Institut für Visualisierung und Interaktive Systeme

Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Bachelorarbeit

Dynamisch-inkrementelle Visualisierung von Personen und deren Kontext aus Nachrichtendaten

Simon Kaier

Studiengang:	Softwaretechnik
Prüfer/in:	Prof. Dr. Thomas Ertl
Betreuer/in:	Johannes Knittel
Beginn am:	26. September 2018
Beendet am:	26. März 2019

Kurzfassung

Der Zugriff auf Nachrichtenquellen aus aller Welt ist dank der globalen Vernetzung durch das Internet einfacher und schneller denn je. Zusammen mit der Verfügbarkeit steigt auch die Menge der Nachrichten, sodass es schon längst nicht mehr möglich ist, auch nur einzelne Kategorien von Nachrichten vollständig im Auge zu behalten. Um dennoch den Überblick zu behalten, verlässt man sich entweder auf Nachrichtenredaktionen, welche Zusammenfassungen vom Weltgeschehen liefern, oder man setzt auf automatisierte Aggregation der Neuigkeiten.

Ein verbreiteter Ansatz dazu ist die sogenannte Topic- oder Eventerkennung, in welcher Nachrichten nach Ereignissen geclustert und visuell dargestellt werden. Anstatt nach Ereignissen zu clustern, werden in der vorliegenden Arbeit zur visuellen Durchsuchbarkeit großer, inhomogener Nachrichtendatensätze Personen erkannt, welche als Ankerpunkt für eine grafische Darstellung dienen. Der Fokus liegt dabei darauf, die zeitliche Entwicklung der Medienpräsenz von Einzelpersonen auf einem Zeitstrahl darzustellen. Zu den Personen wird Kontext in einer Detailansicht geliefert, und zur Nachvollziehbarkeit von Ereignissen kann der Inhalt zugrundeliegender Nachrichtenartikel angezeigt und gelesen werden.

Diese Abschlussarbeit hält den Entwicklungsprozess der entstandenen Anwendung fest und beschreibt die dabei erlangten Erkenntnisse. Technische Schwierigkeiten, die während der Durchführung auftauchten, sowie Strategien zu deren Lösung werden diskutiert. Abschließend wird auf die Vor- und Nachteile des hier gewählten Ansatzes gegenüber ereignisorientierter Visualisierungen von Nachrichtendatensätzen eingegangen.

Essentiell festzuhalten ist, dass der personenorientierte Ansatz zur Nachrichtenvisualisierung ein kleineres Anwendungsgebiet abdeckt, dafür aber Vorteile in der einfacheren technischen Umsetzung bietet und weniger Rechenleistung benötigt, da auf aufwendiges Clustering verzichtet werden kann. Weiter sei zu erwähnen, dass die vorgestellte Art der Nachrichtenvisualisierung einen guten Überblick über die am häufigsten auftauchenden Personen und die diesbezügliche zeitliche Entwicklung liefert. Auch Ereignisse können mit dieser Anwendung erkannt werden, jedoch erfolgt dies weniger transparent und zuverlässig als in bekannten Projekten mit einem expliziten Fokus darauf. Neue Nachrichtenartikel können ohne Verzögerung inkrementell in die Visualisierung einbezogen und die Darstellung dynamisch erweitert werden, was den Einsatz als Echtzeit-Monitoring-Tool für Nachrichtenquellen ermöglicht.

Abstract

Accessing news sources from around the world is easier and faster than ever thanks to the Internet. As availability increases, the amount of news also does meaning it is no longer viable to keep track of individual categories of news. To follow the world affairs, one either relies on newsrooms that provide summaries of current events, or on automated news aggregation.

A common approach to this is the so-called topic or event detection, in which news are clustered by events and displayed visually. Rather than building clusters of events, the present work on the visual searchability of large, inhomogeneous message datasets identifies persons who serve as anchor points for a graphical representation. The focus here is on representing the temporal evolution of the media presence of individuals on a timeline. Context is delivered to the persons in a detail view, and the content of underlying news articles can be viewed and read for the traceability of events.

This thesis documents the development process of the resulting application and describes the findings gained. Technical difficulties that arose during the implementation and strategies for their solution are discussed. Finally, the advantages and disadvantages of the chosen approach towards event-oriented visualization of message data sets are discussed.

It should be noted that the person-oriented approach to message visualization covers a smaller area of application, but offers advantages in the simpler implementation and less computational power needed, as can be dispensed with complex clustering. It should also be mentioned that the presented type of message visualization provides a good overview of the most frequently occurring persons and the temporal development in this respect. Events can also be detected with this application, but this is less transparent and reliable than in known projects with an explicit focus event recognition. New news articles can be incrementally included in the visualization without delay, and the presentation expanded dynamically, allowing it to be used as a real-time news source monitoring tool.

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

Motivation

Nachrichtensammlungen wie Nachrichten-Websites oder Weblogs sind eine wichtige Informationsquelle für beispielsweise die Entscheidungsfindung in Unternehmen, Studien im Umfeld Sozial- und Kulturwissenschaften und politische Entscheidungen. Sie spiegeln Aspekte des Weltgeschehens wieder, wobei der Zeitpunkt ihrer Veröffentlichung essenziell für die korrekte Einordnung der Zusammenhänge ist. Eine große Herausforderung in der Auswertung dieser Informationsquellen stellt der hohe Umfang der täglich veröffentlichten Neuigkeiten, deren Inhomogenität bezüglich Länge, politischer Motivation, Informationsgehalt und der auf den Leser bezogenen Relevanz dar. Eine automatische Auswertung von Textsammlungen bietet hierbei Möglichkeiten zur umfangreicheren Abdeckung der Nachrichtenüberwachung. Eine vollständig algorithmische Aufbereitung und Zusammenfassung von unstrukturiertem Text stellt eine technisch schwierige Aufgabe dar und die rein textuelle Darstellung der Ergebnisse wären nicht optimal für das menschliche Verständnis der Inhalte. Eine wesentlich effizientere perzeptuelle Aufnahme von präsentierten Informationen ist bei einer interaktiven Visualisierung von den zugrundeliegenden Daten möglich. Dabei wird außerdem eine geringere Aggregation und Analyse in der Vorverarbeitung benötigt. Hierzu wird ausgenutzt, dass der Mensch sehr effizient Informationen aus Grafiken aufnehmen und Muster erkennen kann.

In dieser Arbeit wird die Idee verfolgt, Personen des öffentlichen Lebens in den Nachrichtendaten zu erkennen und diese als Ausgangspunkt für weitere Analysen zu verwenden. Hauptsächlich soll dabei dargestellt werden, welche Personen zu welchen Zeitpunkten wie häufig erwähnt werden, um auf Trends und brisante Ereignisse aufmerksam zu machen.

Angelehnt an die Prinzipien des Visual Analytics Mantra [Kei+06] lässt sich folgender roter Faden formulieren, der die Grundanforderungen an eine Anwendung für eine effektive visuelle Analyse von großen Nachrichtensammlungen darstellt:

- Stelle einen maschinellen Vorverarbeitungs-/Analysemechanismus für den gesamten Nachrichtendatensatz bereit, um die Bewältigung umfangreicher Datenmengen zu erleichtern.
- Gib einen visuellen Überblick über alle wichtigen Ereignisse und deren zeitliche Einordnung.

- Ermögliche einen zeitlichen zoom, filtere uninteressante Informationen und ermögliche eine interaktive Analyse der für den Betrachter interessanten Dokumente.

Unter Berücksichtigung der oben aufgezählten Ansprüche wurde im Rahmen dieser Arbeit eine Anwendung entworfen, die Erwähnungen von Personen aus Nachrichtendaten erkennt und diese zur Übersicht auf einem Zeitstrahl einordnet. Die Vorverarbeitung der Nachrichtendaten wird auf der Server-Komponente der Anwendung ausgeführt. Eine Übersichtsseite bietet den Überblick über die erkannten Personen, von wo aus Kontextinformationen angefragt werden können. Dazu wurde eine Detailansicht implementiert, die zu einer Person für einen frei wählbaren Zeitraum Schlagwörter, eine Übersicht über in Relation stehende weitere Personen und Nachrichtenartikel auflistet, welche in diesem Zeitraum veröffentlicht wurden und jene Person erwähnen. Interessiert sich der Anwender für die Nachrichtenartikel im einzelnen, kann er dort ihren Inhalt aufrufen. Die Übersichtsseite bietet die Möglichkeit, über die Zeit zu zoomen. Uninteressante Informationen werden dadurch gefiltert, dass selten erwähnte Personen ab einem gewissen Schwellenwert ignoriert und die am häufigsten auftretenden Personen ganz oben angeordnet werden.

Eine Schwierigkeit, die mit diesem Ansatz einhergeht ist unter anderem, dass im Durchschnitt zu viele Personen in den Nachrichten erwähnt werden, um sie alle gleichzeitig anzeigen zu können. Dazu kommt, dass Personen mit geringer Medienpräsenz oft nur sporadisch oder zu bestimmen Events auftauchen, was für große freie Flächen sorgt, wenn alle Personen in der Visualisierung gleich viel Platz zugewiesen bekommen. Um dieses Problem zu adressieren, wurde ein Mechanismus für das Layout entworfen, der möglichst viele Personen gemeinsam anzeigt.

Gliederung

Der folgende Teil der Arbeit ist wie folgt strukturiert:

Kapitel 2 - Verwandte Arbeiten: Hier werden verwandte Arbeiten im Bereich ereignisorientierter Nachrichtenvisualisierung aufgeführt.

Kapitel 3 - Visualisierung: In diesem Kapitel wird der konzeptuelle Ansatz der implementierten Visualisierung beschrieben. Interaktionen mit der Oberfläche und Umsetzungen der Informationsdarstellung werden aufgeführt.

Kapitel 4 - Datenverarbeitung: Der Abschnitt zur Datenverarbeitung stellt verwendete Frameworks und Technologien zur Vorverarbeitung der Nachrichtendaten im Server auf konzeptueller Ebene vor.

Kapitel 5 - Implementierung: Dieses Kapitel zeigt die Anwendung aus technischer Sicht. Die Wahl der verwendeten Technologien, die Architektur von Server und Client, sowie Details zu entstandenem Code finden sich dort.

Kapitel 6 - Evaluierung: In diesem Abschnitt wird eine durchgeführte Expertenbefragung vorgestellt und die Ergebnisse präsentiert.

Kapitel 7 - Zusammenfassung und Ausblick: Zuletzt wird eine Zusammenfassung der Arbeit gegeben und mögliche weitere Entwicklungen auf diesem Gebiet vorgestellt.

2 Verwandte Arbeiten

Dieser Abschnitt befasst sich mit verwandten Arbeiten zum Thema "Dynamische Visualisierung von Ereignissen aus Nachrichtendaten". Zum Zeitpunkt der Entstehung der Arbeit listet die Computer Science Bibliography dblp[1] keine Dokumente zur Anfrage "visualization of people in news", weshalb dieses Kapitel sich mit Arbeiten zum Thema Text- und Ereignis-Visualisierung von Nachrichtendaten unter Betrachtung zeitlicher Zusammenhänge beschäftigt. Auch auf Google Scholar[2] finden sich nach Kenntnisstand des Autors keine Arbeiten, welche die Visualisierung der Erwähnungen von Personen in Nachrichtenartikeln umsetzen. Eine verwandte Kategorie in der Visualisierung ist das Verfolgen von Schlüsselwörtern und die Änderung derer Bedeutung über die Zeit bezüglich einer Dokumentensammlung. Ein repräsentativer Ansatz in dieser Kategorie ist ThemeRiver[Hav+02], der die Stärkeänderungen einzelner Schlüsselwörter als Strömungen innerhalb eines Flusses entlang einer Zeitachse darstellt. Ein allgemeinerer Ansatz zur Visualisierung großer, hochdimensionaler Datensätze ist in Circle Segments [AKK96] umgesetzt, welcher Zeitserien mittels Kreissegmenten darstellt. Im Bereich Erkennen, Clustering und Visualisierung von Ereignissen (Events) gibt es verschiedene wissenschaftliche Arbeiten und Visualisierungssysteme, von denen im Folgenden drei zentrale Vertreter vorgestellt werden.

2.1 CloudLines

CloudLines (Abbildung 2.1) ist eine Visualisierung von Ereignisserien, auf welche Nachrichtenartikel hindeuten. Dabei greift CloudLines auf Datensätze von EMM news service[3] zurück. Ereignisse innerhalb dieser Datensätze werden algorithmisch identifiziert, während Serien von Events visuell erforscht werden können. Nutzer können damit verkettete Ereignisse und deren Verläufe visuell-interaktiv nachvollziehen. Nachrichtenartikel werden so wenig wie möglich aggregiert, um Details innerhalb der Nachrichten zu erhalten und einen Zoom der Zeitachse (horizontal) in der Event-Serie zu erlauben. Damit lassen sich interessante Zeitabschnitte genauer betrachten und Zusammenhänge verdeutlichen. Die Zeitachse ist dabei auf Wunsch entweder linear oder logarithmisch, wobei die logarithmische Skala aktuellen Ereignissen mehr Platz in der Darstellung zur Verfügung stellt, als Ereignissen aus der Vergangenheit. Eine Zeile

[1] https://dblp.org/
[2] https://scholar.google.de/scholar
[3] http://emm.newsbrief.eu

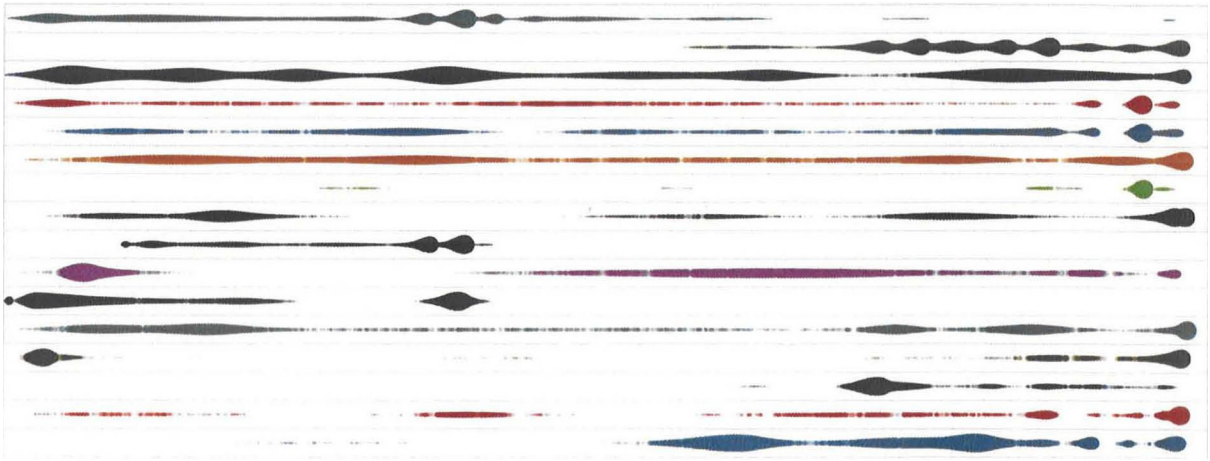

Abbildung 2.1: Quelle: CloudLines [MKK11]

entspricht dabei einem Event, ist in eine von elf Farben gefärbt (um sie besser von naheliegenden Zeilen unterscheiden zu können) und hat eine feste Höhe. Die Relevanz des Ereignisses, welche aus der Medienpräsenz abgeleitet wird, wird über die Dicke der Linie innerhalb der Zeile dargestellt. Sie ist auf die Höhe der Zeile normiert, was zum einen eine bessere Lesbarkeit und zum anderen eine kompakte Darstellung ermöglicht, jedoch keine Vergleichbarkeit in der Brisanz gegenüber anderer Events bietet. Eine dynamische Einbindung neuer Datensätze ist in CloudLines nicht möglich, da nicht alle Algorithmen der Datenvorverarbeitung inkrementell arbeiten. Die Erweiterung der Daten erfordert daher eine komplette Verarbeitung einschließlich der alten Daten, was eine Nutzung in Verbindung mit einem kontinuierlichen Nachrichtenstrom ausschließt. [MKK11]

2.2 EventRiver

EventRiver (Abbildung 2.2) ist genau wie CloudLines eine Visualisierung von Ereignisserien über eine Zeitachse. Die Ziele sind dabei, einen visuellen Überblick über aktuelle Ereignisse zu geben, diese durchsuchbar zu machen, sie in Verbindung miteinander zu setzen und Nachrichtenartikel im Detail und in Gänze aufrufen zu können. Außerdem sollen Ereignisketten nachverfolgbar sein. EventRiver verspricht, damit auch komplexe Ereignisserien – sogenannte Stories – erfassen zu können, um deren zeitliche Entwicklung und Zusammenhänge nachvollziehbar zu machen. In der Datenvorverarbeitung werden dafür Nachrichten nach Inhalt und Zeit geclustert. Die Zeitkomponente erlaubt im Clustering eine Unterteilung der erkannten Ereignisse in verschiedene Entwicklungsstufen. Zusammengehörende Ereignisse einer Story werden visuell gruppiert angezeigt, wofür diese verschiedenen Gruppen zugeordnet werden. Die so entstehenden Ereignisgruppen werden abhängig vom Zoom-Level jeweils zusammen mit Kontextwörtern angezeigt, welche markant für das jeweilige Cluster sind. Anders als bei CloudLines werden Daten inkrementell verarbeitet, um eine Echtzeitanwendung und hohe

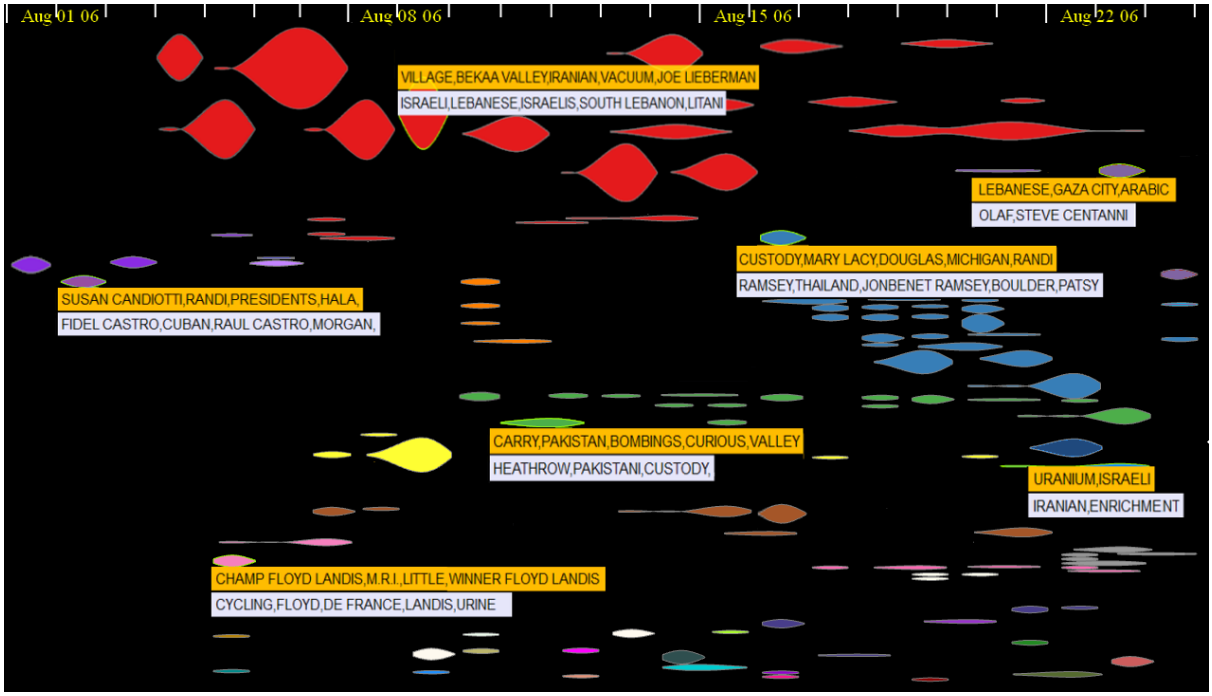

Abbildung 2.2: Quelle EventRiver [DL10]

Effizienz zu ermöglichen. Eine Schwäche von EventRiver ist verglichen mit beispielsweise TextFlow der weniger ansprechende ästhetische Aspekt der Visualisierung. Das äußert sich etwa in überladenen Overlay-Menüs im Windows XP Stil und inakkuraten kubischen Spline-Interpolationen der Event-Blasen, welche ins Negative schwingen (siehe Abbildung 2.2, zweite rote Blase von oben, links oben).

Die Datenverarbeitung erfolgt zweistufig: Im ersten Schritt werden die Rohdaten über einen festen Zeitraum (z.B. 24 Stunden) gesammelt und in detaillierten, statistischen Zusammenfassungen zu Stapeln (Batches) gebündelt. Im zweiten Schritt werden die Batches jeweils unabhängig von sämtlichen anderen Daten analysiert und geclustert, um sie anschließend mit den bereits bestehenden Datensätzen zusammenzuführen. [DL10]

2.3 TextFlow

TextFlow ist ebenfalls eine Visualisierung von Ereignisserien in Nachrichtenartikeln, jedoch ist sie sowohl in der Datenvorverarbeitung, als auch in der Darstellung komplexer als EventRiver und CloudLines. Der Kern der Darstellung besteht aus drei Komponenten: Glyphen/kritische Ereignisse, Schlüsselwortstränge und Themenflüsse. Feine Linien in der Darstellung sind die Schlüsselwortstränge, deren Schlüsselwörter mit der gleichen Farbe eingezeichnet sind. Die räumliche Distanz zwischen Schlüsselwortsträngen bildet die Korrelation zwischen Schlüsselwörtern ab. Je mehr gemeinsame Überschneidungen sie haben, desto näher sind sie zusammen.

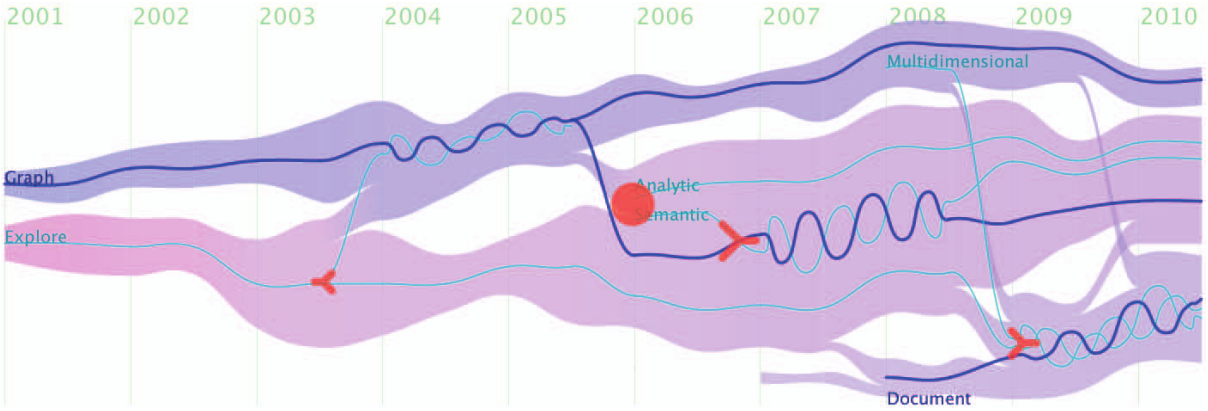

Abbildung 2.3: Quelle: TextFlow [WC11]

Kritische Ereignisse werden als Glyphen visualisiert und stellen Beginn, Ende, Verzweigung oder Zusammenführung von Schlüsselwortsträngen dar. Die großen farbigen Stränge im Hintergrund sind Themenflüsse, welche für jeweils ein Cluster von Events stehen. Ein weiteres Element ist das Wellenbündel, welches in der Abbildung 2.3 drei mal zu sehen ist. Es stellt die Zusammenführung (mehrerer) Schlüsselwortstränge dar, welche zuvor nicht im selben Kontext aufgetaucht sind, in diesem Zeitabschnitt aber eine erhöhte Korrelation aufweisen. Die Amplitude der Wellenbündel steht für die Anzahl der Schlüsselwörter, die dabei korrelieren. Auch TextFlow setzt auf Clustering der Nachrichtenartikel, um Themen und Ereignisse zu erkennen. [WC11]

3 Visualisierung

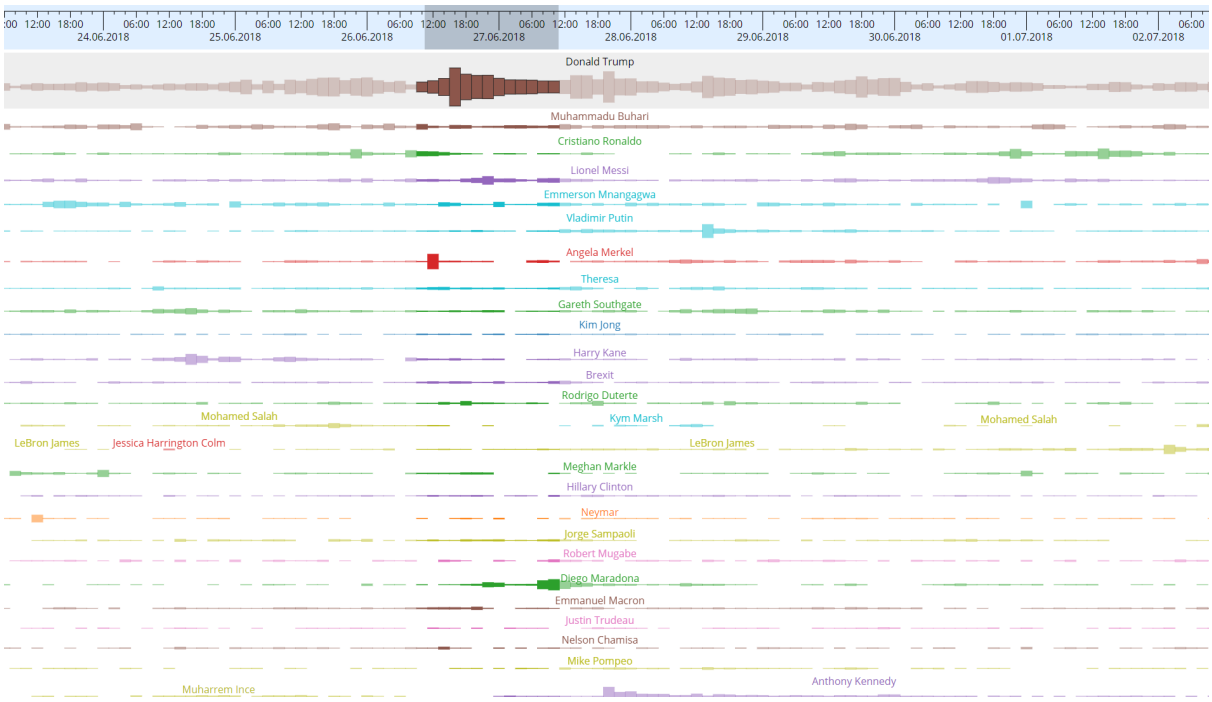

Abbildung 3.1: Screenshot der Übersichtsseite

In diesem Kapitel wird die Visualisierung vorgestellt. Nach dem generellen Visualisierungsansatz werden Designentscheidungen und Details diskutiert.

Die Anwendung wird im Web-Browser ausgeführt und besteht aus zwei grundlegenden Ansichten: Einer Übersicht (siehe Abbildung 3.1) über Personen in einem Zeitraum und einer Detailseite (siehe Abbildung 3.2), welche sich für weitere Informationen über die Übersicht legt.

Die Übersichtsseite gibt einen Überblick über die Personen, die im aktiven Zeitraum am häufigsten in den zugrundeliegenden Nachrichtendaten vertreten sind. Als Zeitachse dient die Horizontale, über welche die Häufigkeitsverteilung der Personen aufgespannt ist. Dabei ist die Anzahl der Erwähnungen einer Person innerhalb eines Zeitfensters durch einen Balken fester Breite dargestellt. Für die Spanne der einzelnen Zeitfenster wurde hier zwei Stunden und somit 12 Zeitfenster pro Tag gewählt. Die Höhe eines Balkens ist linear zur Anzahl der

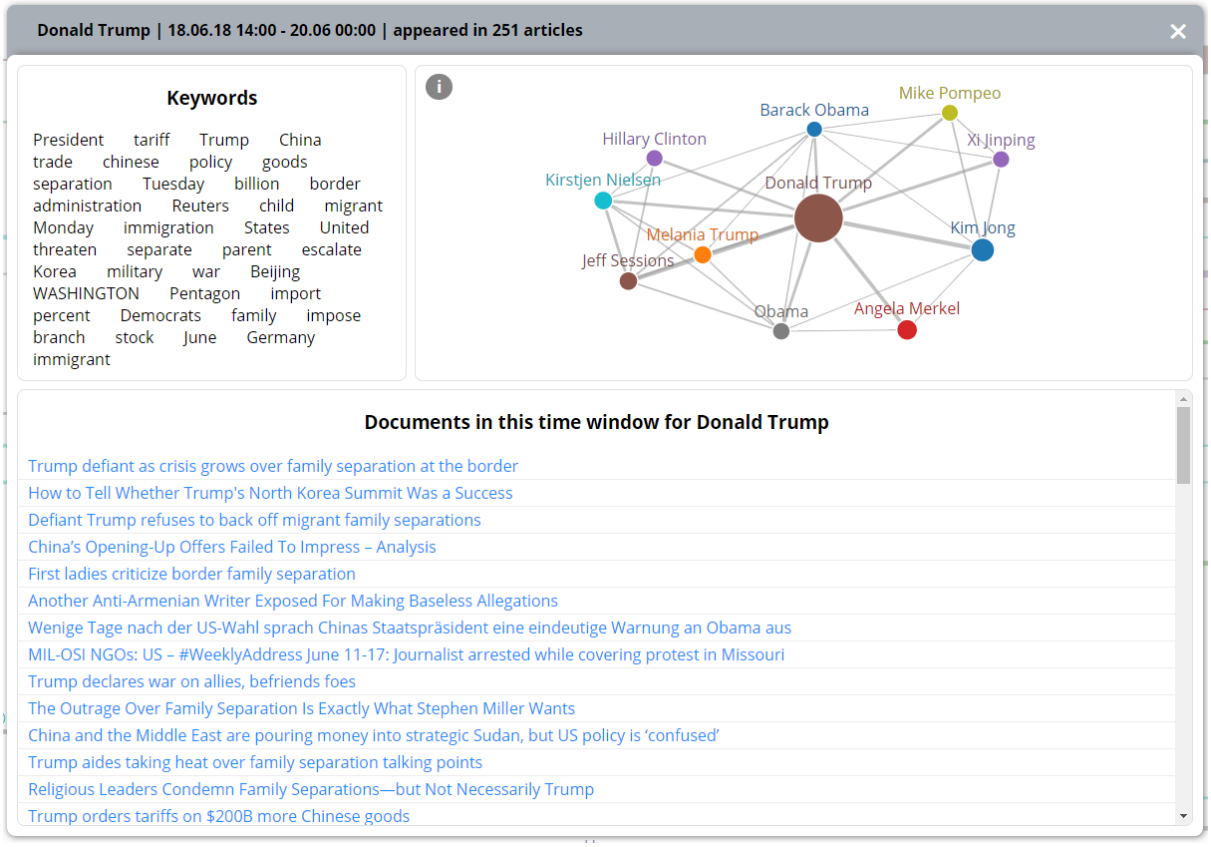

Abbildung 3.2: Screenshot der Detailseite

Erwähnungen in diesem Zeitraum und alle Erwähnungsbalken einer Person befinden sich auf einer gemeinsamen horizontalen Spur.

Die Detailseite liefert Informationen zum Kontext einer Person. Sie bezieht sich jeweils auf den Zeitraum, der in der Übersichtsseite ausgewählt wurde. Hier werden Schlüsselwörter gelistet, die auf Ereignisse hindeuten, die mit der Person in Verbindung stehen. Um genaueren Kontext nachzuvollziehen, können von hier aus Dokumente, in denen die Person erwähnt wurden, in Volltext aufgerufen werden. Als weiteres Element werden weitere Personen, welche häufig gemeinsam mit der primär betrachteten Person in Dokumenten auftauchen, als Relationenraph dargestellt.

3.1 Darstellung der Personen

Die Anforderungen an die Darstellung der Personen geben Aufschluss über einige der im Folgenden erwähnten Designentscheidungen. Da der Platz auf dem Monitor sehr beschränkt ist, ist es eine Kernanforderung, die Darstellung so kompakt wie möglich zu halten, wie in Abschnitt 3.2 genauer beschrieben wird. Um detaillierte Inspektionen vornehmen zu können,

muss der betrachtete Zeitraum interaktiv änderbar sein (siehe auch Abschnitt 3.4). Weitere Anforderungen sind, dass die Darstellung so genau wie möglich die zugrundeliegenden Daten widerspiegelt und zudem die Aggregation gering zu halten, um keine Details zu verlieren.

Die Übersichtsseite zeigt einen Zeitraum von initial einer Woche, mit aufsteigendem Datum von links nach rechts. Dazu wird der betrachtete Zeitraum in *Zeitfenster* von einigen Stunden unterteilt. Für jedes dieser Zeitfenster wird festgehalten, in wie vielen Nachrichtenartikeln, die in diesem Zeitfenster veröffentlicht wurden, eine Person vorkommt. Ein Zeitfenster wird als Rechteck (im Folgenden auch als *Erwähnungsbalken* bezeichnet) dargestellt, dessen Breite der abgedeckten Zeitspanne entspricht, während die Höhe linear zur Anzahl der Erwähnungen verläuft. Auf eine Weichzeichnung der Darstellung, wie man sie etwa in CloudLines oder EventRiver sieht, wird hier bewusst verzichtet, um die Visualisierung so nah wie möglich an den tatsächlichen Daten zu halten. Alle Erwähnungsbalken einer Person befinden sich auf einer gemeinsamen horizontalen *Spur*. Personen werden nach ihrer Medienpräsenz sortiert und von oben nach unten über den Bildschirm angeordnet. Relevante Personen stehen also weit oben, während selten erwähnte Menschen weiter unten, erst nach Verschieben des Sichtfeldes nach unten, oder ab einem festlegten Grenzwert gar nicht mehr angezeigt werden. Jeder Person wird eine von zehn Farben fest zugeordnet, in der ihr Namenslabel und ihre Erwähnungsbalken gefärbt sind.

3.2 Platzoptimierung

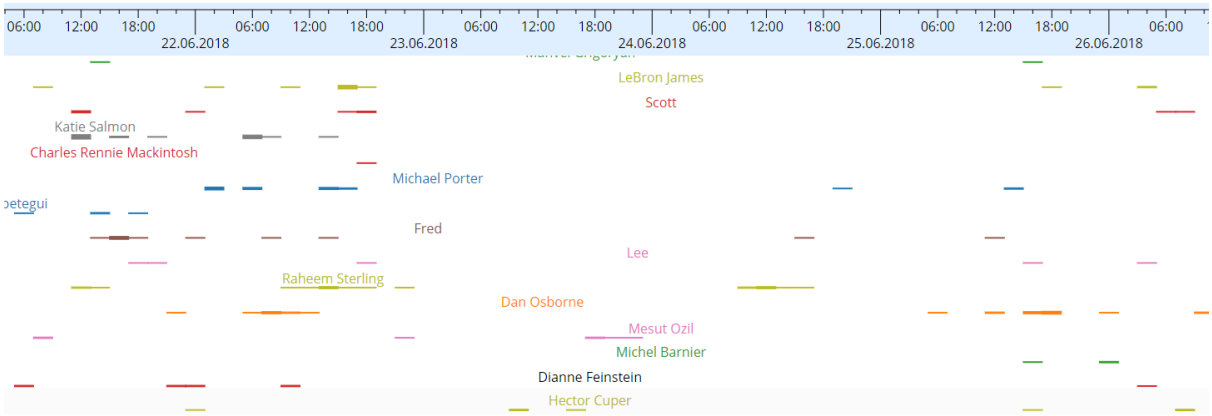

Abbildung 3.3: Screenshot der ausdünnenden Übersicht

Um so viele Personen wie möglich mit einem Blick erfassen zu können, wurden folgende Probleme adressiert: Während sehr prominente Personen, wie beispielsweise Donald Trump (Stand 2019) so gut wie täglich in den Medien auftauchen, gibt es viele Personen, die selten oder nur in Verbindung mit bestimmten Ereignissen erwähnt werden, wie beispielsweise Fußballer. Gibt man jeder Person eine eigene horizontale Spur, ergeben sich zwei Probleme: Die Darstellung dünnt nach unten hin aus, wodurch sich große ungenutzte Areale auf dem

Display ergeben und sie expandiert stark in der horizontalen Richtung. Screenshot 3.3 zeigt eine solche Stelle der Visualisierung mit wenigen Messpunkten.

Um diesem Problem entgegenzuwirken, werden freie Stellen einer Spur mit den Erwähnungsbalken weiterer Personen aufgefüllt. Damit dabei keine Überlappungen entstehen, werden nur solche Personen auf eine gemeinsame Spur gebracht, deren Erwähnungsbalken sich nirgends im aktiven Zeitraum überlappen. Die verschiedenen Farben der Personen helfen dabei, mehrere Personen auf derselben Spur auseinanderhalten zu können.

Screenshot 3.4 zeigt den selben Zeitraum mit der Optimierung. Hier ist zu sehen, dass “Katie Salmon” (links oben) nur vor dem 23.06. auftritt. Die optimierte Darstellung nutzt dieselbe Zeile, um zusätzlich noch “Malik Tahir Javaid” und “Gavin Williamson” anzuzeigen, da diese sich nicht überschneiden.

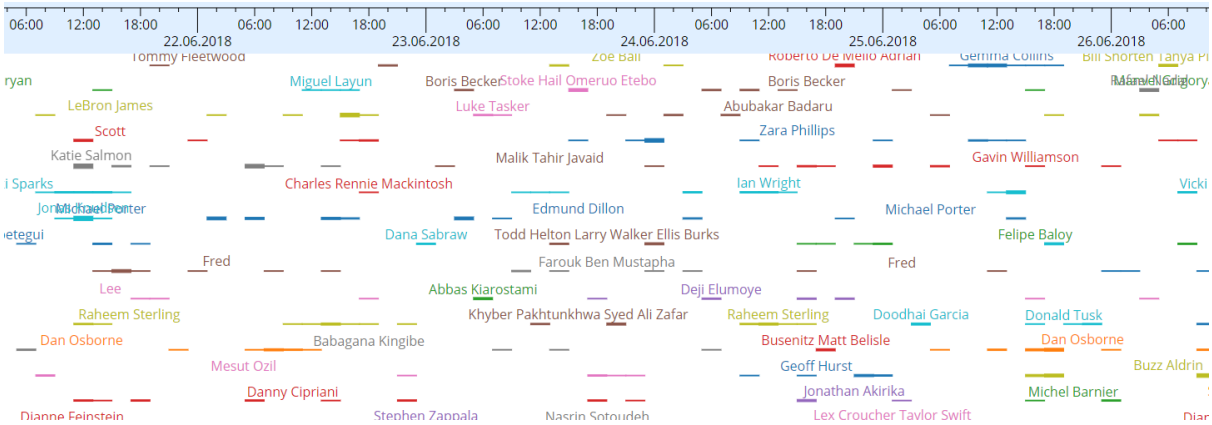

Abbildung 3.4: Screenshot der optimierten Übersicht

3.3 Inkrementeller Aspekt der Visualisierung

Der aktive Zeitraum lässt sich mittels Mausgesten variabel zoomen und verschieben, worauf in Abschnitt 3.4 genauer eingegangen wird. Wird der aktive Zeitraum durch die Nutzereingabe erweitert, sodass er einen Zeitraum abdeckt, der zuvor nicht angezeigt wurde, werden die Daten vom Server angefragt und die Anzeige um die neuen Daten erweitert. Aktionen vom Nutzer, die eine Anfrage an den Server erfordern, blockieren die Anzeige nicht, damit auf den bisherigen Daten weitergearbeitet werden kann. Mit dem Ändern der aktiven Daten ändert sich gegebenenfalls auch die Reihenfolge der Personen. Die Personen, welche sich eine gemeinsame Spur teilen, ändern sich ebenfalls, wenn sie im dazukommenden Zeitabschnitt gemeinsam auftreten. Um beim vertikalen Verschieben der Personen zu verhindern, dass der Betrachter eine beobachtete Person aus den Augen verliert und erst wieder suchen muss, erfolgt die Verschiebung durch eine ein bis zweisekündige Animation. Da Animationen und Änderungen der Darstellung in der Visualisierung dafür bekannt sind, den Betrachter potenziell

zu irritieren, vor allem dann, wenn die Änderung nicht im aktuellen Fokus des Betrachters liegt, werden diese Übergänge ausschließlich unmittelbar durch Mausgesten ausgelöst. Als weitere Maßnahme ist die Sortierung der Personen nach Relevanz nicht nur von der Anzahl der Erwähnungen im aktiven Zeitraum abhängig, sondern behält bis zu einem Grenzwert die Sortierung im Augenblick vor der Zeitraumänderung bei, um Flimmern zu minimieren.

3.4 Interaktion

In der Übersichtsseite kann der Nutzer den horizontalen Zoom durch Rollen mit dem Mausrad einstellen. Der aktive Zeitraum kann mit der Maus per Dragging stufenlos verschoben werden. In der Zeitleiste, die sich am oberen Rand der Anwendung befindet, kann man ein Zeitinterval per Dragging auswählen. Diese Zeitauswahl wird genutzt, um den Zeitraum für die Detailansicht zu bestimmen. Eine Detailansicht öffnet sich durch einen Mausklick auf die Erwähnungsbalken einer Person. Wird keine Zeitauswahl explizit getroffen, so wird für die Detailansicht der gesamte aktive Zeitraum gewählt, in dem die Person auftritt. Für die Liste der Personen auf der Übersichtsseite gibt es einen Filter, mit dem man explizit nach Namen suchen kann. Die Suchzeile dafür ist dauerhaft aktiv, um Eingaben direkt anzunehmen, wird aber nur angezeigt, solange sie nicht leer ist.

3.5 Detailansicht

Die Detailansicht bezieht sich immer auf eine Person und den gewählten Zeitraum (siehe Abschnitt 3.4). Sie (Abbildung 3.2) bietet drei Funktionen zur tieferen Detailermittlung: eine Liste mit Schlüsselwörtern, eine Graphenansicht mit korrelierenden Personen und eine Liste der Dokumente aus dem gewählten Zeitraum, in denen die Person erwähnt wird.

Die Schlüsselwörter werden über ein Ranking algorithmisch als charakterisierend für den Kontext eingestuft. Sie sollen einen schnellen Überblick geben, um welche Themen sich die Nachrichtenartikel in diesem Zeitraum drehen.

Die Graphenansicht soll einen schnellen Überblick geben, welche Personen häufig im selben Kontext wie die primäre Person auftauchen. Personen werden als Kreise dargestellt und sind auf zehn Stück limitiert, um nur die relevantesten zu listen und Übersichtlichkeit zu gewährleisten. Kanten zwischen den Kreisen reflektieren die Anzahl der gemeinsamen Erwähnungen von jeweils zwei Personen durch ihre Dicke. Der Flächeninhalt der Kreise ist linear zum Auftreten der Person im aktiven Zeitraum.

Die Liste der Dokumente enthält die Titel der Nachrichtenartikel, aus welchen die Anzahl der Erwähnungen, die Schlüsselwörter und die Korrelationen zwischen den Personen extrahiert wurden. Ein Mausklick auf einen Titel öffnet den Volltext des Artikels, um die genauen Details nachlesen zu können. Die Basisfilterkriterien für die Liste sind der Name der aktiven Person

und der gewählte Zeitraum. Als weiteres Filterkriterium lässt sich entweder eine Person aus den Relationen oder ein Schlüsselwort aus der Schlüsselwortliste anwählen, um die Such zu einzuschränken.

4 Datenverarbeitung

Diese Kapitel stellt die Frameworks und Technologien auf konzeptioneller Ebene vor, die zur Vorbereitung der zugrundeliegenden Nachrichtendaten für die Verwendung in der Visualisierung verwendet wurden.

Eine zentrale Rolle spielt die Erkennung von Personen in den Nachrichtentexten, welche mittels maschineller Sprachverarbeitung (Abschnitt 4.1) umgesetzt wird. Außerdem wird die Verwendung der Datenbank MongoDB in Abschnitt 4.2 und die dort verwendeten Datenstrukturen beschrieben.

4.1 NLP

Das maschinelle Verarbeiten von natürlicher Sprache wird im englischen als *Natural Language Processing* oder kurz NLP bezeichnet. Es umfasst unter anderem das Analysieren von Strukturen, Kategorisieren von Worten und Satzbausteinen, sowie semantische Auswertungen [Man+14]. So können beispielsweise Personen, Institutionen, Orte oder andere Entitäten erkannt werden. Im Kontext dieser Arbeit entspricht eine Entität einer realen Person, welche in den Nachrichtendaten erkannt wurde. NLP-Software wird für gewöhnlich in kompakten Anwendungen, sogenannten NLP-Tools, bereitgestellt und in Form von Bibliotheken in anderen Softwaresystemen eingebunden. Das in dieser Implementierung verwendete NLP-Tool ist das *Stanford CoreNLP Toolkit* [Man+14]. Das Grundprinzip dieser NLP-Tools ist das Annotieren von Worten (Worte hier im Sprach-theoretischen Sinn), was das Hinterlegen zusätzlicher Informationen zu einem Wort, Satz oder Satzkonstellation bezeichnet. Diese Worte werden im Folgenden auch als Tokens bezeichnet. Annotationen werden meist in Baumstrukturen organisiert und verarbeitet. Für das Erstellen von Annotationen sind sogenannte Annotatoren (engl. Annotators) zuständig, welche die Funktionen der Sprachverarbeitung implementieren. Annotators bauen aufeinander auf und besitzen hierarchische Abhängigkeiten. Um beispielsweise einen *Words To Sentence Annotator* ausführen zu können, welcher einzelne Wörter zu Sätzen gruppiert, muss der Text zuvor mittels *Tokenizer Annotator* in einzelne Wörter unterteilt werden. Ein typischer Ablauf einer Textanalyse ist also die Ausführung mehrerer Annotatoren hintereinander, was als Pipeline bezeichnet wird. NLP-Tools bieten in der Regel vorgefertigte Pipelines für gängige Nutzungsszenarien, aber um eine feingranularere Kontrolle über die Rechenzeit und das Ergebnis zu haben, können Pipelines auch individuell und programmatisch aus einzelnen Annotatoren zusammengestellt werden.

Abbildung 4.1: Minimale NLP-Pipeline

Eine minimale Pipeline für die Erkennung von Personen ist im Fall von Stanford CoreNLP in Abbildung 4.1 dargestellt und in folgender Liste erklärt.

Tokenzier: der Tokenizer unterteilt den Text in eine Liste von Tokens. Im Deutschen und im Englischen wird dafür im Wesentlichen nach Leerzeichen getrennt. Im Web-Umfeld werden Markup-Tags berücksichtigt. Die Annotation beinhaltet dann die Positionen der einzelnen Worte (Tokens).

Words To Sentence: Unterteilt eine Liste von Tokens in Sätze.

Pos-Tagger: Der Part-Of-Speach-Tagger annotiert Tokens mit der Information darüber, welchen Teil des Satzes sie bilden, also beispielsweise “Substantiv”, “Verb”, “Adjektiv” etc.

Lemmatizing: Der Lemmatizer, oder auch MorphAnnotator überführt Tokens in ihre kanonische Form, also ihren Wortstamm. Zum Beispiel würde “Die Schafe saßen..” mit {“Die”, “Schaf”, “sitzen”} vom Lemmatizer annotiert werden.

NER: Kurz für *Named Entity Recognition*. Dieser Annotator erkennt namentlich erwähnte Entitäten, wie Personen, Institutionen oder Orte. Er wird häufig mittels maschinellem Lernen umgesetzt und ist relativ rechenintensiv.

4.2 Datenbank Schema

Um die Nachrichtenartikel zu persistieren, welche die Datengrundlage für die Visualisierung darstellen, speichert die implementierte Serveranwendung diese in einer Dokumentendatenbank.

Die Architektur des Servers und wie die Datenbank angebunden ist, wird schematisch in Abbildung 5.1 im Kapitel 5: Implementierung dargestellt. Im Unterabschnitt 5.2.1 wird auf die Details zur Wahl der MongoDB als Dokumentendatenbank eingegangen.

Abbildung 4.2 zeigt einen Auszug aus dem verwendeten Datenbankschema. Aus Gründen der Übersichtlichkeit wurden einige, für die Nachvollziehbarkeit nicht notwendigen Felder ausgelassen.

Die *Raw Articles* bilden den Inhalt und die vorhandenen Metadaten der importierten Nachrichtenartikel ab. Ihre ID (_id in der Abbildung) ist automatisch generiert. Das Feld *content* enthält den Nachrichteninhalt und die *description* die Zusammenfassung der Artikel, falls vorhanden.

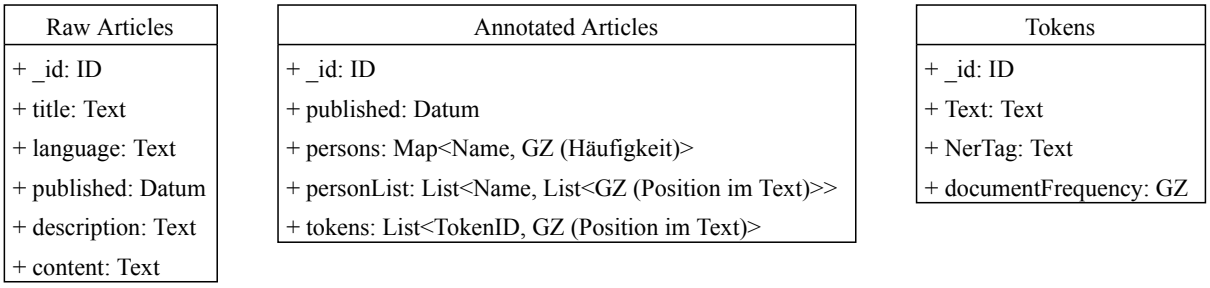

Abbildung 4.2: Vereinfachtes Datenbankschema

Die *Annotated Articles* Tabelle enthält alle Daten, die beim Importieren der Nachrichtenartikel (der Vorverarbeitung) extrahiert wurden. Details zur Extraktion finden sich im Abschnitt 5.2. Jedem *Raw Article* wird genau ein *Annotated Article* zugeordnet. Die jeweiligen Artikel-Paare teilen sich dieselbe ID, um im späteren Verlauf aufeinander referenzieren zu können. Das Feld *published* wird für die Filterung nach Zeitraum bei REST-Anfragen des Clients genutzt und besitzt deshalb einen Index für einen schnellen Zugriff. In *persons* ist abgelegt, welche Namen von Personen im Artikel vorkommen und wie oft. Ihre Map-Struktur ermöglicht eine effiziente Filterung nach Namen. *personList* enthält ebenfalls die vorkommenden Namen, jedoch als Liste und jeweils gemeinsam mit den Positionen, an denen sie im Text auftauchen. Sie kann dazu genutzt werden, nachzuvollziehen ob bestimmte Personen häufig im selben Satz genannt werden, um Relationen zwischen ihnen besser auswerten zu können. Die *tokens* Liste bildet alle Wörter aus dem *content* der ursprünglichen Artikel, welche nicht auf den im weiteren Verlauf erläuterten Stoppwortlisten stehen, auf IDs ab, welche auf die Tokens in der Dokumententabelle *Tokens* verweisen. Die Ersetzung des Textes dient zur Minimierung des erforderlichen Speicherplatzes.

Der Grund, wieso die *Annotated Articles* in einer eigenen Tabelle abgelegt sind und nicht einfach in den *Raw Articles* ergänzt werden liegt daran, wie MongoDB Daten lädt und puffert: Wenn Felder eines Dokuments aus der Datenbank abgefragt werden, so wird zunächst das gesamte Dokument in den Arbeitsspeicher geladen. Im nächsten Schritt werden alle angefragten Felder an den Aufrufer ausgeliefert. Anschließend wird das vollständige Dokument so lange im Arbeitsspeicher behalten, bis dem System der verfügbare RAM knapp wird und gepufferte Dokumente in aufsteigender Reihenfolge des Alters des letzten Zugriffs wieder aus dem RAM entfernt werden. Zugriffe auf die *Annotated Articles* Tabelle erfolgen also im Schnitt wesentlich schneller als Zugriffe auf *Raw Articles*, da sie auch bei größeren Datensätzen noch vollständig in den RAM passen.

Die Tabelle *Tokens* enthält alle im Datensatz vorkommenden NerTag-Token-Kombinationen und deren Häufigkeit. Sie wird im Verlauf der Arbeit als Tokenliste bezeichnet. *Text* ist der Text des Tokens und *NerTag* der Tag, den die NLP-Pipeline (siehe Abschnitt 4.1) dem Token zugeordnet hat. Ein Token taucht dann in Kombination mit verschiedenen NerTags auf, wenn es je nach Kontext verschiedene Bedeutungen hat und diese erkannt wurden. So kann Beispielsweise “Guttenberg” je nach Kontext zum einen als Ort und zum anderen als Person (“Karl-Theodor

zu Guttenberg") klassifiziert werden, was eine explizite Unterscheidung bei der Suche nach bestimmten Dokumenten erlaubt.

5 Implementierung

Dieses Kapitel zeigt die Anwendung aus der technischen Perspektive. Es diskutiert die Wahl der verwendeten Technologien und geht auf einzelne Implementierungsdetails ein. Außerdem werden technische Schwierigkeiten diskutiert, die bei der Umsetzung der Visualisierung und der Datenvorverarbeitung auftraten und wie diese adressiert wurden.

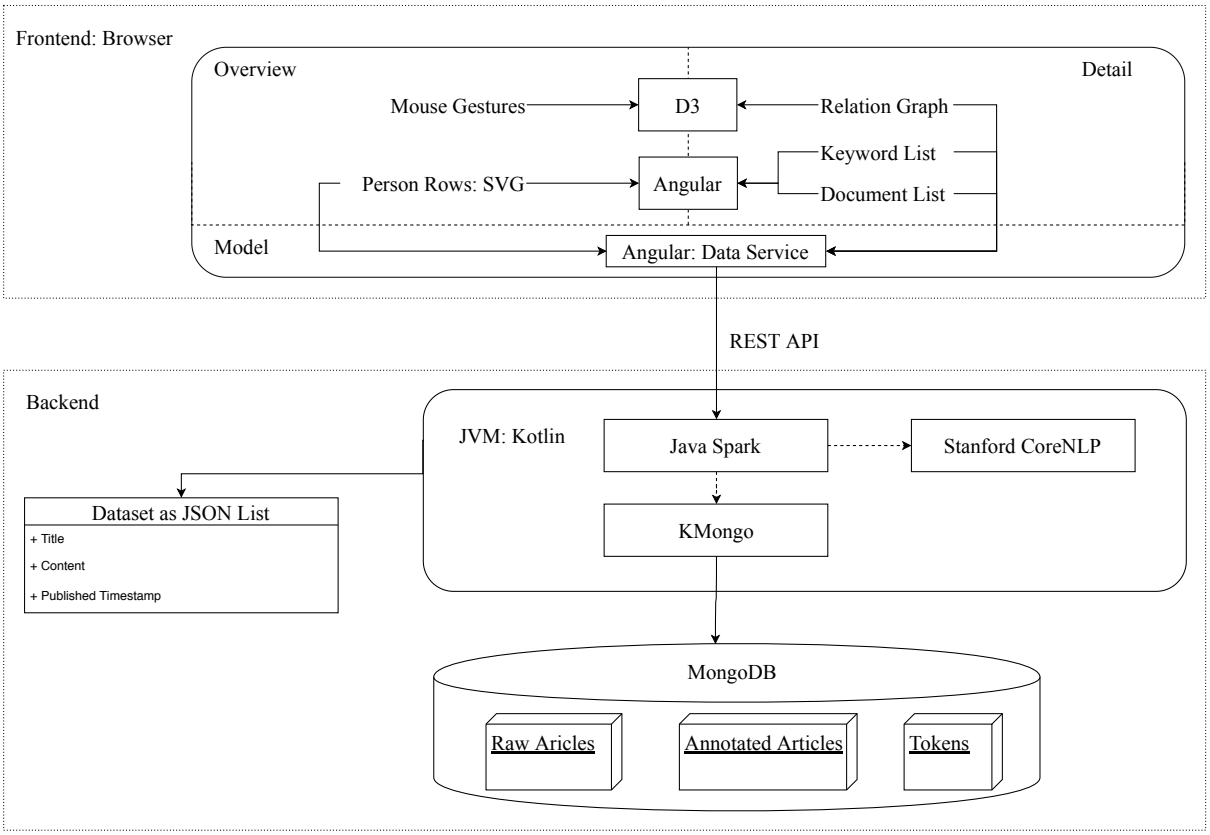

Abbildung 5.1: Architekturüberblick

Die Abbildung 5.1 zeigt einen Überblick über die Client-Server-Architektur, welche in diesem Kapitel ausgeführt wird.

5.1 Implementierung der Visualisierung

Die Visualisierung bildet das Frontend der Anwendung und ist webbasiert. Im Folgenden werden die architekturellen Designentscheidungen und Implementierungsdetails beschrieben.

5.1.1 Wahl des Technologie-Stacks

Die generelle Wahl der Programmierumgebung fiel auf die der Webanwendung. Der Grund dafür war die hohe Flexibilität und Zeiteffizienz bei der Entwicklung durch ein umfangreiches Ökosystem von Third-Party-Libraries. Als Stack kam eine Kombination aus Angular[1] (TypeScript, HTML, SCSS) und D3[2] (JavaScript) zum Einsatz, um eine Visualisierung basierend auf SVG (Scalable Vector Graphics) zu erzielen. Die Kommunikation zum Server erfolgt per REST.

SVG wurde als Grundlage für die grafische Umsetzung gewählt, denn:

- Aktuelle Browser können SVG unter Nutzung der Grafikkarte effizient rendern.
- Die Interaktion der Maus mit einzelnen Grafikelementen ist ein Standardfeature, welches die Umsetzung der Nutzerinteraktion vereinfacht.
- SVG wird mittels des HTML-DOMs beschrieben. Das ermöglicht eine Manipulation des SVGs durch JavaScript, was eine dynamische Änderung der Darstellung zur Laufzeit sowohl einfach, als auf effizient macht.
- SVG kann per CSS-Transitionen und CSS-Animationen animiert - und einzelne Elemente oder Gruppen von Elementen einfach skaliert oder verschoben werden.

Als Schwäche von SVG ist im Laufe des Projekts zu tragen gekommen, dass zu viele einzelne grafische Elemente den DOM aufblähen und die Performance schmälern können. Um diesem Problem entgegenzuwirken, können gruppierte Elemente, wie beispielsweise eine Gruppe von Messpunkten auf ein Canvas gezeichnet werden, welches dann über ein *foreignObject* in das SVG eingebunden wird.

Angular ist ein umfangreiches Framework von Google für Single-Page-Applications. Aufgrund folgender Features wurde es für diese Arbeit ausgesucht:

- Angular ermöglicht eine effiziente DOM-Manipulation mit einer Template-Engine und einem DOM-Renderer.

[1] https://angular.io/
[2] https://d3js.org/

- Das Data Binding von Angular kann dazu genutzt werden, um ein Datenmodell in den DOM und somit das SVG zu überführen und Änderungen an den Daten direkt auf das SVG zu spiegeln, was das Aktualisieren der Anzeige bei einer Änderung der Daten vereinfacht.
- Angular erleichtert das Entwerfen und Pflegen einer skalierbaren Architektur durch die Unterteilung der Anwendung in Module, Komponenten, Direktiven und Services.
- Mit webpack[3] liefert Angular eine einfache Möglichkeit, sämtliche Ressourcen zu einer kompakten Applikation zu bündeln.
- Der LiteServer von Angular beschleunigt den Entwicklungsprozess, indem er Änderungen am Quelltext innerhalb weniger Sekunden als gebaute Anwendung in einem Webbrowser anzeigt.

D3 ist ebenfalls ein Framework, das den DOM von SVGs rendert und darüber hinaus einige fachliche Funktionen für Visualisierungen wie beispielsweise Graph-Layouts mitbringt. In dieser Arbeit wurde D3 dazu verwendet, um die Zoom- und Dragging-Interaktion in der Übersichtsseite sowie die Graphen-Visualisierung in der Detailseite umzusetzen.

5.1.2 Algorithmen und Datenstrukturen

Das Herzstück der Übersichtsseite bildet ein zweistufiges Datenmodell: Das erste beschreibt die Daten, die der Client vom Server anfragt und das zweite - das View-Model - ist eng an die Visualisierung gekoppelt. Angulars Data Binding wird dazu verwendet, das View-Model direkt an den Dom des SVGs der Übersichtsseite zu binden, welches dafür sorgt, dass Änderungen am Modell direkt auf das SVG und somit die Visualisierung übertragen werden.

Das ermöglicht die in Abbildung 5.2 dargestellte Visualisierungspipeline:

1. Daten für den aktiven Zeitraum werde vom Server angefragt und in das erste Datenmodell - das Server-Model eingefügt.
2. Ein Layout-Algorithmus (Siehe Abschnitt 5.1.3) überführt unter Berücksichtigung von Nutzereingaben das Server-Model in das View-Model. Er wird durch Nutzereingaben oder beim Eintreffen von neuen Daten vom Server angestoßen.
3. Angulars Data Binding überführt das View-Model in den DOM des SVGs, das die gesamte Übersichtsseite abdeckt. Die exakten Instruktionen, wie diese Überführung durchgeführt wird, wird durch *HTML Templates* definiert, welche an Angulars Template-Engine übergeben wird.

[3] https://webpack.js.org/

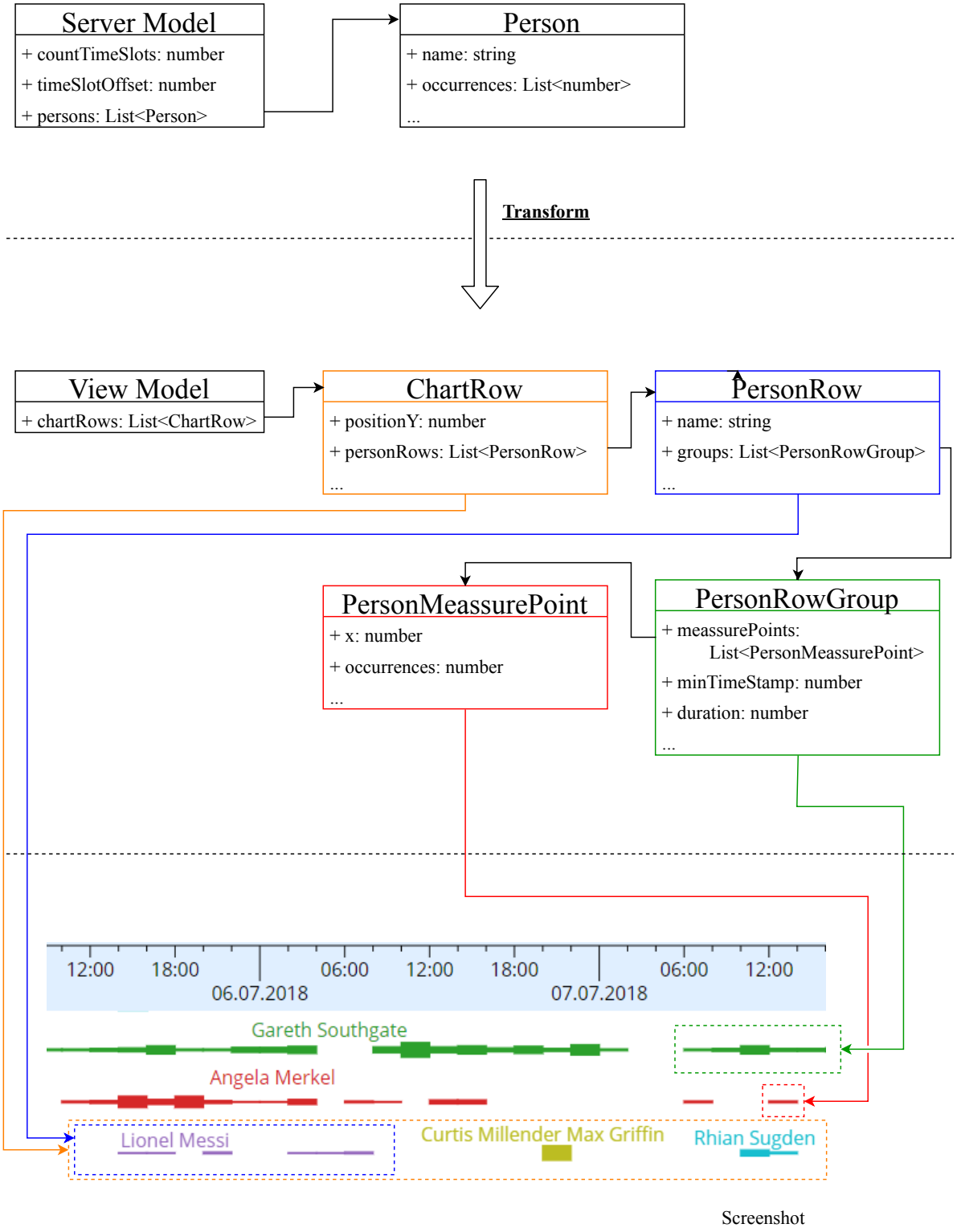

Abbildung 5.2: Data Binding Pipeline

Abbildung 5.2 zeigt das Server-Model (oben), das View-Model (farbig in der Mitte) und die konkrete Verknüpfung mit der Darstellung (Screenshot). Die farbigen Verbindungslinien zwischen dem Screenshot und dem View-Model symbolisieren die Überführung (Schritt 3) in den SVG-Dom durch das Data Binding. Um die Darstellung nicht stärker zu überladen als nötig, wurde das Datenmodell auf die wesentlichen Felder reduziert und die gestrichenen, welche beispielsweise für den Layout-Algorithmus benötigt werden, durch jeweils drei Punkte abgekürzt. Der Schritt zwei der Visualisierungspipeline beschreibt den Übergang, der in der Abbildung als **Transform** markiert ist.

Das Server-Model beinhaltet Informationen über die Anzahl der Zeitspannen *(countTimeSlot)*, den Beginn des Zeitraums *(timeSlotOffset)* und die liste der Personen, die in diesem Zeitraum in den Nachrichtendaten gefunden wurden. Das Feld *(occurrences)* in *Person* hält die Anzahl der Dokumente, die in der Zeitspanne veröffentlicht wurden, in welchen diese Person erwähnt wird. Die *occurrences* werden durch den Layout-Algorithmus in die *meassurePoints* des View-Models überführt, welche in der Abbildung rot umrahmt sind.

Das View-Model ist umfangreicher als das Server-Model, da es an die Struktur des SVGs gekoppelt ist, während das Server-Model in erster Linie kompakt sein soll. Die *ChartRow* entspricht einer Spur der Darstellung und enthält das Feld *positionY* für die vertikale Positionierung der Zeile. In jeder *ChartRow* können sich beliebig viele *PersonRows* befinden, welche ihrerseits alle Erwähnungen einer Person im aktiven Zeitraum beinhaltet. *PersonRows* sind in Gruppen (*PersonRowGroups*) unterteilt, welche dadurch definiert sind, dass alle enthaltenen, aufeinander folgenden *meassurePoints* nicht weiter als eine feste Zeitspanne (hier 9 Stunden) voneinander entfernt liegen. Diese Gruppierung findet hauptsächlich im Layout-Algorithmus Verwendung und garantiert dort, dass innerhalb einer Gruppe keine *meassurePoints* anderer Personen “eingefügt” werden. *PersonMeassurePoints* sind den Erwähnungsbalken zugehörig (hier rot umrahmt). Ihre *occurrences* werden linear auf die Höhe der Balken übertragen.

5.1.3 Der Layout-Algorithmus für die Übersichtsseite

Die Routine (Algorithmus 5.1), welche ein wesentlicher Bestandteil der Überführung des Server-Models in das View-Model darstellt, beeinflusst folgende Attribute der Visualisierung: 1.: Die Reihenfolge, mit der die Personen auf die Spuren von oben nach unten einsortiert werden. 2.: Die Höhe der einzelnen Spuren. 3.: Welche Personen sich gemeinsame Spuren teilen. 4.: Wo sich die Labels der einzelnen Personen befinden.

Das Vorgehen ist dabei in 4 Schritte unterteilt:

1. Sortiere die darzustellenden Personen nach Relevanz absteigend.
2. Füge jede Person an höchstmöglicher Stelle in die Spuren in der Darstellung ein. Die Reihenfolge der Personen entspricht der Sortierung aus Schritt 1. Dazu wird von oben nach unten die erste Spur gesucht, in der die Erwähnungsbalken der Person nicht mit

Algorithmus 5.1 Calculate View Layout

```
1:  procedure layoutPersonRows(personRows: List<PersonRow>)
2:      chartRows ← new List          // Die Spuren, die auf der Oberfläche angezeigt werden
3:      sort personRows descending by occurrences          // Iteriere über die Personen mit absteigender Relevanz
4:      for every personRow in personRows do
5:          fittingChartRow ← findFittingChartRow(personRow)     // Füge die Personen in die erste Spur ein, in der sie mit keiner Person überlappt, die vor ihr eingefügt wurde
6:          if fittingChartRow IS NULL then
7:              addToList(fittingChartRow.rowItems, personRow)
8:              personRow.posYIndex ← fittingChartRow.index
9:              personRow.visible ← true
10:         else
11:             personRow.visible ← false
12:         end if
13:     end for
14:     positionY ← 0     // Setze die vertikale Position der Spuren in Abhängigkeit von der Position der darüber liegenden Spur und ihrer eigenen Höhe
15:     for every chartRow in chartRows do
16:         halfHeight ← chartRow.calculateHeight() / 2 + PADDING
17:         positionY ← positionY + halfHeight
18:         chartRow.positionY ← positionY
19:         positionY ← positionY + halfHeight
20:         organizeLabelsOfPersons(chartRow)   // Erstelle ein gemeinsames Label für alle PersonRowGroups innerhalb der chartRow, die nicht durch eine Gruppe einer anderen Person getrennt werden
21:     end for
22:     return chartRows
23: end procedure
24: procedure findFittingChartRow(personRow: PersonRow, chartRow: List<ChartRow>)
25:     for chartRow of chartRows do
26:         intersectingChartRow ← FIND ITEM IN chartRow.rowItems
27:             WHERE rowItem INTERSECTS personRow
28:         if intersectingRow IS NULL then
29:             return chartRow
30:         end if
31:     end for
32:     newChartRow ← new ChartRow() // Erstelle eine neue Spur, da auf den vorhanden kein Platz für die Person gefunden wurde
33:     addToList(chartRows, newChartRow)
34:     return newChartRow
35: end procedure
```

einer den bereits vorhandenen kollidieren. Wird keine Spur gefunden, so wird eine neue Spur unter den bisherigen eingefügt und die Person auf diese Spur gesetzt.

3. Berechne die vertikale Position jeder Spur, indem ihre eigene Höhe zu der Position der darüber liegenden Spur addiert wird. Die Höhe entspricht dem höchsten Erwähnungsbalken der Personen, welche in Schritt 2 der Spur zugeordnet wurden.

4. Erstelle für jede Erwähnungsbalken-Gruppe einer Person auf einer Spur, welche nicht durch die einer anderen Person unterbrochen wird, ein Label mit dem Namen der Person. So werden so wenige Labels wie möglich auf der Oberfläche dargestellt, ohne einzelne Gruppen unbeschriftet lassen zu müssen.

5.2 Implementierung des Backends

Das Backend der Anwendung dient der Datenvorverarbeitung und der Speicherung. Die Logik für die Erkennung einzelner Personen in den Nachrichtenartikeln, sowie das Mapping der Daten für die REST-Schnittstelle zum Frontend ist hier untergebracht.

5.2.1 Technologiewahl des Backends

Der Kern des Servers stellt das Framework Stanford CoreNLP (Abschnitt 4.1) zur Verarbeitung natürlicher Sprache dar. Da das Framework hauptsächlich in Java geschrieben wurde und auf der JVM (Java Virtual Machine) ausgeführt wird, wurde diese auch als Plattform für den Server gewählt. CoreNLP wurde in der Arbeit dazu verwendet, aus den Nachrichtenartikeln die einzelnen Personen als sogenannte *Named Entities* zu extrahieren. Ein weiterer Grund für die Wahl der JVM ist die gute Balance aus Laufzeitperformanz und Komfortfunktionen, wie die Garbage Collection, was den Programmieraufwand gering halten soll. Zudem war eine einfache Möglichkeit zur parallelen Ausführung auf mehreren Kernen wichtig für die rechenintensive Vorverarbeitung mithilfe des NLP Tools.

Als Programmiersprache wurde Kotlin verwendet, da sie zu Java kompatibel - und die persönliche Präferenz des Autors gegenüber Java ist. Der einzige architekturelle Unterschied, den die Entscheidung für Kotlin gegenüber Java mit sich brachte, war die Verwendung des Datenbankadapters, was im weiteren Text beschrieben wird.

Die Ansprüche an die Datenbank waren zu Beginn des Projekts hauptsächlich das einfache Ablegen und Anfragen von bearbeiteten Datensätzen, was praktisch alle gängigen Datenbanken erfüllen. Die Wahl fiel auf die dokumentenorientierte Datenbank MongoDB[4], da diese aufgrund ihres dynamischen Datenmodells keiner aufwendigen Modellierung der Tabellen bedarf und schnelle Zugriffe auf Dokumente mit teilweise sehr unterschiedlicher Größe erlaubt. Ein Nachrichtenartikel und seine Metadaten, wie Datum der Veröffentlichung oder sein Titel werden als einzelnes Dokument im Sinne der Dokumentenorientierung geführt.

Als Datenbankadapter kommt KMongo[5] zum Einsatz. KMongo übernimmt das Mapping zwischen MongoDB-Dokumenten und Java, bzw. Kotlin-Objekten und bietet eine typisierte Abfragesprache.

5.2.2 Extraktion der Personen

Die Erkennung von namentlich erwähnten Entitäten wird in der maschinellen Sprachverarbeitung als *Named Entity Recognition*, kurz *NER* bezeichnet. Das generelle Vorgehen ist dabei,

[4] https://www.mongodb.com/
[5] https://litote.org/kmongo/

dass einzelne Worte als *Tokens* erkannt werden, bevor der NER-Annotator (beschrieben in Abschnitt 4.1) entscheidet, ob es sich dabei um eine Person handelt oder nicht und Ergebnis an das Token annotiert. Die Implementierung in dieser Arbeit nimmt die annotierten Tokens und speichert sie vorerst ohne eine weitere Verarbeitung in der Datenbank.

Eine Schwierigkeit, die dabei auftritt, ist, dass Personen häufig nicht bei vollem Namen genannt werden. So wird beispielsweise auf "Hillary Clinton" oft sowohl mit "Hillary", als auch mit "Clinton" verwiesen. Es kann außerdem vorkommen, dass in einem Nachrichtenartikel ausschließlich mit "Clinten" auf "Hillary Clinten" verwiesen wird. In der Visualisierung wäre es ungünstig, für eine Person ("Hillary Clinton") drei einzelne Entitäten ("Hillary", "Clinton", "Hillary Clinton") zu haben. Ein Betrachter, der sich "Hillary" anschaut, würde die Nachrichtenartikel, die nur den anderen zwei Namen zugeordnet sind, fälschlicherweise gar nicht sehen. Deshalb besteht der Anspruch, diese einzelnen Namen auf eine eindeutige Person zurückführen zu können. Ein naiver Ansatz um das umzusetzen wäre es, in einem Nachrichtenartikel alle Teil-Namen einem zusammengesetzten Namen zuzuordnen, der diese vollständig enthält und seinerseits im selben Dokument auftaucht. Das führt aber bei einer Sammlung von mehreren Nachrichtenartikeln zu einem Fehler: Dokument A bis E enthalten "Donald Trump", Dokument F enthält nur "Trump". Die mit hoher Wahrscheinlichkeit richtige Zuordnung wäre auch in Dokument F "Donald Trump", könnte aber mit dem naiven Ansatz nicht getätigt werden. Um auch über mehrere Dokumente hinweg mit einer geringen Fehlerrate einzelne Namen auf zusammengesetzte Name abbilden zu können, wurde in dieser Arbeit folgender Ansatz verfolgt:

Über den betrachteten Datensatz wird ein Graph aufgestellt, welcher die erkannten Namen als Knoten und Relationen zwischen den Namen als Kanten hat. Die Namen sind dabei entweder zusammengesetzt ("Hillary Clinton") oder einzeln und jeweils zusammen mit der Anzahl der Dokumente, in denen diese auftauchen, als Häufigkeitskomponente in den Knoten abgelegt. Eine Kante zwischen einem Paar von Namen reflektiert die Anzahl der Dokumente, in denen beide Namen gemeinsam auftauchen. Die Grafik 5.3 stellt einen solchen Graphen beispielhaft dar.

Die Fragestellung, die mit diesem Graph beantwortet werden kann, lautet wie folgt: Gegeben sei ein Name ("Trump") aus einem Dokument, welches sonst keine weiteren Namen enthält. Gibt es einen zusammengesetzten Namen im Graph, der diesen Namen vollständig enthält? Wenn ja, wie wahrscheinlich ist es, dass dieser zusammengesetzte Name auch tatsächlich der gewünschte Name ist?

Die Auswertung geht dabei wie folgt vor:

1. Wähle alle Knoten, die eine Kante zu "Donald" haben und deren Namen "Donald" als Teil-String enthalten.
2. Kombiniere das Gewicht der Kanten mit den Häufigkeitskomponenten der Zielknoten.
3. Wähle den Knoten mit der größten kombinierten Gewichtung.

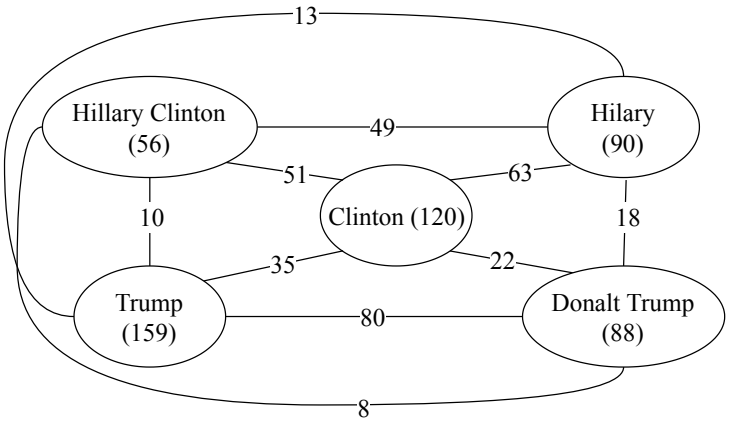

Abbildung 5.3: Relationen Graph zwischen Personen

4. Interpretiere die kombinierte Gewichtung als Wahrscheinlichkeit für eine korrekte Korrelation.

Bei der schlussendlichen Zusammenführung der Namen auf eindeutige Entitäten wird für jeden Namen ein spezifischerer Name mithilfe des Graphen gesucht. Wenn die errechnete Wahrscheinlichkeit über einer festgelegten Grenze liegt, wird der Name in den spezifischeren Namen überführt.

Der aufgestellte Graph wird außerdem in der Detailseite dazu genutzt, um die Relationen der Personen darzustellen.

5.2.3 Schlagwörter finden

Für einen schnellen Überblick über mögliche Geschehnisse in einem Zeitraum, mit denen eine Person in Verbindung gebracht wird, sollen die Schlüsselwörter der Detailseite dienen.

Um relevante Wörter (im Folgenden als Terme bezeichnet) algorithmisch zu finden, kommt hier das Tf·Idf-Ranking zum Einsatz. Als Datenquelle dienen die Nachrichtenartikel (Dokumente) aus diesem Zeitraum, in denen diese Person vorkommt. Die Anzahl dieser Artikel wird im Folgenden mit **N** benannt. Enthaltene Wörter im Inhalt der Artikel werden Terme, kurz **t**, genannt. Jeder Term bekommt einen Tf·Idf Ranking Score **R** und die Terme mit dem höchsten Score werden als relevant erachtet.

Der Score R errechnet sich je Term wie folgt:

$$R_{t,d} = (1 + \log tf_{t,d}) \cdot \log \frac{N}{df_t} \tag{5.1}$$

$tf_{t,d}$ *(Term Frequency)* beschreibt wie häufig der Term *t* im Dokument *d* vorkommt. df_t *(Document Frequency)* ist die Anzahl der Dokumente, in denen der Term *t* insgesamt vorkommt.

Ein Term bekommt also einen höheren Score, wenn er häufiger pro Dokument vorkommt, und/oder er insgesamt in weniger Dokumenten auftritt.

Um das Ergebnis zu weiter zu verbessern, kommt eine Stoppwortliste für die jeweilige Sprache zum Einsatz. Auf den Stoppwortlisten stehen Pronomen, Artikel und andere Wörter, die bekanntermaßen keinen oder nur wenig Kontextinformation beinhalten.

Zur Entwicklungszeit kam ein Datensatz mit etwa 1,7 Millionen einzelnen Nachrichtenartikeln und einer Größe von etwa 10 Gigabyte zu Einsatz. Um ein effizientes Tf·Idf-Ranking zu ermöglichen, werden für das Ranking nicht die ursprünglichen Dokumente *(Raw Articles)* verwendet, sondern die im Kapitel 4.2 (Datenbank Schema) erwähnten *Annotated Articles*. Die *tokens*-Liste der *Annotated Articles* wird anstelle des reinen Textes verwendet, um die *Term Frequency* zu errechnen. Da in den dort hinterlegten Tokens die Wörter durch jeweils eine 32-Bit Zahl (die ID des Tokens in der Tokenliste) ersetzt wurden, sind die *Annotated Articles* im Schnitt bei dem zugrundeliegenden englischen Datensatz nur etwa ein Neuntel so groß wie die *Raw Articles*. Mit den Indices aus den Tokens wird über die Tokenliste der Datenbank die *Document Frequeny* bezüglich der gesamten Textsammlung abgefragt.

Die geringere Größe der Dokumente und das Arbeiten auf Indices anstatt auf Strings hat drei wesentliche Vorteile: Erstens müssen bei der Datenbankabfrage weniger Daten geladen, durchsucht und übertragen werden, zweitens sind Operationen auf Integer-Arrays sowie der Vergleich von Integern mit Integern wesentlich schneller als der von Strings mit String und drittens passen die geringeren Datenmengen, auf denen gearbeitet wird, besser in die CPU-Caches, was ebenfalls für eine bessere Laufzeit sorgt. Die Ergebnisse des Tf·Idf-Rankings sind dann die Token-Indices (32-Bit Integer) gepaart mit dem Ranking Wert, die mithilfe der Tokenliste wieder zurück in die ursprünglichen Worte übersetzt werden, um sie dem Nutzer präsentieren zu können.

Eine weitere Verbesserung der Ergebnisse konnte erzielt werden, indem Wörter im Volltext der Nachrichtenartikel in der Datenvorverarbeitung mittels Stamming in deren Stammform überführt wurden. “laufen”, “liefen”, oder “laufender” werden Beispielsweise dort alle in “laufen” überführt. Das Stamming erfüllt einen ähnlichen Zweck wie das Zuordnen von Namen von Personen zu eindeutigen Entitäten, welches in Unterabschnitt 5.2.2 beschrieben wurde: Es fasst alle Wörter zusammen, die dieselbe Bedeutung haben. Anstelle von “Unwetter” und “[des] Unwetters” steht in den Schlagworten der Detailseite somit nur “Unwetter”.

Da die Schlagwörter die Ereignisse des vom Nutzer angewählten Zeitraums widerspiegeln sollen, wurde damit experimentiert, die Tf·Idf-Rankings gegen jene aus dem Zeitraum davor und danach für die entsprechende Person abzugleichen. Damit konnten Wörter ausgefiltert werden, die generell häufig mit einer Person auftauchen (beispielsweise “Chancellor” bei “Angela Merkel“). Wählt man nur einen einzelnen Tag als betrachteten Zeitraum aus, so fällt auf, dass häufig der jeweilige Wochentag als relevantestes Wort gewertet wird, was ein durchaus erwartetes Ergebnis ist. Gegenüber dem nicht-vergleichenden Ranking liefert dieses Vorgehen häufiger Schlagworte, die signifikant für die Ereignisse an genau diesem Tag sind, lässt aber

Anfrage	∅ Antwortgröße komprimiert / unkomprimiert	∅ Antwortzeit
Übersichtsseite: Initial	45 / 224kb	4500ms
Übersichtsseite: Erweiterung um neue Daten	25 / 93kb	700ms
Detailseite: Schlüsselwörter und Graph	1,4 / 5kb	20-200ms
Detailseite: Liste der Dokumente	5 / 16kb	200-600ms
Detailseite: Volltext eines Artikels	1-10kb / 2-20kb	20ms

Tabelle 5.1: Antwortzeiten in der Anwendung

Kontext zu längerfristigen Ereignissen weg. Aus diesem Grund wird der Ansatz ergänzend zu den bisherig gefundenen Schlagwörtern in der hiesigen Implementierung verwendet.

5.2.4 Skalierbarkeit und Performanz

Der Server-Part der Anwendung wurde so konzipiert, dass neue Anfragen an den Server ohne großen Aufwand implementiert - und bestehende Anfragen abgeändert werden können. Um das erzielen zu können, wurde fast vollständig auf die Aggregation der Daten in der Vorverarbeitung verzichtet, was eine höhere Laufzeitperformanz ermöglicht hätte. So muss beispielsweise bei der Anfrage für die Daten der Übersichtsseite, welche nur die Anzahl der Erwähnungen der Personen, aggegiert in Zeitspannen von mehreren Stunden enthält, sämtliche Dokumente für diesen Zeitraum aus der Datenbank abgefragt werden. Diese Daten könnten beim Einpflegen neuer Nachrichtenartikel in aggregierter Form als zusätzliche Datenstrukturen in der Datenbank abgelegt werden, was den Zugriff zur Laufzeit für den 10 Gigabyte großen Beispielsdatensatz von einigen hundert Millisekunden auf wenige Millisekunden reduzieren würde.

Mit dem Wissen, dass folgende Angaben zu den Verarbeitungszeiten für Serveranfragen aus den oben genannten Gründen nur unter Vorbehalt zu genießen sind, zeigt die Tabelle 5.1 die REST-Requests vom Web-Client an den Server. Der zugrundeliegende Datensatz umfasst 1,783,635 Nachrichtenartikel, welche in der Datenbank 10,9GB Speicher beanspruchen. Die annotierten und komprimierten Dokumente, die aus der Datenvorverarbeitung hinausgingen und für nahezu alle REST Anfragen verwendet werden, bestehen aus ebenfalls 1,783,635 einzelnen Dokumenten in der Datenbank mit einer durchschnittlichen Größe von 724,3 Bytes - insgesamt also 1,2 GB.

Als Hardware kam ein Server mit 16 AMD Epyc Kernen mit einem Basistakt von jeweils 2400Mhz, 32GB DDR4 RAM und 64GB SSD-Speicher zum Einsatz. Die Latenz zum Server lag zum Zeitpunkt der Messung bei durchschnittlich 10 Millisekunden.

Diese Antwortzeiten wären für eine Anwendung für Endanwender zu lang, könnten aber mit entsprechender Optimierung durch vor-aggregierte Datenstrukturen relativ einfach auf ein Niveau gehoben werden, das einer Echtzeitanwendung gerecht wird.

Bis auf eine Ausnahme besteht in der aktuellen Umsetzung keine Datenstruktur, die eine Aggregation bisheriger Daten darstellt. Dadurch kann das Datenmodell des Servers inkrementell um neue Nachrichtenartikel erweitert werden, ohne bisherige Daten in den Importprozess einbeziehen zu müssen. Der Zeitaufwand für den Import entspricht somit dem Aufwand, der bei dem initialen Import eines großen Datensatzes pro Nachrichtenartikel anfällt, welcher sich auf wenige Millisekunden beläuft. Damit wäre eine Echtzeitanwendung mit dem inkrementellen Hinzufügen neuer Nachrichtenartikel zum Zeitpunkt derer Veröffentlichung möglich. Die genannte Ausnahme bildet die Tokenliste, welche die Document-Frequency für das Tf·Idf-Ranking für jedes Token enthält. Wird dem Datenmodell ein weiterer Nachrichtenartikel hinzugefügt, muss zum einen die Tokenliste um diejenigen Tokens erweitert werden, welche im Artikel, jedoch bisher nicht in der Liste enthalten sind. Zum andern müssen die bisherigen TF-Werte der Tokens, die sowohl im Artikel, als auch in der Tokenliste vorkommen, inkrementiert werden. Da dabei keine Betrachtung bisheriger Nachrichtenartikel notwendig ist, geht auch dieses Update der Daten mit sehr geringem Speicher- sowie CPU-Aufwand.

6 Evaluierung

In diesem Kapitel wird ein Anwendungsszenario für die Implementierung vorgestellt, das die Grundlage für eine Expertenbefragung war. Die Expertenbefragung wurde vom Autor dieser Abschlussarbeit durchgeführt und dient zur Einschätzung der Arbeit aus der Sicht von Visualisierungexperten. Das Szenario soll beispielhaft aufzeigen, wie Informationen über Personen und Ereignisse, mit denen die Personen in Verbindung stehen, aus der Visualisierung abgelesen werden können.

6.1 Anwendungsszenario

Betrachtet werden soll der Zeitraum zwischen dem 13.06.2018 und dem 16.06.2018. Folgende Fragen sollen mithilfe der Anwendung beantwortet werden können:

1. Mit welchen Ereignissen wird Angela Merkel in diesem Zeitraum vorrangig in Verbindung gebracht?
2. Welche Personen tauchen in diesem Zeitraum häufig gemeinsam mit Angela Merkel auf?
3. Wer sind die drei relevantesten Personen des Datensatzes in diesem Zeitraum? Teilen diese denselben Kontext?

Abbildung 6.1 zeigt exemplarisch den Vorgang, den die Befragten zum Erörtern der Fragen 1. und 2. durchlaufen haben. Die Pfeile symbolisieren die Übergänge, welche durch Mausklicks an der Stelle des Starts des Pfeils ausgelöst wurden. Folgende Schritte wurden dabei von den Befragten so oder in leicht abgewandelter Form gewählt:

1. Zum Zeitraum 13.06. bis 16.06.2018 scrollen und in der Zeitleiste markieren.
2. Mit einem Mausklick auf die Zeile, in der “Angela Merkel” auftaucht die Detailseite öffnen.
3. Schlüsselwörter überfliegen und nach hinweisen auf Ereignisse suchen.
4. “refugee”, “migration” oder von den Personen “Emmanuel Macron” als interessant erachten und anklicken.

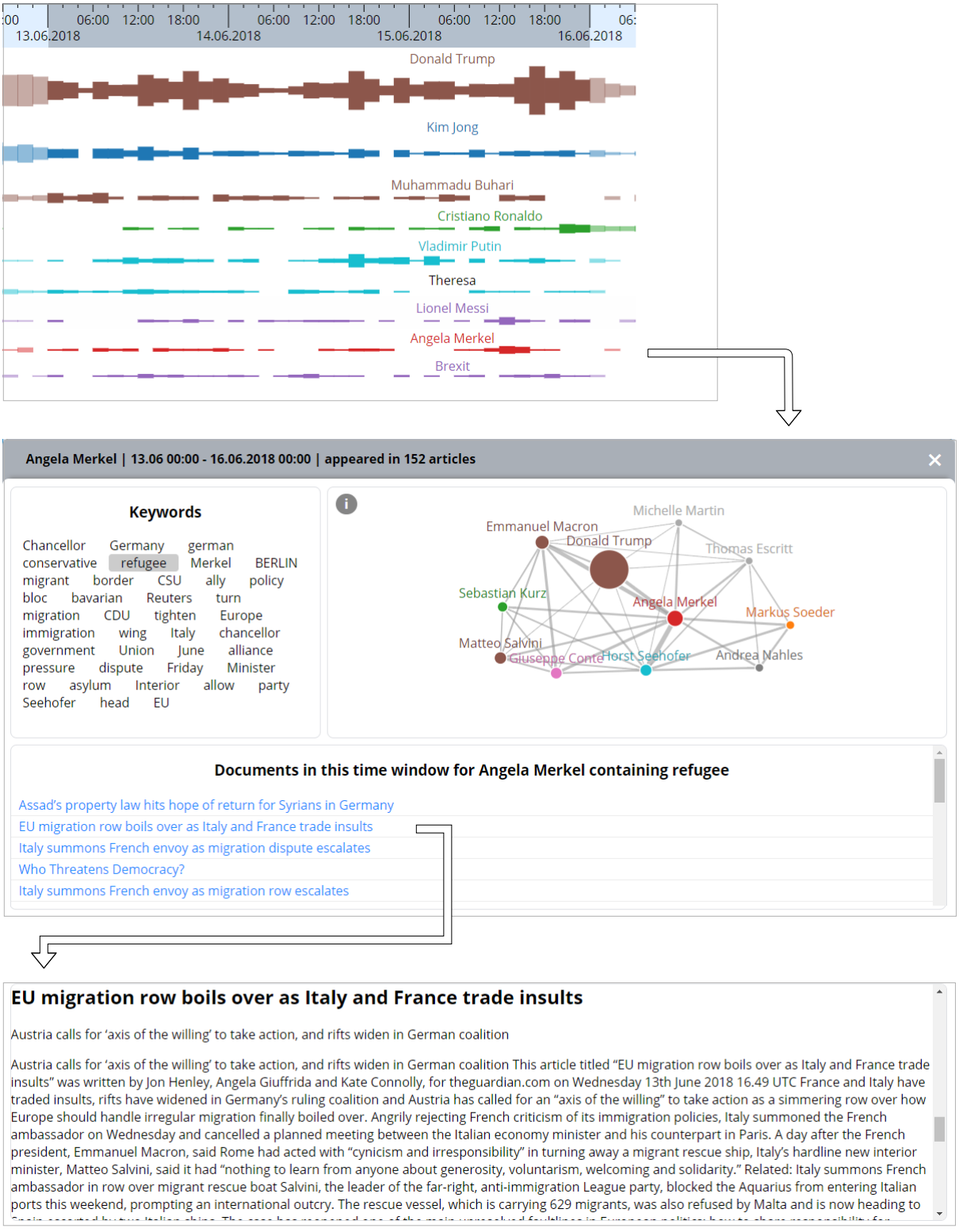

Abbildung 6.1: Screenshot: Drilldown vom 13.06. - 16.06.2018

5. Titel der Nachrichtenartikel überfliegen und gegebenenfalls den Volltext von Artikeln aufrufen und den Anfang überfliegen.

6.2 Auswertung

Dieser Abschnitt trägt die Ergebnisse der Befragung bezüglich des Anwendungsszenarios zusammen. Befragt wurden fünf wissenschaftliche Mitarbeiter im Fachgebiet Visual Analytics am Institut für Visualisierung und Interaktive Systeme (VIS) der Universität Stuttgart. Alle Befragten waren zu diesem Zeitpunkt Doktoranden.

Zu Frage 1: Alle befragten Personen konnten eine Antwort zur Frage liefern und jeder führte "die Flüchtlingskrise" als relevantes Ereignis im betrachteten Zeitraum an.

Zu Frage 2: Die Befragten nannten die im Personen-Relationen-Graph aufgeführten Personen mit den dicksten Verbindungslinien zu Angela Merkel. Zwei der Experten merkten hierzu an, dass der Graph diese Frage nur unbefriedigend erfüllt. Er diene als grober Überblick, jedoch seien nicht alle dort kodierten Informationen nachvollziehbar. Außerdem gebe der Graph nur spärliche Kontextinformationen und böte zu wenige Möglichkeiten zur Interaktion. Ein Grund dafür könnte sein, dass die Idee für das Feature des Personen-Relationen-Graphen sehr spät während der Arbeit entstanden ist und experimentell erörtert werden sollte, welchen Nutzen der Graph bieten kann. Gedacht war der Graph für eine grobe Übersicht der gemeinsam auftretenden Personen.

Zu Frage 3: Als relevanteste Personen in diesem Zeitraum wurden "Donald Trump", "Kim Jong" und "Muhammadu Buhari" genannt, da diese in der Übersicht die größten Erwähnungsbalken aufweisen und zudem zu oberst in der Visualisierung auftauchen. Die Frage, ob diese sich einen gemeinsamen Kontext teilen, empfanden die befragten Experten als wahlweise schwierig oder umständlich zu beantworten. Das lege daran, dass die Anwendung keine automatische Möglichkeit zu einem solchen Vergleich ermögliche. Es wäre erforderlich gewesen, jeweils zu den Personen die Detailseite aufzurufen und anhand der Artikel, der Schlüsselwörter und des Personen-Relationen-Graphen einzuschätzen, welchen Bezug die drei Personen zueinander haben.

Das generelle Stimmungsbild war, dass die Anwendung übersichtlich und einfach zu bedienen sei. Gelieferte Informationen seien umfangreich, ohne überladen zu wirken und die Möglichkeit, bis zur Ebene der tatsächlichen Nachrichtenartikel "hinunter" gehen zu können, sei sehr hilfreich, um sich ein detailliertes Bild von den Themen in den Nachrichten machen zu können.

Kritik wurde vor allem daran geäußert, dass sich mehrere Personen in der Übersichtsseite auf einer Spur befinden können, was fälschlicherweise eine Relation der Personen bezüglich ihres Kontext suggeriere. Das vertikale, animierte Verschieben von Personen in der Übersichtsseite beim Wechsel des betrachteten Zeitraums sei funktional genau dann, wenn man nur eine

Person im Auge behalten will. Wolle man hingegen mehrere Personen beobachten, sei dies aufgrund der gleichzeitigen Verschiebung mehrerer Personen oft nicht möglich.

Ein Vorschlag eines Befragten für eine mögliche Verbesserung ist, in den Schlüsselwörtern in der Detailseite farblich die Tf·Idf-Werte zu kodieren. Als weitere Verbesserung wurde vorgeschlagen, den Personen-Relationen-Graphen durch eine Adjazenzmatrix zu ersetzen, welche mehr Kontext und eine bessere Platzausnutzung bieten könnte. In der Ansicht des Volltextes von Nachrichtenartikeln könnten die auftauchenden Schlüsselwörter farblich hervorgehoben oder markiert werden. Das umstrittene Feature mehrerer Personen auf einer Spur, welches zum Platzsparen gedacht war, sollte optional und in einem Einstellungen-Menü an - und abschaltbar sein. Eine weitere Idee war es, in der Detailseite einen Bezug zwischen den Schlüsselwörtern und den Personen im Graph herzustellen. Dazu könnte man beispielsweise bestimmte Schlüsselwörter hervorheben, wenn der Nutzer mit der Maus über eine Person im Graph fährt.

In Abbildung 6.2 und 6.3 sind zwei der Verbesserungen zu sehen, die nach der Expertenbefragung aufgrund der Vorschläge und Anregungen entstanden sind.

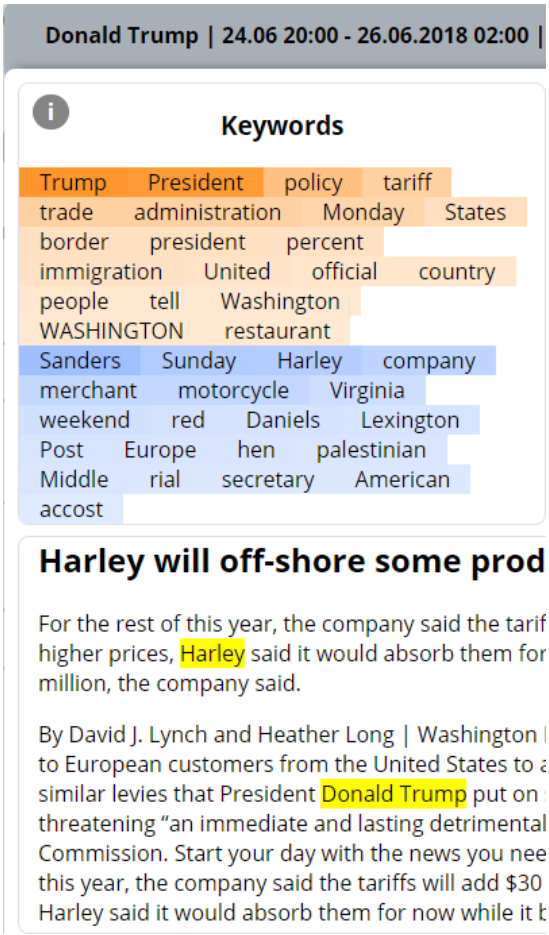

Abbildung 6.2: Überarbeitete Detailseite

Abbildung 6.2 zeigt die Umsetzung eine farbliche Hervorhebung der Schlüsselwörter in der Detailseite. Die orange hinterlegten Schlüsselwörter sind die herkömmlichen Schlüsselwörter, welche schon in den übrigen Screenshots der Detailseite zu sehen waren. Blau hinterlegte Schlüsselwörter sind expliziter auf den ausgewählten Zeitraum beschränkt, wie am Ende des Unterabschnitts 5.2.3: “Schlagwörter finden” genauer erläutert wird. Die Stärke der Farben Orange und Blau ergibt sich aus den Tf·Idf-Werten und soll die Nachvollziehbarkeit für die Bedeutung der Schlüsselwörter erhöhen.

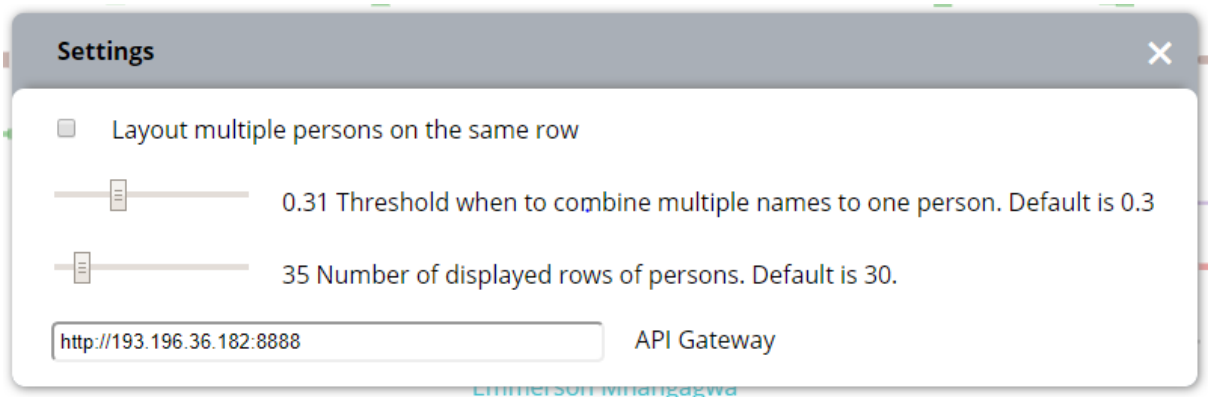

Abbildung 6.3: Einstellungsseite zur individuellen Anpassung

In Abbildung 6.3 ist die Implementierung einer einfachen Einstellungsseite zu sehen, von wo aus der Server-Endpunkt komfortabel während der Laufzeit geändert werden kann. Das Layouting mehrerer Personen auf eine gemeinsame Spur kann hier auf Wunsch des Nutzers abgestellt - und die Anzahl der angezeigten Spuren angepasst werden.

7 Zusammenfassung und Ausblick

Das Verfolgen von Nachrichten mit dem Fokus auf den erwähnten Personen ist ein Ansatz, der bislang noch wenig Beachtung fand. Diese Arbeit hat gezeigt, dass die Umsetzung des Ansatzes verhältnismäßig effizient möglich ist, da im Gegensatz beispielsweise zur ereignisorientierten Nachrichtenverfolgung, wie in EventRiver oder CloudLines kein aufwendiges und fehleranfälliges Clustering zur Event-Detektion nötig ist. Allerdings kann dieser Ansatz nicht direkt mit ereignisorientierter Nachrichtenvisualisierung verglichen werten, da einzelne Ereignisse mit dem in dieser Arbeit verfolgten Ansatz nicht so einfach von Nutzern zu erfassen sind.

Eine Stärke des hier verfolgten Ansatzes kann in der einfachen inkrementellen Erweiterung des Datensatzes gesehen werden. Diese ermöglicht eine vergleichsweise einfache Umsetzung einer Echtzeitanwendung, in welcher gerade erst veröffentlichte Artikel innerhalb weniger Sekunden in die Visualisierung eingebunden werden könnten.

Die dynamische Komponente der sich wandelnden Darstellung beim Ändern des betrachteten Zeitraums konnte in dieser Arbeit nicht mit abschließender Zufriedenstellung umgesetzt werden. Zu viel Bewegung auf dem Bildschirm erschwert es dem Benutzer, den Faden nicht zu verlieren, wenn er sich die Entwicklung mehrere Personen über einen größeren Zeitraum anschauen will. Was sich ebenfalls als schwierig erwiesen hat, ist die effiziente Ausnutzung des Platzes auf dem Bildschirm. Schuld daran ist die große Anzahl der potentiell interessanten Personen, die in den Nachrichtenartikeln erwähnt werden und die in dieser Arbeit verfolgten Idee, wie diese dem Betrachter präsentiert werden. Der hier gewählte (später optionale) Ansatz zum Eindämmen des Problems, indem Whitespace in selten auftauchenden Personen mit weiteren Personen gefüllt wird, wurde von Nutzern in einer Expertenbefragung teilweise als nicht intuitiv eingestuft, da dies auf Kosten der Übersichtlichkeit passiere.

Ausblick

Zukünftige Arbeiten könnten sich darauf konzentrieren, weiteren Kontext zu Personen zu liefern. Eine Möglichkeit dazu, die in dieser Arbeit angeschnitten wurde und einen erheblichen Mehrwert verspricht, ist das Gruppieren von Personen. Ein mögliches Kriterium anhand dessen gruppiert werden könnte, ist der gemeinsame Kontext, in dem diese Personen auftauchen. Würde man einzelne Personen hierarchisch zu Gruppen zusammenfassen, könnte man sie deutlich übersichtlicher darstellen. So könnte man außerdem Ereignisse, in welche mehrere Personen involviert sind, umfassender und gesammelt präsentieren, die momentan über die Detailseiten von mehreren Personen verteilt liegen.

Weitere Arbeiten könnten sich auch auf Basis der hier gesammelten Erkenntnisse damit beschäftigen, wie die inkrementelle Darstellung beim Wechseln des betrachteten Zeitraums so gestaltet werden könnte, dass sie den Nutzer besser beim Verstehen der Änderung innerhalb der Daten unterstützt.

Literaturverzeichnis

[AKK96] M. Ankerst, D. Keim, H.-p. Kriegel. „Circle Segments : A Technique for Visually Exploring Large Multidimensional Data Sets“. In: *First publ. in: Visualization '96, Hot Topic Session, San Francisco, CA, November, 1996* (Jan. 1996) (zitiert auf S. 13).

[DL10] M. K.W.R.D. K. Dongning Luo Jing Yang. „EventRiver: Visually Exploring Text Collections With Temporal References“. In: *IEEE Transactions on Visualization and Computer Graphics* 18.1 (Okt. 2010), S. 93 –105. DOI: 10.1109/TVCG.2010.225 (zitiert auf S. 15).

[Hav+02] S. Havre, E. Hetzler, P. Whitney, L. Nowell. „ThemeRiver: Visualizing Thematic Changes in Large Document Collections“. In: *IEEE Transactions on Visualization and Computer Graphics* 8.1 (Jan. 2002), S. 9–20. ISSN: 1077-2626. DOI: 10.1109/2945.981848. URL: http://dx.doi.org/10.1109/2945.981848 (zitiert auf S. 13).

[Kei+06] D. A. Keim, F. Mansmann, J. Schneidewind, H. Ziegler. „Challenges in Visual Data Analysis“. In: *Proceedings of the Conference on Information Visualization*. IV '06. Washington, DC, USA: IEEE Computer Society, 2006, S. 9–16. ISBN: 0-7695-2602-0. DOI: 10.1109/IV.2006.31. URL: http://dx.doi.org/10.1109/IV.2006.31 (zitiert auf S. 9).

[MKK11] E. B. Milos Krstajic, D. A. Keim. „CloudLines: Compact Display of Event Episodes in Multiple Time-Series“. In: *IEEE Transactions on Visualization and Computer Graphics* 17.12 (Nov. 2011), S. 2432–2439. DOI: 10.1109/TVCG.2011.179 (zitiert auf S. 14).

[Man+14] C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, P. Inc, S. J. Bethard, D. Mcclosky. „The Stanford CoreNLP natural language processing toolkit“. In: *In ACL, System Demonstrations*. 2014 (zitiert auf S. 23).

[WC11] M. I.L.T.C.S.Y.S.M.I.Z.J.G.X.T.H. Q. Weiwei Cui Shixia Liu. „TextFlow: Towards Better Understanding of Evolving Topics in Text“. In: *IEEE Transactions on Visualization and Computer Graphics* 17.12 (Nov. 2011), S. 2412 –2421. DOI: 10.1109/TVCG.2011.239 (zitiert auf S. 16).

Alle URLs wurden zuletzt am 22. 03. 2019 geprüft.

Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

Ort, Datum, Unterschrift