

Ein entscheidender Faktor für die Anforderungen an unsere Arbeit ist die Quantität der Daten. Mensch[1] zählt mit über 2 Millionen Instanzen zu den umfangreichsten Klassen [32], daher gehen wir im Folgenden von mehreren Millionen Items aus.

(A1) Entwurf einer interaktiven Visualisierung, die für diese Größenordnung geeignet ist. Sie soll die übergebenen Daten möglichst intuitiv repräsentieren. Idealerweise sollte ein Nutzer ohne jegliche Vorkenntnisse sofort in der Lage sein, die Visualisierung semantisch zu erfassen. Weiterhin sollte die Visualisierung möglichst flexibel für unterschiedliche Datensätze geeignet sein.

(A2) Entwurf einer effizienten Datenstruktur als Eingabe für die Visualisierung. Diese soll auch eine Filterung der Daten ermöglichen, wie sie in Aspekt (3) der Aufgabenstellung beschrieben wird.

(A3) Erzeugung dieser Datenstruktur aus den Wikidata Daten mit Wikidata Toolkit.

(A4) Entwicklung der Visualisierung als interaktive Webanwendung. Die Webanwendung soll ausschließlich aus offenen Webstandards und freien Softwarekomponenten aufgebaut sein. Der Verzicht auf proprietäre und lizenzgebundene Software ermöglicht eine hohe Wiederverwendbarkeit der Anwendung.
Wir möchten dem Anspruch einer guten Usability gerecht werden. Die Bedienung der Anwendung soll selbsterklärend sein und keine speziellen Vorkenntnisse oder Instruktionen erforderlich machen. Die Visualisierung soll im Client berechnet werden, so dass die Anwendung auch ohne einen leistungsstarken Server auf größere Nutzerzahlen skalieren kann. Trotzdem soll die Anwendung auf aktuellen Systemen performant laufen und ein gutes Nutzungserlebnis bieten. Das Interface ist für das Widescreen Format zu optimieren, da dieses heute auf Desktop Rechnern die größte Verbreitung findet [29].

3.3 Abgrenzungskriterien

Die folgende Liste mit Problemstellungen, welche wir explizit nicht behandeln wollen, dient zur weiteren Eingrenzung der angestrebten Lösung.

- Die Visualisierung ist auf umfangreiche Datensätze ausgelegt. Eine sinnvolle Repräsentation kleiner Datenmengen ist wünschenswert, aber nicht Ziel dieser Arbeit.
- Es werden keine Annahmen bezüglich der Vollständigkeit oder sonstiger Mängel der Daten getroffen. Die Daten sollen genau in der Form repräsentiert werden, wie sie in Wikidata vorliegen.

[1] https://www.wikidata.org/wiki/Q5

- Die Webanwendung arbeitet auf Grundlage von statischen, vorberechneten Daten. Eine Vorberechnung zur Laufzeit, beispielsweise durch die Eingabe eines Querys, ist nicht möglich. Weiterhin ist es die Aufgabe des Betreibers der Anwendung, zu entscheiden, welche Datensätze bereitgestellt werden sollen und diese zu generieren. Es gibt kein automatisiertes Verfahren zur Auswahl „interessanter“ Datensätze.
- Veraltete Webbrowser, welche die verwendeten Technologien nicht unterstützen, werden nicht berücksichtigt.

3.4 Verwandte Arbeiten

Es existieren zahlreiche Bibliotheken (*Libraries*) und Spezifikationen, die beim Erstellen unserer Webanwendung nützlich sein können. Aufgrund der Vielzahl von Tools, kann hier nur eine exemplarische Übersicht gegeben werden.

Basierend auf *JSON* (Javascript Object Notation [14]) spezifiziert *GeoJSON* [8] ein Austauschformat zur Codierung geometrischer Datenstrukturen. *D3* [7] ist eine Javascript Library für datenbasierte DOM Manipulation. *three.js* [25] ist eine Javascript Library für 3D Rendering. *Highcharts* [4] ist ein Javascript Framework zum Erstellen von interaktiven Diagrammen. *Google Fusion Tables Heatmaps* [15] sind serverseitig gerenderte Heatmap Layer für Google Maps. *heatmap.js* [17] ist eine Javascript Library zum Zeichnen von Heatmaps.

Weiterhin gibt es eine Reihe von fertig entwickelten Lösungsansätzen für ähnliche Problemstellungen. Eine Auswahl soll hier kurz vorgestellt werden.

Wikidata tempo-spacial display [23] (Abbildung 3.1) nutzt einen Zeitstreifen und Marker auf einer Karte zur Darstellung von Ereignissen. Bei Selektion einzelner Ereignisse wird durch Transparenzeffekte die relative zeitliche Position der anderen Marker auf der Karte kenntlich gemacht. Eine Repräesentation einzelner Datenpunkte durch Marker ist jedoch schon ab mehreren hundert Punkten nicht mehr praktikabel und für unsere Größenordnung ungeeignet.

Deutsche Nationalbibliothek - Data Explorer [24] (Abbildung 3.2) zeigt in einem Grid aus unterschiedlich transparenten Hexaedern die Datendichte der für einen gewählten Zeitraum existenten Orts- und Bevölkerungsdaten. Dazu werden dynamisch GeoJSON Daten von einem Backend ausgeliefert und als Layer über einer Weltkarte dargestellt. Die Neuberechnung verhält sich jedoch teilweise etwas träge und ist auf die serverseitige Berechnung angewiesen.

Wikipedia Map [35] (Abbildung 3.3) ist eine auf dem RENDER Toolkit basierende Visualisierung aller Artikel aus Wikipedia, die mit geometrischen Daten verknüpft sind. Dies ist eine dynamische Darstellung einer sehr großen Punktmenge. Sie enthält jedoch keinen Zeitbezug und bietet wenig Interaktionsmöglichkeiten.

CartoDB [9] ist eine cloudbasierte Mapping Engine, die laut Herstellerangaben in der Lage ist, „Millionen von Datenpunkten“ zu visualisieren und mit *Torque* [30] zudem Unterstützung für zeitbasiertes Mapping von Daten bietet. CartoDB ist ein kommerzielles Produkt und erlaubt eine kostenlose Nutzung bis maximal 50MB Kartendaten, wobei ein Wasserzeichen eingeblendet wird.

Die hier gelisteten Anwendungen verstoßen allesamt gegen eine oder mehrere Aspekte unserer Anforderungen. Als Inspirationsquelle sind die aufgeführten Beispiele dennoch sehr wertvoll. Aus den unterschiedlichen Arbeiten können wir geeignete Ansätze entnehmen und zu einem ganzheitlichen Konzept vereinen.

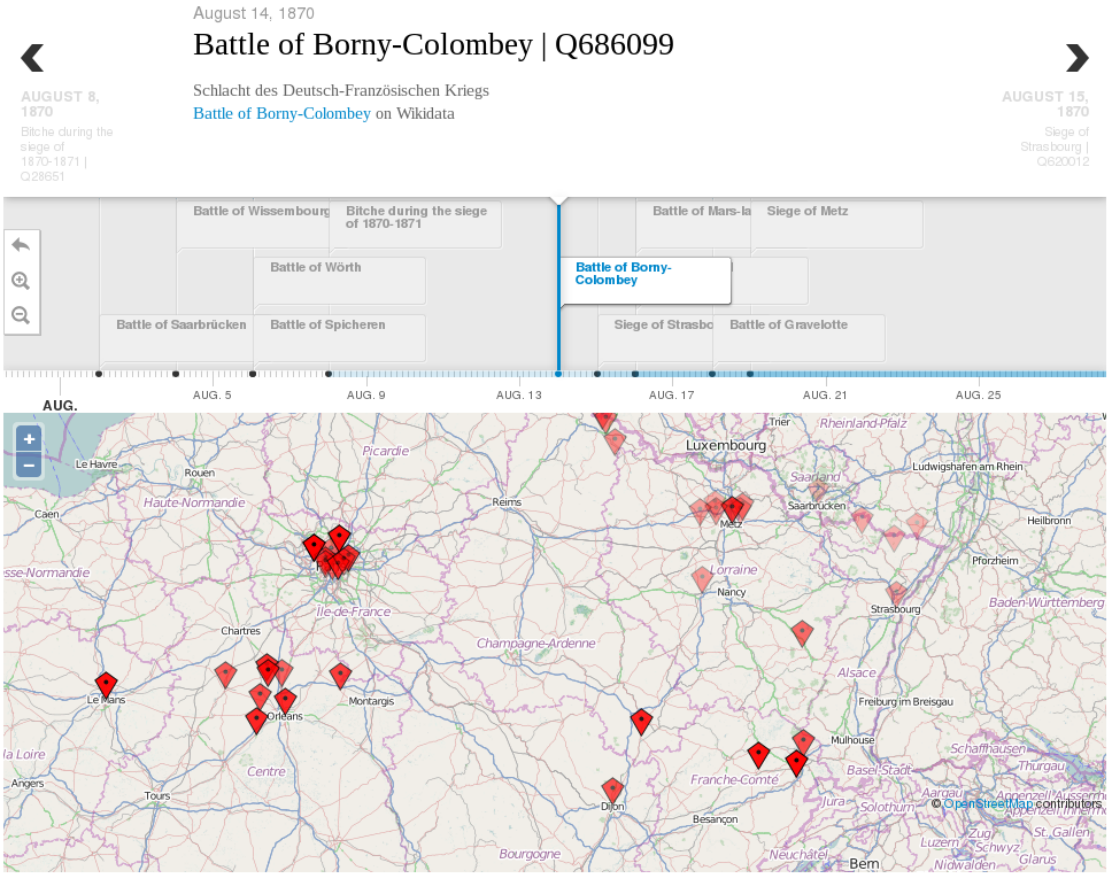

Abbildung 3.1: Wikidata tempo-spacial display [23]

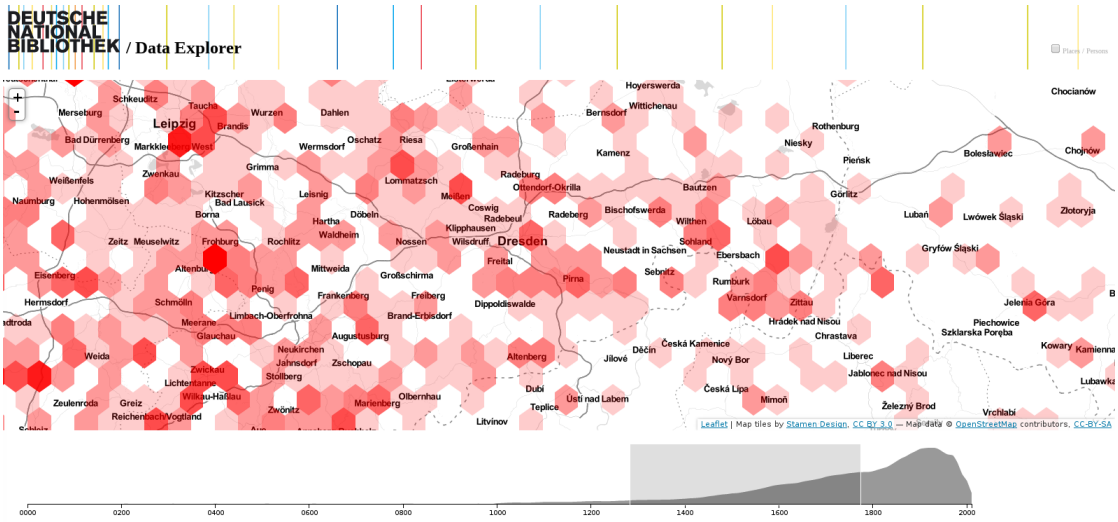

Abbildung 3.2: Deutsche Nationalbibliothek - Data Explorer [24]

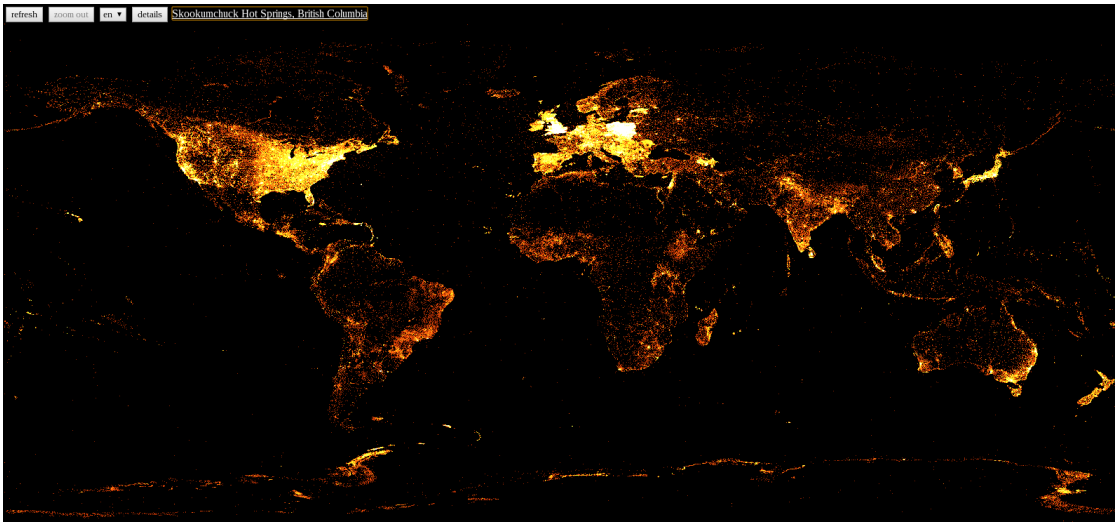

Abbildung 3.3: Wikipedia Map [35]

Kapitel 4

Konzept und Design

Ziel dieses Kapitels ist die gestalterische Konzeption unserer Anwendung. Der Schwerpunkt liegt auf dem in A1 geforderten Entwurf einer interaktiven Visualisierung für große Datensätze. Weiterhin werden wir ein Design für das Nutzerinterface entwickeln, da dieses eng mit der Interaktionsmetapher der Visualisierung zusammenhängt.

4.1 Datenbeschaffenheit

Zunächst ist es sinnvoll, sich mit der Form der darzustellenden Daten zu befassen. Das betrifft in erster Linie Quantität, aber auch qualitative Merkmale wie Format und Präzision.

Da wir tempospatiale Daten darstellen möchten, kommen nur Items infrage, die Statements mit Claims bezüglich räumlicher und zeitlicher Verortung besitzen. Eine Einschränkung auf zeitliche und räumliche Datentypen allein ist jedoch problematisch, denn diese können unterschiedliche Ausprägungen annehmen, welche wiederum besondere Anforderungen an die Visualisierung stellen. Beispielsweise ist es ein Unterschied, ob wir Zeitspannen und Raumflächen verarbeiten möchten, oder nur einzelne Punkte in Zeit und Raum.

Die Einschränkung auf eine bestimmte Teilmenge der verfügbaren Daten bietet Vor- und Nachteile. Einerseits ist es für die Optimierung der Visualisierung nützlich, die Beschaffenheit der Daten möglichst genau zu kennen, andererseits wird die Visualisierung durch eine solche Falloptimierung weniger flexibel. Entscheidend ist hier, einen guten Mittelweg zu finden.

Da sich der Datenbestand von Wikidata noch in einem jungen Stadium befindet [31], müssen wir zudem nicht nur die Frage stellen, welche Daten wir verwenden möchten, sondern auch welche Informationen überhaupt in welchem Umfang zur Verfügung stehen.

Für den Entwurf der Visualisierung und im späteren Verlauf auch für die Entwicklung der Anwendung werden die Geburts- und Sterbedaten aller Menschen[1] als exemplarischer

[1] Instanzen von „Mensch“ (`Q5`)

Datensatz dienen. Um zum Beispiel die Geburten sinnvoll in Raum und Zeit verorten zu können, sind sowohl ein Geburtsdatum als auch ein Geburtsort erforderlich. Nur etwas mehr als 500.000 von über 2 Millionen Menschen in Wikidata besitzen beide Angaben (siehe Abschnitt 5.4.6). Das ist jedoch immer noch eine große Anzahl und für die Entwicklung ausreichend. Darüber hinaus sind für Menschen auch einige leicht verfügbare und verständliche Filtereigenschaften denkbar, wie zum Beispiel Geschlecht, Berufsfeld, Religion oder Ethnie.

4.2 Ziele der Visualisierung

Die menschliche visuelle Wahrnehmung kann geometrische Formen und Farben wesentlich leichter verarbeiten als eine tabellarische Auflistung von Zeichen und Zahlen. Deshalb geht es darum, die relativ abstrakten Daten in eine Form zu bringen, die leichter interpretierbar ist. Das ist der elementare Sinn und Zweck einer Visualisierung [27].

Mit dieser Visualisierung könnten bestimmte Trends und Entwicklungen sichtbar gemacht werden. Auch kann sich offenbaren, wie sich das Ausmaß der Datenerfassung in verschiedenen Regionen und Epochen verteilt. Durch eine kombinierte Darstellung mehrerer Datensätze könnte beispielsweise auch die Verteilungen verschiedener Bevölkerungsgruppen verglichen werden. Es ist darüber hinaus vorstellbar, dass die erzielten Resultate zu unerwarteten neuen Anwendungsmöglichkeiten der Visualisierung führen können.

4.3 Entwurf der Visualisierung

Die zu visualisierenden Datenpunkte besitzen zwei Werte für die geografische Position (Breiten- und Längengrad) sowie einen Wert für die zeitliche Position. Die Daten bilden also im Prinzip eine dreidimensionale Matrix, in der jeder Zeitpunkt eine geografische Ebene repräsentiert, oder auch jeder Punkt der Ebene eine Zeitlinie. Eine dreidimensionale Visualisierung scheint sich daher zwar anzubieten, jedoch nehmen wir an, dass sich eine akzeptable Umsetzung als sehr schwierig erweisen könnte. Insbesondere wegen der bei Computergrafik technisch bedingten Projektion auf die Anzeigeebene bestünde die Gefahr, dass die Darstellung sehr unübersichtlich wird und damit an Aussagekraft verliert. Aus diesem Grund entscheiden wir uns stattdessen für eine Kombination von zwei Visualisierungskomponenten, die in ähnlicher Weise in „Deutsche Nationalbibliothek - Data Explorer“ [24] zum Einsatz kommt.

Wie in Abbildung 4.1 skizziert, verwenden wir für die Zeitwerte einen Graph, der auf der y-Achse die Anzahl der vorhandenen Geodaten anzeigt. Dafür ist eine Aggregation der Zeitwerte notwendig, die wir aufgrund des großen Wertebereichs auf ganze Jahre festlegen. Die Geodaten werden auf eine navigierbare Karte geplottet. Die Daten werden in einem Raster (*Grid*) aggregiert, dessen Auflösung an die Zoomstufe der Karte angepasst wird. Die relative Auflösung des Grids kann durch den Benutzer verändert

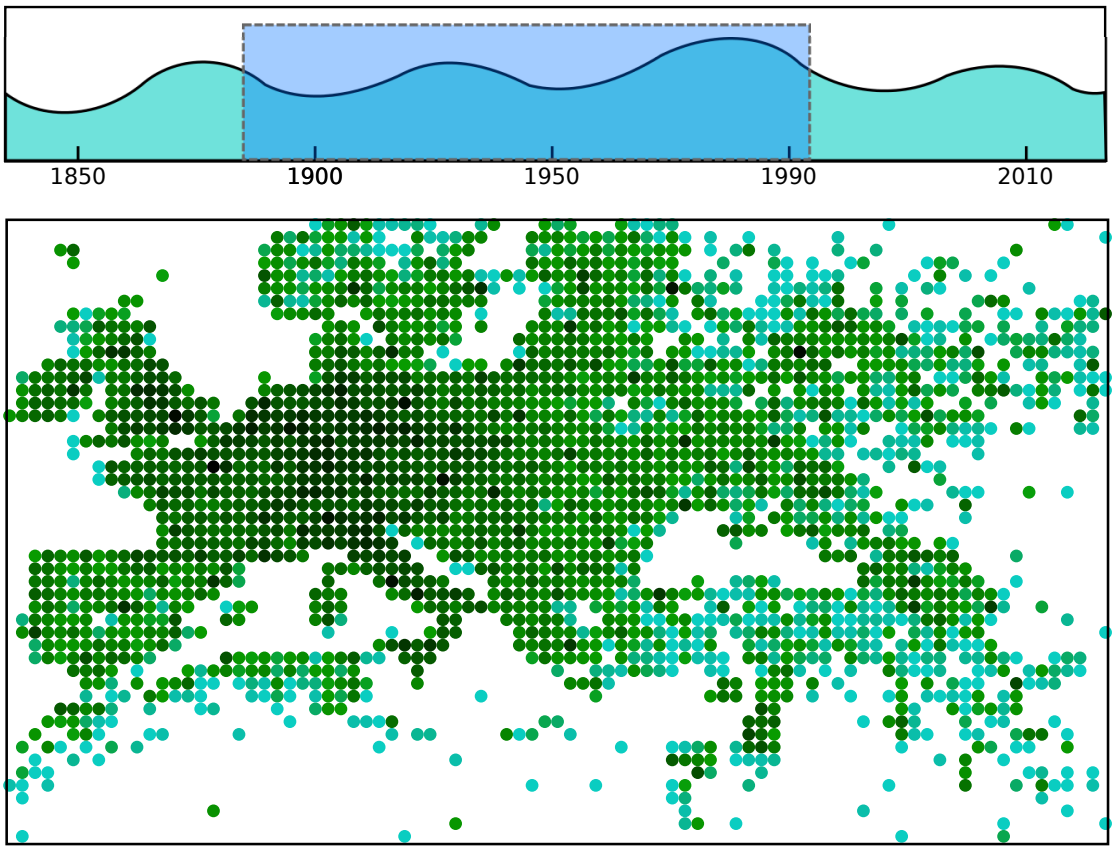

Abbildung 4.1: Konzept der Visualisierungskomponenten

werden. Jede Zelle des Grids, die mindestens einen Datenpunkt aggregiert hat, wird als Punkt auf der Karte dargestellt. Die Anzahl der aggregierten Datenpunkt wird dabei in jedem Punkt farblich codiert. Dabei stehen helle Farben für niedrige und dunkle Farben für hohe Werte. Durch individuelle Farbtöne wird eine unterscheidbare Überlagerung meherer Datensätze möglich. Zudem informiert ein Tooltip beim Überfahren der Punkte auf der Karte über die genaue Anzahl der aggregierten Datenpunkte.

Beide Komponenten sind interaktiv und erlauben durch Auswahl eines Zeitintervalls, respektive eines bestimmten Kartenbereichs, die Selektion eines Quaders im dreidimensionalen Werteraum. Dabei hängt die Anzeige beider Komponenten immer von dieser Selektion ab. Das heißt, der Zeitgraph zeigt immer nur die Summen der sich im aktuell sichtbaren Kartenausschnitt befindlichen Werte und die Karte zeigt nur die Werte innerhalb des selektierten Zeitintervalls. Um das Verständnis zu verbessern, wird ne-

ben der Visualisierung in Worten eine kurze, aussagekräftige Beschreibung der aktuellen Selektion angezeigt. Durch Veränderung der Selektion wird dadurch für den Nutzer in Echtzeit die zusammenhängende Veränderung der Daten ersichtlich. Auch abspielbare Animationen durch Automatisierung solcher Verschiebungen sind denkbar.

Die zur Verfügung stehenden Datensätze können frei zu- und weggeschaltet werden, was die Visualisierung einer Überlagerung aller aktivierten Datensätze erlaubt. Zusätzlich kann die Datenmenge der Datensätze über individuelle Filter eingeschränkt werden. Die entsprechenden Daten vorausgesetzt, wäre es also beispielsweise möglich zu untersuchen, ob die Wohnorte von Katzenbesitzern tatsächlich räumlich und zeitlich mit Rückgängigmachungen von Änderungen in Wiki-Artikeln korrelieren.[2]

4.4 Interface

Das Interface setzt sich aus mehreren Komponenten zusammen. Neben Zeitgraph und Karte, die im vorherigen Abschnitt beschrieben wurden, befindet sich das *Steuerpanel*, das die Einstellungen für Selektion und Filterung der verfügbaren Datensätze beinhaltet. Weiterhin gibt es eine Legende in der die Beschreibung der aktuellen Selektion und andere Informationen angezeigt werden können.

Das Layout der Komponenten soll den Kontrollfluss der Anwendung leicht begreiflich machen. Der Benutzer kann zu jeder Zeit auf Steuerpanel, Zeitgraph und Karte einwirken. Wir gehen davon aus, dass die Betrachtung der Veränderungen auf der Karte bei Manipulation des Zeitintervalls einen häufigeren Anwendungsfall darstellt als anders herum. Daher ist der Zeitgraph über der Karte positioniert, um eine Konzeption als primäre Steuerkomponente zu evozieren. Beide werden jedoch von dem Filterpanel beherrscht, das von linker Seite auf sie einwirkt. Das entspricht der natürlichen Leserichtung westlicher Kulturkreise. Die Pfeile in Abbildung 4.2 sollen dieses Konzept verdeutlichen.

Die Anordnung und visuelle Gruppierung der Bedienelemente helfen zu erfassen, mit welchen Elementen die Visualisierung kontrolliert werden kann (Gesetz der Nähe und Gleichheit [6]). Im Gegensatz dazu ist die Legende als rein informative Anzeige etwas dezenter gestaltet und belegt den freien Platz an der linken Seite. Der Großteil der Fläche bleibt somit für die Karte, die den Hauptbestandteil der Visualisierung darstellt.

[2] https://commons.wikimedia.org/wiki/File:IM_IN_UR_WIKI_RVRTING_UR_EDITS_lolcat.jpg

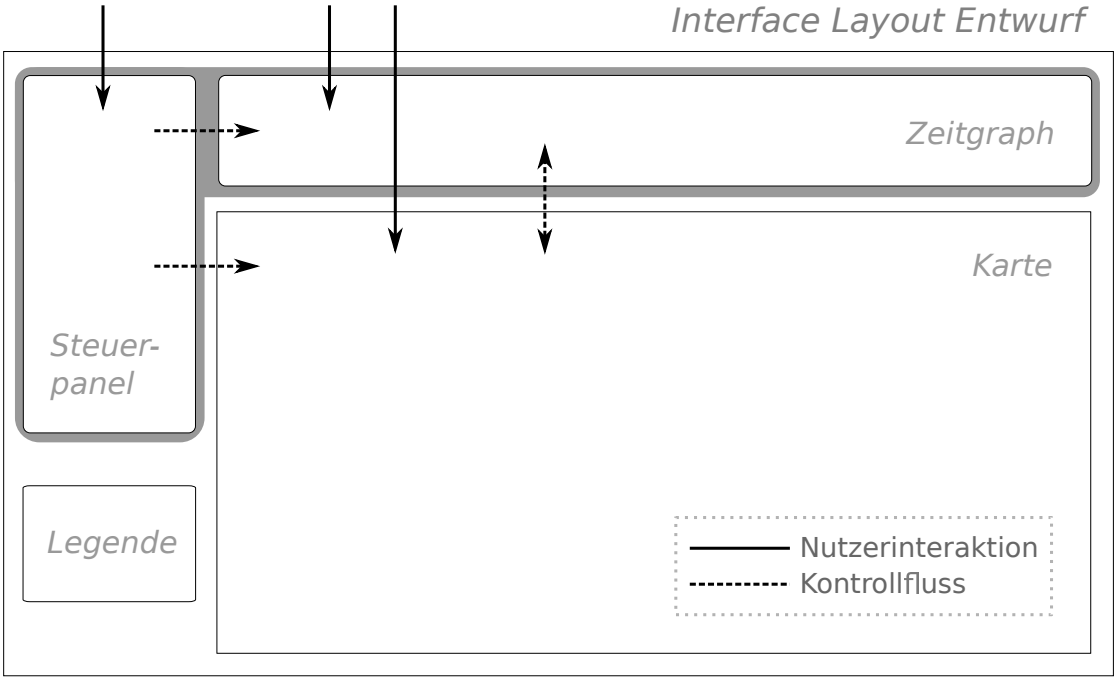

Abbildung 4.2: Einfacher Layout Entwurf

Kapitel 5

Umsetzung

Dieses Kapitel beschreibt die implementierte Webanwendung und die dafür getroffenen Designentscheidungen. Nach einer einführenden Erklärung der Architektur wird detaillierter auf die einzelnen Bestandteile der Anwendung eingegangen.

Bei der Umsetzung hat sich herausgestellt, dass der vollständige Funktionsumfang der in Kapitel 4 entworfenen Visualisierung für eine Realisierung im Rahmen dieser Bachelorarbeit zu umfangreich war. Diese Diskrepanzen werden in Kapitel 6 angesprochen.

Unsere Webanwendung trägt den Namen *ViziData*. Die in Abbildung 5.1 dargestellte Version ist auf http://ow.ly/B3atw[1] abrufbar. Zusätzlich ist der Quellcode der aktuellen Version auf GitHub frei verfügbar[2].

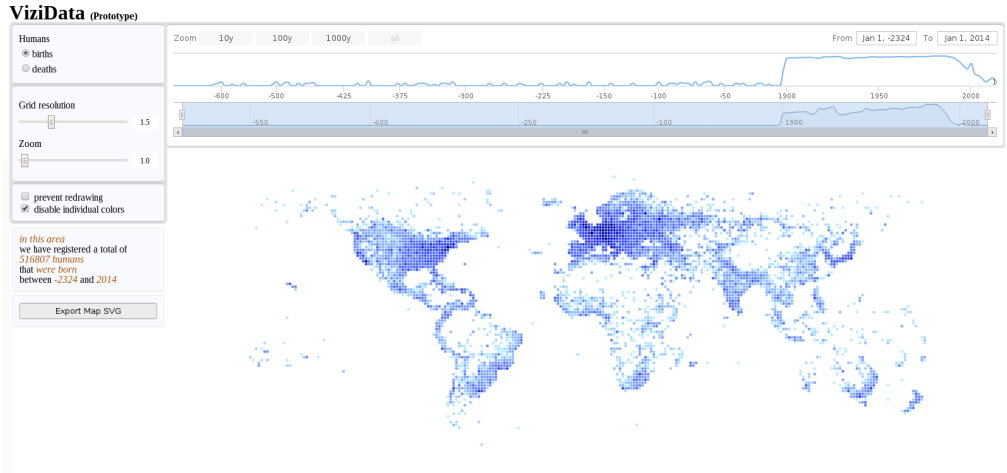

Abbildung 5.1: ViziData

[1] http://wwwpub.zih.tu-dresden.de/~s5219191/vizidata/
[2] https://github.com/gordelwig/ViziData

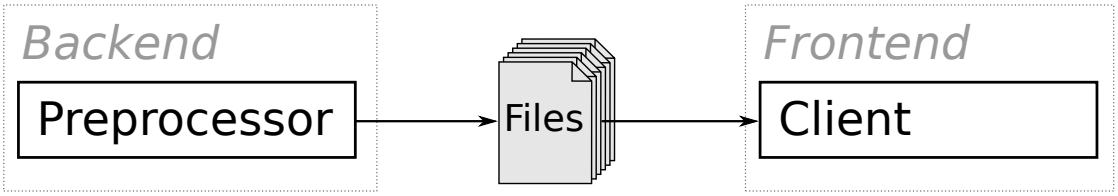

Abbildung 5.2: Architektur-Schema

5.1 Architektur im Überblick

Wie in Abbildung 5.2 zu erkennen ist, lässt sich die Anwendung in zwei Bestandteile unterteilen, *Preprocessor* und *Client*. Der Preprocessor ist dafür zuständig, aus dem gesamten Wikidata Dump einen Datensatz für die Anwendung zu berechnen und diesen als JSON auf das Dateisystem zu schreiben. Der Client stellt die eigentliche Visualisierungsanwendung dar. Er liest diese JSON-Daten ein und arbeitet von da an komplett eigenständig ohne Backend, mit der SVG Export Funktionalität (siehe 5.4.5) als einzige Ausnahme. Dies entspricht der in (A4) geforderten Visualisierungsberechnung im Client und bietet außerdem den Vorteil, dass Verzögerungen durch HTTP Übertragungen bei Nutzerinteraktionen vermieden werden. Diese könnten sich andernfalls bei schlechter Verbindungsqualität negativ auf das Interaktionserlebnis auswirken.

Verwendete Technologien

Der Preprocessor läuft in Java und verwendet Wikidata Toolkit für die Verarbeitung der Datendumps.

Der Client ist eine moderne Webanwendung, basierend auf HTML5, CSS3, Javascript und SVG. Verwendete Bibliotheken sind jQuery [26] für allgemeine DOM Manipulation, D3 [7] zum Zeichnen des Kartenplots und Highcharts [4] für den interaktiven Zeitgraphen.

5.2 Struktur der vorberechneten Eingabedaten

Die Datenstruktur bietet unter Berücksichtigung von (A2) einen ausgewogenen Kompromiss aus Kompaktheit und effizienter Iterierbarkeit. Die Organisation der Daten erfolgt in *Datagroups*, die wiederum mindestens ein *Dataset* enthalten. „Menschen“ ist ein Beispiel für eine Datagroup, „Geburten“ ein mögliches Dataset. Zusätzlich besitzt jede Datagroup genau ein *Properties* Objekt, das die verfügbaren Eigenschaften zu jedem Item enthält, das in mindestens einem Dataset der Datagroup enthalten ist. Dataset Einträge bestehen indes nur aus ihrer Verortung im Werteraum und einer Referenz auf den entsprechenden Eintrag der Properties. Diese Auslagerung der Properties dient zur

Vermeidung von Duplikaten der Daten zu Items, die in mehreren Datasets vorkommen. Die Properties ermöglichen eine effiziente und flexible Umsetzung der Filterfunktionalität, die in Aspekt (3) der Aufgabenstellung gefordert wird. Datagroup, Properties und jedes Dataset werden in individuellen Dateien abgelegt. Die Datei der Datagroup wird dadurch sehr kompakt, da sie nur noch Metadaten enthält, sowie die Information, in welchen Dateien die eigentlichen Daten zu finden sind. Anstatt immer den vollständigen Datenbestand laden zu müssen, erhält der Client dadurch die Möglichkeit, selektiv Daten anzufordern und in seine speicherinterne Repräsentation der Datenstruktur einzufügen.

Die Daten der Datasets sind in einer mehrstufigen Baumstruktur angeordnet. Als Indizes auf oberster Ebene dienen die Jahreszahlen. Wir gruppieren die Koordinatenwerte anhand der Längengrade in *Tiles* mit einheitlicher Breite. Die Geodaten werden zusammen mit der Property-Referenz unterhalb des zugehörigen Jahres gemäß ihrer geographischen Länge in dem entsprechendem Tile abgelegt. Durch diese Gruppierung kann die Iteration der meisten Geodaten, die sich außerhalb des aktuell dargestellten Kartenausschnittes befinden, vermieden werden. Aufgrund der Form der Erde und der Annahme, dass die Betrachtung von Regionen mit geringer Datendichte ein seltener Anwendungsfall sein wird, rechnen wir bei einer zusätzlichen Unterteilung nach Breitengraden mit keiner nennenswerten Verbesserung. Außerdem ist beispielsweise bei Betrachtung einer kleinen Region in Südamerika die Iteration sehr schnell, auch wenn alle Daten der nordamerikanischen Ostküste ebenfalls in den Tiles enthalten sind.

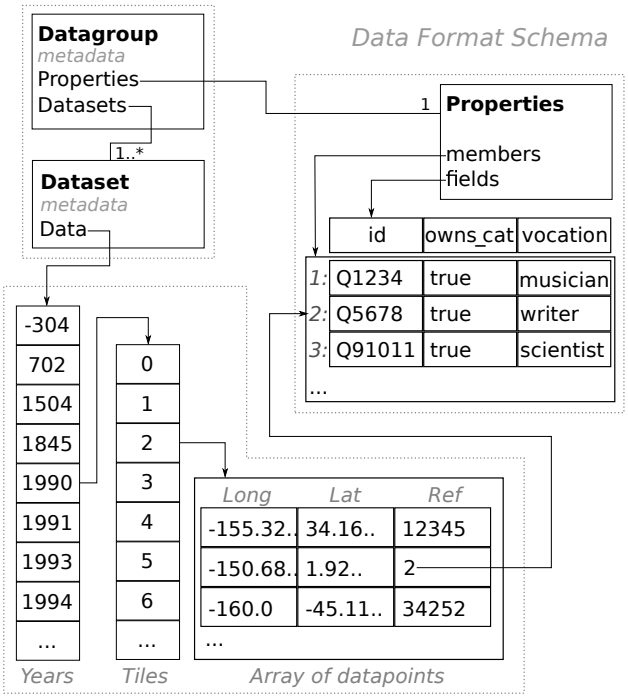

Abbildung 5.3: Datenstruktur

Ein weiterer relevanter Ansatz wäre der *R-Tree*, eine balancierte Indexstruktur, die überwiegend in Datenbanksystemen und Geo-Informationssystemen angewendet wird. Durch eine gleichverteilte Gruppierung in Datenregionen ermöglicht der R-Tree effizientes Suchen in einem mehrdimensionalen Raum [16]. Da wir jedoch stets ganze Regionen aggregieren, brächte diese Struktur keinen großen Vorteil gegenüber dem von uns gewählten Ansatz mit Spalten einheitlicher Breite. Wir vermuten, dass eine mögliche Effizienzsteigerung durch den R-Tree zu gering wäre, um die durch ihn eingeführte Erhöhung der Datenstrukturkomplexität zu rechtfertigen.

5.3 Preprocessor

Der Preprocessor erzeugt die im vorigen Abschnitt erläuterte Datenstruktur. Dazu werden mithilfe der Bibliothek Wikidata Toolkit alle Daten aus Wikidata durchsucht und die benötigten Informationen in Hashmaps gesammelt. Gemäß unserem Beispiel sammeln wir alle Zeitpunkte und Orte für Geburten und Tode. Die Statements über Geburts- und Sterbeorte verweisen auf Items, also müssen wir die gesuchten Koordinaten aus den verknüpften Items gewinnen. Daher sammeln wir auch alle Globe coordinate Werte von Items in einer Hashmap. So können wir den Datenpunkten in unserer Ausgabe explizit die Koordinatendaten zuweisen. Nach Beendigung des Suchlaufs verwenden wir Instanzen der Klasse `JSONObject` aus dem Paket `org.json` [11], um die Daten der einzelnen Datasets, die Datagroup, und die Properties zu erzeugen und anschließend in Dateien zu schreiben.

5.4 Client

Der Client präsentiert ein einzelnes Dataset gemäß der aktiven Filter- und Darstellungsoptionen. Er kennt und listet die verfügbaren Datagroups und deren Datasets, die bei Bedarf asynchron in den Speicher geladen werden.

5.4.1 Bedienelemente

Die Interaktionsmöglichkeiten des Benutzers lassen sich in drei Bereiche gruppieren (vgl. Abbildung 4.2). Auf linker Seite befindet sich das Steuerpanel. Dort kann das aktive Dataset ausgewählt werden. Darunter befinden sich Schieberegler, mit denen die Auflösung des Aggregationsgrids und die Zoomstufe des angezeigten Kartenbereichs angepasst werden können. Zusätzlich existieren zwei Kontrollboxen zur Veränderung des Anwendungsverhaltens. Mit „prevent redrawing“ wird die automatische Anzeigeerneuerung bei Interaktion deaktiviert, was für eine genauere Inspektion eines gegebenen Visualisierungszustandes nützlich sein kann. Die in den Metadaten eines Datasets enthaltene Farbskala (siehe auch 5.4.4) kann mit „disable individual colors“ durch eine Standardskala ersetzt werden, wodurch die Vergleichbarkeit zweier Datasets verbessert werden kann.

Der Zeitgraph erlaubt die Auswahl eines Zeitintervalls als Filter unserer Datenstruktur. Die Karte ermöglicht schließlich eine Auswahl des Bereichs der anzuzeigenden Geodaten. Die Karte kann mit dem Zeiger verschoben und mit dem Scrollrad gezoomt werden. Die Zoomfunktion agiert dabei analog zu dem entsprechenden Regler im Steuerpanel, von dem stets die aktuelle Zoomstufe der Karte abgelesen werden kann. Ansonsten hat die Navigation der Karte nur Einfluss auf die Karte selbst.

5.4.2 Zeitgraph

Der Zeitgraph wird mit der Highcharts Bibliothek [4] durch ein Stockchart realisiert. Um die Visualisierungsdaten zu berechnen, wird über das gesamte Dataset iteriert und für jedes Jahr werden die darunter liegenden Koordinateneinträge aufsummiert. Als Workaround für einen Softwarebug [5] wird bei Änderung des Datasets das gesamte Widget zerstört und neu initialisiert.

Entgegen der Konzeption in Kapitel 4 wird die Anzeige des Zeitgraphen nicht durch die Gebietsselektion der Karte beeinflusst. Die verwendete Library verhält sich im Umgang mit den aggregierten Zeitdaten weniger flexibel als erhofft. Eine Änderung der Werte führt aufgrund der angewendeten Neuinitialisierung zu einem Verlust des Selektionszustandes. Weiterhin wäre eine Einflussnahme der Karte auf den Zeitgraph in vielen Fällen nicht gut verständlich. Da die Library den Graphen entsprechend der sichtbaren Maximalwerte auf die zur Verfügung stehende Höhe skaliert, würden Änderungen oft kaum auffallen. Daher haben wir uns entschieden, den Zeitgraphen als statische Repräsentation des gesamten Datasets zu verwenden. Durch die visuelle Verbindung zum Steuerpanel wirkt diese Konfiguration nicht unlogisch und wir erwarten daher keine negativen Auswirkungen auf die Usability. Der Schwerpunkt der Visualisierung verlagert sich allerdings noch mehr auf die Karte.

Highcharts ist nicht optimal auf die Anforderungen unserer Webanwendung abgestimmt. Durch die vielen mitgelieferten Features ist die Library anfälliger für fehlerhaftes Verhalten bei großen Datensätzen. Für eine Weiterentwicklung der Anwendung wäre es sinnvoll, die Implementierung auf eine effizientere Komponente mit reduziertem Funktionsumfang umzustellen.

5.4.3 Aggregationskonzept

Die Generierung des *Aggregationsgrids* für die Karte ist ein zentrales Element der Anwendungslogik. Gemäß der aktuellen Selektion relevante Daten werden durch eine Verschachtelung mehrerer Schleifen einer Aggregationsfunktion zugeführt. Wie dem folgendem Pseudocode zu entnehmen ist, beruht die Filterreihenfolge auf dem in Abschnitt 5.2 vorgestellten Datenformat.

```
for (i = minYear to maxYear)
    if (data[i] exists)
        for (j = minTile to maxTile)
            if (data[i][j] exists)
                foreach (c in data[i][j])
                    if (c visible in map)
                        aggregate(c);
```

Die Funktion `aggregate()` bildet die Koordinatendaten in einer Map auf die Indizes der entsprechenden Zellen im verwendeten Aggregationsgrid ab. Existiert der Index schon

in der Map, wird der zugehörige Wert inkrementiert, andernfalls wird er mit dem Wert 1 erzeugt.

Das verwendete Grid wird durch eine einfache Berechnungsvorschrift implizit definiert und indiziert. Bei gegebener Auflösung $r > 0$ können wir für ein Koordinatenpaar (l, b) den Index i berechnen.

$$i = \left(\left\lfloor \frac{b}{r} \right\rfloor * \frac{360}{r} \right) + \left\lfloor \frac{l}{r} \right\rfloor$$

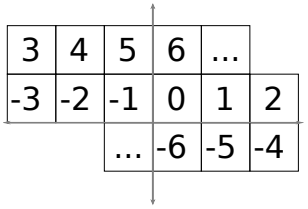

Abbildung 5.4: Grid

So entsteht ein in Abbildung 5.4 skizziertes Raster, dessen Zelle mit Index $i = 0$ an dem Punkt $(0, 0)$ des Koordinatensystems liegt. Mit aufsteigendem Index wandert die Zellposition von links nach rechts zeilenweise nach oben, bei absteigendem Index in die entgegengesetzte Richtung. Daraus ergeben sich folgende Umkehrungen für die Zurückführung der Indizes auf Koordinaten:

$$l = \left(\left(i - \frac{c}{2} \right) \bmod c \right) - \frac{c}{2} \qquad b = \left(\frac{i + \frac{c}{2}}{c} \right) * r \qquad \text{mit } c = \frac{360}{r}$$

5.4.4 Karte

Die Karte ist ein SVG Element, das die Zellen des Aggregationsgrids als `<circle>` Elemente enthält. Bei jeder Veränderung wird die Karte geleert und neu berechnet. Die Zeichenfunktion erhält das vom Aggregator erstellte Mapping von Indizes auf die Anzahl der in der Zelle aggregierten Werte. Daraus berechnen wir ein Array, das für jede Zelle den Wert und die aus dem Index berechneten Koordinaten enthält. Für die korrekte Positionierung ist in unserem Fall außerdem eine Anpassung erforderlich. Da die y-Achse in SVG von oben nach unten verläuft, invertieren wir die Breitengrade. Zusätzlich normalisieren wir sie auf einen Wertebereich von $[0, 180]$ und die Längengrade auf einen Wertebereich von $[0, 360]$. Mit Hilfe von D3 [7] wird für jedes Element dieses Arrays ein `<circle>` Element erstellt. Diese werden mit den Daten verknüpft, gemäß der Koordinaten platziert, eingefärbt und mit Event Listenern versehen.

Die Farbe der Kreise codiert die Anzahl der aggregierten Datensätze. Ein Dataset enthält in den Metadaten die Definition einer individuellen Farbskala. Diese wird definiert durch die Angabe von minimalen und maximalen Werten k_{min}, k_{max} für die Kanäle Rot, Grün und Blau. Der aktuelle Wert v wird relativ zum Höchstwert v_m in eine Farbe aus der Farbskala übersetzt. Folgendermaßen leiten wir den jeweiligen Kanalwert k logarithmisch oder linear aus v ab:

$$k_{log} = k_{max} - \left(\frac{k_{max} - k_{min}}{\log v_m} \right) \log v \qquad k_{lin} = k_{max} - \left(\frac{k_{max} - k_{min}}{v_m} \right) v$$

Ist v_{max} unverhältnismäßig größer als der Durchschnittswert, führt eine lineare Skala tendenziell dazu, dass verschiedene Werte im unteren Bereich kaum unterscheidbar sind.

Eine logarithmische Skala verhindert umgekehrt, dass extrem hohe Werte angemessen aus den übrigen Werten hervorstechen. Wir finden einen guten Kompromiss für diese Problematik, indem wir zwei der Kanäle logarithmisch skalieren und einen linear.

Befindet sich der Mauszeiger über einem `<circle>` Element, wird dieses farblich hervorgehoben und ein Tooltip eingeblendet. Der Tooltip wird neben dem Mauszeiger positioniert und informiert über die Anzahl der aggregierten Daten und die Koordinatenposition des Kreises.

5.4.5 Legende

Die Legende befindet sich unterhalb des Steuerpanels. Ein kurzer Text informiert in Worten über die aktuelle Darstellung und benennt die Gesamtsumme aller dargestellten Datenpunkte. Darunter befindet sich ein Button, der erlaubt, den aktuellen Kartenzustand als SVG zu exportieren. Da mit Javascript alleine keine Dateien an den User-Agent ausgeliefert werden können, erzeugen wir in einem unsichtbaren iFrame ein Formular. Darüber senden wir die Kartendaten an ein PHP-Script, welches die Daten als Dateidownload zurückliefert. Diese Implementierung basiert auf einem Tutorial [3].

5.4.6 Performanz Evaluation

Die beschriebene Implementierung ist das Resultat eines iterativen Entwicklungsprozesses. In Hinblick auf eine ungenügende Performanz wurden wiederholt Komponenten ausgetauscht und strukturelle Änderungen vorgenommen. Dadurch konnten wir die Zeitdauer für die Erzeugung des Aggregationsgrids erheblich verkürzen.

Für ungefähre quantitative Benchmarks nutzen wir das Javascript Date Objekt. Unser Testsystem ist eine virtuelle Maschine in VirtualBox [10] Version 4.3.12. Es besitzt 5120MB Hauptspeicher und zwei Kerne eines Intel Core i5-2500k mit einer Taktfrequenz von 3,4GHz. Für die Messung der Werte verwenden wir den Chromium Browser Version 34.0.1847.116 unter Linux Mint 16. Bei der Messung kam es immer wieder zu größeren Schwankungen und teilweise zu Verbesserungen nach mehrmaliger Ausführung einer Operation. Mögliche Ursachen dafür sind zum einen Hintergrundlasten auf dem Testsystem und zum anderen Caching- und Optimierungsmaßnamen der verwendeten Javascript Engine. Zum Testen verwenden wir das Dataset „Geburten von Menschen“ mit insgesamt 516.807 Datenpunkten. Die Größe der unkomprimierten JSON Datei beträgt 15,1 MiB. Auf dem Zeitgraph bleibt der gesamte Wertebereich selektiert, um die größtmöglichsten Punktmengen zu erhalten.

Wir wählen drei Kartenregionen in unterschiedlichen Zoomstufen und generieren dort das Aggregationsgrid mit drei verschiedene Einstellungen des Sliders für die Auflösung. Dadurch erhalten wir die folgenden Messwerte:

	Auflösung	**0,5**	**1,5**	**4,5**
Welt `zoom: 1` 516.807 Geburten	Punkte	12.896	3.786	779
	Aggregation (ms)	227	177	177
	Zeichnen (ms)	389	110	23
	Gesamt (ms)	633	292	254
Europa `zoom: 3` 358.090 Geburten	Punkte	21.083	6.846	1.554
	Aggregation (ms)	118	182	150
	Zeichnen (ms)	527	173	43
	Gesamt (ms)	657	388	227
Uruguay `zoom: 6` 1.205 Geburten	Punkte	227	225	208
	Aggregation (ms)	8	16	19
	Zeichnen (ms)	13	12	14
	Gesamt (ms)	28	30	37

Wir gliedern die Berechnung des Aggregationsgrids in die Teilschritte Aggregation und Zeichnen. Die Zeitdauer der Aggregation ist von vielen Faktoren abhängig und deswegen kaum in eine Gesetzmäßigkeit überführbar. Die Zeichengeschwindigkeit scheint dagegen annähernd linear von der Menge der verarbeiteten `<circle>` Elemente abzuhängen. Wir stellen auch fest, dass die Gesamtdauer immer größer ist, als die Summe der beiden Einzelschritte, obwohl bei der Zeitmessung keine Berechnungsschritte übergangen wurden. Mögliche Erklärungen dafür wären laufzeitbedingte Overheads wie Speicherallokation oder Garbage Collection, sowie auch die Summe der durch Konsolenausgaben eingeführten Latenzen.

Wir erachten die Performanz der Anwendung in ihrem derzeitigem Zustand als ausreichend reaktiv für ein interaktives Nutzungserlebnis, sehen hier aber auch Potenzial für mögliche Verbesserungen. Auf diese soll in Kapitel 6.2.2 näher eingegangen werden.

5.4.7 Projektionsproblem

Zur Vereinfachung der Berechnung verwendet die Anwendung eine Rektangularprojektion. Das bedeutet, dass die Geokoordinaten aus den Daten wie kartesische Koordinaten interpretiert werden. Diese Art der Darstellung ist prinzipiell für eine rasterartige Visualisierung geeignet, insbesondere wegen der einfachen Relation zwischen Geokoordinaten und Rasterpunkten. Der Nachteil ist jedoch, dass die Zellen des Aggregationsgrids Gebiete von unterschiedlicher Größe repräsentieren und diese Gebiete trotzdem als gleich große Punkte auf der Karte dargestellt werden.

Um das Ausmaß der dadurch entstehenden Verzerrung zu bemessen, möchten wir für einen Breitengrad, der den Winkelabstand α von seinem nächsten Pol entfernt ist, näherungsweise die Entfernung a zwischen zwei Längengraden berechnen. Dafür betrachten wir die Erde vereinfachend als regelmäßige Kugel mit Radius $r = 6367,4447\text{km}$ [37]. Wir wählen einen beliebigen Punkt auf dem Breitengrad und fällen von dort aus ein

Lot auf die Erdachse. Mit der Erdmitte als dritten Punkt beschreiben wir so ein rechtwinkliges Dreieck mit einer Hypotenuse der Länge r. Das Lot auf die Erdachse lässt sich berechnen mit $b = r\sin(\alpha)$. Bei bekanntem Breitenkreisradius b ergibt sich der Abstand zweier Längengrade leicht durch $a = \frac{2b\pi}{360}$. Damit erhalten wir Formel 5.1

$$a = r\sin(\alpha)\frac{2\pi}{360} \tag{5.1}$$

Der Plot in Abbildung 5.5 zeigt auf der y-Achse für $\alpha \in [0°, 180°]$ den Abstand der Längengrade in Kilometern. Wir erkennen, dass die Größenunterschiede der von den Gridzellen repräsentierten Flächen keinesfalls vernachlässigbar gering sind. Mit einer flächentreuen Projektion könnten die gezeichneten Kreise so skaliert und positioniert werden, dass sie die Größe des jeweiligen Gebietes widerspiegeln. Eine testweise Implementierung der Hammer-Aitov-Projektion verlängert jedoch die Berechnungszeit für das Aggregationsgrid mindestens um den Faktor 2. Aus diesem Grund verzichten wir auf die Anwendung einer Projektion und nehmen in Kauf, dass die einheitliche Größe der Rasterpunkte fälschlicherweise als einheitliche Größe der repräsentierten Regionen interpretiert werden könnte.

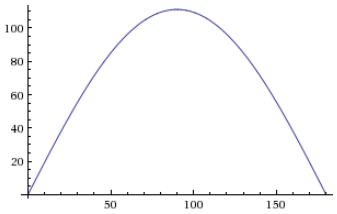

Abbildung 5.5: Formel 5.1 [36]

Eine andere Herangehensweise für dieses Problem wäre die Visualisierung von Datenpunktdichten (zum Beispiel Geburten $* \text{km}^{-2}$) statt Absolutwerten. Durch eine Gewichtung nach der Fläche, die der Aggregationspunkt repräsentiert, könnte die Punktdichte flächentreu dargestellt werden. Die Qualität und Vollständigkeit der Wikidata Daten lässt sich jedoch schwer quantifizieren. Obwohl wir uns in dieser Arbeit nicht mit der Datenqualität auseinandersetzen, gab es Bedenken, dass eventuelle Mängel durch eine implizite Interpretation zu größeren Mängeln in der Repräsentation führen könnten. Deshalb haben wir bei der Konzeption von diesem Ansatz abgesehen. Beispielsweise gibt es viele verstreute Aggregationspunkte, die nur einen Datenpunkt enthalten. Diese könnten irrtümlicherweise den Eindruck erwecken, dass die Geburtenraten mit zunehmender Entfernung vom Äquator grundsätzlich höher werden.

Im Rahmen einer Weiterentwicklung dieser Arbeit könnte man auch darüber nachdenken, ob eine Kombination mehrerer Ansätze zu einer guten Lösung für dieses Problem führen kann. Entscheidend ist in jedem Fall, dass bei Betrachtung der Visualsierung klar ist, was eigentlich konkret dargestellt wird.

Kapitel 6

Fazit

Eine der wesentlichen Schwierigkeiten war das breite Spektrum an thematischen Teilgebieten, die von dieser Arbeit abgedeckt werden. Eine tiefgehende Behandlung aller relevanten Aspekte hätte weit über den herkömmlichen Rahmen einer Bachelorarbeit hinausgeführt. Die richtige Balance und Verteilung der Prioritäten zu finden, so dass die Arbeit zu einem sinnvollen Endergebnis geführt werden konnte, war eine große Herausforderung. Insgesamt kam es bei der Umsetzung in vielen Teilschritten zu unerwarteten Problemstellungen, die einen signifikantem Mehraufwand erforderlich machten. Daraus leitet sich die Diskrepanz zwischen angestrebtem Visualisierungsentwurf und tatsächlich implementiertem Funktionsumfang ab.

In diesem Kapitel werden wir beurteilen, ob diese Arbeit eine Lösung für die Aufgabenstellung darstellt und anschließend einen Ausblick auf mögliche weiterführende Arbeiten geben.

6.1 Lösungsanalyse

Wir haben eine Visualisierung für Ort und Zeit von Datenpunkten entworfen und als Webanwendung umgesetzt. Die Möglichkeit, mehrere Ergebnismengen überlagert darzustellen, ist zwar nicht implementiert, die entworfene Visualisierung ist aber für eine Überlagerung mehrerer Ergebnismengen geeignet. In Kapitel 6.2.1 werden dafür mögliche Umsetzungen beschrieben. Die Daten für die Visualisierung berechnen wir mit Wikidata Toolkit und verketten die Koordinatendaten explizit mit den Datenpunkten. Die Aspekte (1), (2) und (4) der Aufgabenstellung haben wir damit behandelt.

Die nötigen Voraussetzungen für Aspekt (3) wurden in unserer Datenstruktur in Form der referenzierten Properties-Einträge geschaffen. Zum gegebenen Zeitpunkt enthalten die Properties aber lediglich den Identifier des jeweiligen Items. Damit wäre es möglich, weitere Informationen zum jeweiligen Item asynchron von Wikidata abzurufen und anzuzeigen. Für eine Filterung der Datenauswahl ist es aus Performanzgründen jedoch

nicht praktikabel, mehrere Hunderttausend Serverabfragen abzuwarten. Die Funktionalität kann effizient umgesetzt werden, indem der Preprocessor so angepasst wird, dass er die gewünschten Filtereigenschaften aus den Dumps extrahiert und in die Properties einfügt. Dafür wäre es auch sinnvoll zu erarbeiten, wie die Bildung von sehr vielen kleinen Einzelgruppen vermieden werden kann. Mögliche Ansätze dafür wären die Wahl geeigneter Superklassen, wie beispielsweise „scientist“[1] statt „computer scientist“[2], und die Zusammenfassung von Gruppen unter einer bestimmten Mindestgröße in „Sonstige“. Unser Hauptziel in dieser Arbeit war, eine Lösung für Verarbeitung und Repräsentation von sehr umfangreichen Datensätzen zu entwickeln. Da die Filterfunktion effektiv eine Verkleinerung der Datenmengen bewirkt, haben wir sie nicht als integralen Bestandteil dieser Kernproblematik eingestuft. Aufgrund der notwendigen Priorisierung war eine vollständige Implementierung deshalb nicht mehr realisierbar.

Die abgeleiteten Anforderungen konnten erfüllt werden. Der Anspruch einer guten Usability wurde in den Gestaltungsprozess integriert. Da dies jedoch keinen essenziellen Teil der Arbeit darstellt, haben wir im Hinblick auf die verfügbaren Kapazitäten von einer Evaluation durch einen geeigneten Usability-Test abgesehen.

Insgesamt sehen wir in unserer Arbeit eine funktionierende Lösung für die gegebene Problemstellung. Während nicht alle Teilaspekte der Aufgabenstellung vollumfänglich umgesetzt werden konnten, geht unsere Lösung in manchen Fällen auch über die gestellten Anforderungen hinaus. Ein Beispiel dafür ist die dynamische Berechnung des Aggregationsgrids für eine beliebige Rasterauflösung.

6.2 Ausblick

Neben der Integration der erweiterten Filterfunktionalität sind vielzählige Erweiterungen und Verbesserungen der Anwendung vorstellbar. Eine Auswahl soll hier vorgestellt werden.

6.2.1 Funktionsumfang

Karte Die Anwendung stellt aktuell lediglich das berechnete Aggregationsgrid dar. Im Verlauf der Entwicklung wurden verschiedene Ansätze verfolgt, wie man die Daten mit einer geeigneten Kartendarstellung unterlegen kann. Aufgrund unbefriedigender Ergebnisse haben wir uns dann jedoch auf die effiziente Darstellung des Grids konzentriert. Diese Darstellung verliert bei hohen Zoomstufen und wenigen Aggregationspunkten jedoch an Aussagekraft. Die Integration einer Karten-Library wie Leaflet [1] wäre daher sinnvoll.

[1] http://www.wikidata.org/wiki/Q901
[2] http://www.wikidata.org/wiki/Q82594

Zeitgraph als z-Graph Die in dieser Arbeit als Zeitgraph bezeichnete Komponente repräsentiert auf abstrakter Ebene nur eine der drei Dimensionen des Datensatzes. Daher ist sie auch für andere numerische Daten verwendbar, wie zum Beispiel Jahresumsätze von Firmen, Vermögen von Personen oder für IP-Addressen die Anzahl der hochgeladenen Katzenbilder.

Überlagerung mehrerer Datasets Eine wünschenswerte Erweiterung ist die Möglichkeit, mehrere Datasets gleichzeitig anzeigen zu können. Man könnte beispielsweise gemäß der additiven Farbmischung für jeden Punkt einen überlagerten Farbwert berechnen. Ausschlaggebend für eine aufschlussreiche Darstellung ist eine geeignete Wahl der einzelnen Farben, wie zum Beispiel Rot und Grün, was bei Überlagerung zu Gelbtönen führen würde. Eine weniger effiziente Möglichkeit wäre die tatsächliche Übereinanderlegung mehrerer Ergebnisplots unter Verwendung von Transparenzen. Dieser Ansatz hätte den Vorteil, dass man ein Art Crossfader in das Interface einfügen könnte, mit dem die Sichtbarkeit der verglichenen Ebenen fließend verstellt werden kann.

Informationen zu einzelnen Items Es wäre aufschlussreich, weitere Informationen zu den angezeigten Daten zur Verfügung zu stellen. Man könnte beispielsweise die Aggregationspunkte selektierbar machen und für den angewählten Punkt eine Liste der enthaltenen Items zur Verfügung stellen. Ähnlich wie in Autolist [22] könnten diese zunächst durch ihre Identifier repräsentiert werden, bis weitere Informationen dynamisch nachgeladen wurden. Dazu bietet sich natürlich auch ein Link zur Wikidata Seite des jeweiligen Items an.

Verlinkbare Konfiguration Man könnte einen konkreten Selektionszustand der Anwendung verlinkbar machen, indem man die Konfigurationsparameter in die URL encodiert. Dazu müsste auch die Anwendung so angepasst werden, dass sie die übergebenen Parameter bei der Initialisierung übernimmt.

Automatischer Datagroup Scan Momentan müssen die Dateinamen der verfügbaren Datagroups als Parameter im Kopf des Client Javascripts deklariert werden. Eine einfache Erweiterung wäre hier ein PHP-Script, das ein Verzeichnis nach Datagroup Dateien durchsucht und eine Liste der Dateinamen zurückgibt. Zur Vereinfachung wäre eine Dateinamenskonvention für Datagroups sinnvoll.

6.2.2 Performanz

Intelligentes Zeichnen Anstatt bei jeder Neuberechnung alle Punkte zu verwerfen und komplett neu zu berechnen, wäre es in vielen Fällen vorteilhaft, nur invalidierte Datenpunkte zu verwerfen und nur die neu hinzukommenden zu berechnen. Es wären Möglichkeiten für eine solches Entscheidungsverfahren zu erarbeiten und zu überprüfen,

ob der durch einen Ansatz eingeführte Overhead in Berechnungszeit und/oder Datenstruktur ausreichend gering ist.

Canvas Ein anderer Ansatz, die Zeichengeschwindigkeit zu erhöhen ist das `<canvas>` Element. SVG hat den Vorteil, dass die dargestellten Elemente über das DOM angesprochen und manipuliert werden können. Bei vielen Tausend Elementen wird dieser Vorteil jedoch durch den großen Speicherbedarf zum Nachteil. Im Gegensatz dazu erzielt Canvas als Rastergrafik gerade bei der Darstellung von sehr großen Punktmengen viel bessere Leistungen. Eine Umstellung auf Canvas würde eine Reihe von Änderungen in der Implementierung erforderlich machen. Dies ist jedoch der erfolgversprechendste Ansatz, um auch bei der Darstellung noch größerer Punktmengen tolerierbare Reaktionszeiten zu erhalten.

Optimierung der Aggregation In der momentanen Implementierung werden die Daten zuerst auf die Indizes des Aggregationsgrids gemappt und dann aus den Indizes wieder die Koordinaten berechnet. Der Hintergrund dieser Vorgehensweise ist ein modulares Anwendungsdesign, in dem die Darstellungskomponenten leicht ausgetauscht werden können. Es könnte eine Iteration über die aggregierten Daten eingespart werden, wenn man die Daten im ersten Schritt direkt auf die von der Anzeigekomponente verwendeten Koordinaten mappt.

Parallelisierung Ein Web Worker ist ein Javascript Script, das in einem eigenen Thread im Hintergrund läuft. Der Worker hat einen anderen Speicherbereich als das Script, von dem er gestartet wird. Eine Kommunikation ist nur über den Austausch von Nachrichten möglich. Es wäre vorstellbar, die Datenaggregation in einen Web Worker auszulagern. Das würde die Berechnung nicht beschleunigen, hätte aber den Vorteil, dass das Interface während der Berechnung reaktiv bleibt. Jedoch müsste auch untersucht werden, mit welchen Performanzeinbußen durch die Thread-Kommunikation zu rechnen ist.

6.3 Zusammenfassung

In dieser Arbeit konnte gezeigt werden, dass es möglich ist, in einer Webanwendung aus mehreren Hunderttausend dreidimensionalen Datenpunkten dynamisch eine informative Visualisierung zu berechnen. Neben der Bestätigung, dass Javascript sich zunehmend als ernst zu nehmende Sprache für derart rechenintensive Anwendungsfälle emanzipiert, wurde eine weitere interessante Anwendungsmöglichkeit der Wikidata Daten aufgezeigt.

Der Name der Anwendung ist ein Wortspiel aus „visualization“ und „Wikidata“.
ViziData wird gesprochen wie „we see data“ und will damit auch zum Ausdruck bringen, dass wir uns in einer Welt allgegenwärtiger Daten befinden. Wenn wir diese Daten in einer globalen Kollaboration erfassen und katalogisieren möchten, ist es auch notwendig, Konzepte für eine menschengerechte Nutzbarmachung zu entwickeln. Im Sinne dieser fortlaufenden Zusammenarbeit haben wir für zukünftige Weiterentwicklungen den Quellcode unserer Anwendung unter der MIT Lizenz zur freien Verwendung auf GitHub veröffentlicht.

Quellen

[1] Vladimir Agafonkin. *Leaflet*. URL: http://leafletjs.com/ (besucht am 28. 08. 2014).

[2] Alexa. *wikipedia.org Site Overview*. 2014. URL: http://www.alexa.com/siteinfo/wikipedia.org (besucht am 19. 08. 2014).

[3] Martin Angelov. *Generating Files with JavaScript — Tutorialzine*. 2011. URL: http://tutorialzine.com/2011/05/generating-files-javascript-php/ (besucht am 28. 08. 2014).

[4] Highsoft AS. *Highcharts JS*. URL: http://www.highcharts.com/ (besucht am 24. 08. 2014).

[5] Highsoft AS. *Issue #3294 highslide-software/highcharts.com GitHub*. 2014. URL: https://github.com/highslide-software/highcharts.com/issues/3294 (besucht am 27. 08. 2014).

[6] Joachim Böhringer, Peter Bühler und Patrick Schlaich. *Kompendium der Mediengestaltung für Digital- und Printmedien*. 4., vollst. überarb. und erw. Aufl. Springer, 2008.

[7] Mike Bostock. *D3*. URL: http://d3js.org/ (besucht am 24. 08. 2014).

[8] Howard Butler, Martin Daly, Allan Doyle, Sean Gillies, Tim Schaub und Christopher Schmidt. *GeoJSON*. 2008. URL: http://geojson.org/ (besucht am 24. 08. 2014).

[9] *CartoDB*. URL: http://cartodb.com/ (besucht am 24. 08. 2014).

[10] Oracle Corporation. *VirtualBox*. 2007. URL: https://www.virtualbox.org/ (besucht am 28. 08. 2014).

[11] Douglas Crockford. *JSON in Java [package org.json]*. URL: https://github.com/douglascrockford/JSON-java (besucht am 03. 09. 2014).

[12] Discogs. *Data Dumps*. 2014. URL: http://www.discogs.com/data (besucht am 04. 03. 2014).

[13] Fredo Erxleben, Michael Günther, Markus Krötzsch, Julian Mendez und Denny Vrandečić. „Introducing Wikidata to the Linked Data Web“. In: *Proceedings of the 13th International Semantic Web Conference (ISWC'14)*. LNCS. to appear. Springer, 2014.

[14] Internet Engineering Task Force. *RFC 7159 - The JavaScript Object Notation (JSON) Data Interchange Format*. 2014. URL: https://tools.ietf.org/html/rfc7159 (besucht am 03. 09. 2014).

[15] *Google Fusion Table Heatmaps*. URL: https://developers.google.com/maps/documentation/javascript/fusiontableslayer#fusion_table_heatmaps (besucht am 24. 08. 2014).

[16] Antonin Guttman. „R-trees: A Dynamic Index Structure for Spatial Searching“. In: *Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data*. SIGMOD '84. Boston, Massachusetts: ACM, 1984, S. 47–57.

[17] *heatmap.js*. URL: https://github.com/pa7/heatmap.js (besucht am 24. 08. 2014).

[18] Pascal Hitzler, Markus Krötzsch, Sebastian Rudolph und York Sure. *Semantic Web : Grundlagen*. 1. Aufl. Springer, 2008.

[19] Markus Krötzsch. *How to use Wikidata*. Presentation at Wikimania. Video. 2014. URL: http://new.livestream.com/wikimania/friday2014/videos/59350537 (besucht am 24. 08. 2014).

[20] Markus Krötzsch. *How to use Wikidata*. Presentation at Wikimania. Presentation Slides. 2014. URL: http://korrekt.org/talks/2014/wikimania-wikidata.svg#76 (besucht am 20. 08. 2014).

[21] Markus Krötzsch. *Wikidata Toolkit*. 2014. URL: https://www.mediawiki.org/wiki/Wikidata_Toolkit.

[22] Magnus Manske. *Autolist*. URL: http://tools.wmflabs.org/wikidata-todo/autolist.html (besucht am 23. 08. 2014).

[23] Magnus Manske. *Wikidata tempo-spacial display*. URL: http://tools.wmflabs.org/wikidata-todo/tempo_spatial_display.html (besucht am 24. 08. 2014).

[24] Sebastian Meier. *Deutsche Nationalbibliothek - Data Explorer*. URL: http://www.sebastianmeier.eu/2014/06/21/deutsche-national-bibliothek-data-explorer/ (besucht am 24. 08. 2014).

[25] Ricardo Cabello Miguel. *three.js*. URL: http://threejs.org/ (besucht am 03. 09. 2014).

[26] John Resig. *jQuery*. 2006. URL: http://jquery.com/ (besucht am 08. 28. 2014).

[27] Murray Scott. *Interactive data visualization for the web*. first. O'Reilly, März 2013.

[28] *Statistics - Wikidata*. 2014. URL: http://www.wikidata.org/wiki/Special:Statistics (besucht am 23. 08. 2014).

[29] StatCounter Global Stats. *Top 10 Desktop, Tablet & Console Screen Resolutions from June 2013 to May 2014*. 2014. URL: http://gs.statcounter.com/#resolution-ww-monthly-201306-201405-bar (besucht am 11. 06. 2014).

[30] Inc Vizzuality. *Torque*. URL: https://github.com/CartoDB/torque (besucht am 24. 08. 2014).

[31] Denny Vrandečić und Markus Krötzsch. „Wikidata: A Free Collaborative Knowledge Base“. In: *Commun. ACM* (2014). to appear.

[32] *Wikidata:Database reports/Popular items*. 2014. URL: https://www.wikidata.org/wiki/Wikidata:Database_reports/Popular_items (besucht am 24. 08. 2014).

[33] Wikimedia. *Wikidata/FAQ*. 2014. URL: https://meta.wikimedia.org/wiki/Wikidata/FAQ (besucht am 19. 08. 2014).

[34] Wikipedia. *List of Wikipedias*. 2014. URL: https://meta.wikimedia.org/wiki/List_of_Wikipedias#Grand_Total (besucht am 19. 08. 2014).

[35] *Wikipedia Map*. URL: http://tools.wmflabs.org/render/toolkit/WikiMap/ (besucht am 24. 08. 2014).

[36] Stephen Wolfram. *Wolfram—Alpha Clip 'n Share*. URL: http://www.wolframalpha.com/share/clip?f=d41d8cd98f00b204e9800998ecf8427errmukuk5f9 (besucht am 26. 08. 2014).

[37] Stephen Wolfram. *Wolfram—Alpha Earth average radius*. URL: http://www.wolframalpha.com/input/?i=Earth+average+radius (besucht am 26. 08. 2014).

Selbstständigkeitserklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Die aus fremden Quellen wörtlich oder sinngemäß übernommenen Gedanken sind als solche kenntlich gemacht. Ich erkläre ferner, dass ich die vorliegende Arbeit an keiner anderen Stelle als Prüfungsarbeit eingereicht habe oder einreichen werde.

_____