Goethe Universität
Institut für Informatik
Texttechnologie

PRG-Praktikum

WiSe 2022 / 2023
Abschlussprojekt

Alexander Mehler Giuseppe Abrami

09.01.2023

Parliament Browser

Einleitung

Die Abschlussaufgabe im Programmierpraktikum im WiSe 2022 / 2023 besteht darin, einen Parliament Browser gemeinschaftlich als Gruppenprojekt zu entwickeln. Zur Lösung dieses Gruppenprojekts haben sich die Teilnehmer*innen jeweils in Gruppen mit je 3 bis 5 Personen zusammengefunden. Die bisherigen Übungen dienten dem Zweck, die Studierenden auf die Erledigung dieser Gruppenaufgabe vorzubereiten. In der Gruppenarbeit wird sichtbar, wie wichtig die Nutzung einer Projekt- und Quellcode-Verwaltung ist. Hierzu verwenden wir, wie auch schon in den Übungen, GitLab[1].

Kompensation der Gruppengröße

Die nachstehenden Aufgaben sind unabhängig der Gruppengröße zu bearbeiten. Allerdings werden Gruppen, die weniger als 5 Mitglieder haben, auf die erreichten Punkte einen Faktor erhalten, um die fehlenden Teammitglieder zu kompensieren.

Vorbereitungen

Jede Gruppe erstellt zur Bearbeitung der Abschlussaufgabe ein neues *privates* Projekt in GitLab und nennt dieses **Parliament_Browser_X_N**, wobei „X_N“ für Ihren Gruppennamen steht[2]. Im Anschluss ist dieses Projekt **allen** Gruppenmitgliedern sowie Ihrer Tutorin / Ihrem Tutor freizugeben, so dass alle damit arbeiten können.

Zugangsdaten MongoDB

Ihre Zugangsdaten für die MongoDB werden in Ihr GitLab-Projekt kopiert. Bitte legen Sie Ihr Projekt daher frühzeitig an.

[1] https://gitlab.texttechnologylab.org
[2] Gruppe 1_Montag_3 wäre dann Parliament_Browser_1_3!

Vorbemerkungen

Grundsätzlich sind alle Ergebnisse ausführlich zu dokumentieren. Bezüglich der Dokumentation beachten Sie bitte die dazugehörige Aufgabe 4. Die Dokumentationsqualität fließt, wie auch in den vorherigen Übungen, in die Bewertung mit ein. Die Verwendung anderer als die angegebenen Programmiersprachen ist nicht erlaubt. Diese Übung ist in **Gruppenarbeit** zu absolvieren, jedoch ist die **individuelle Einzelleistung** kenntlich zu machen. Hierzu gehört auch die eigenständige Nutzung von GitLab.

Individuelle Einzelleistung. . .

. . . bedeutet, dass sichtbar und erkennbar dokumentiert sein muss, welchen Beitrag jede*r Teilnehmende der Gruppe zum Projekt geleistet hat. Jede*r Teilnehmende wird auch nur für den geleisteten Beitrag bewertet. Der jeweilige Verdienst am Projektfortschritt bzw. am Projektergebnis fließt anteilig ein.

Aufgabenbeschreibung

Sie entwickeln als Gruppe einen Parliament Browser bestehend aus zwei Komponenten: einer Server-Komponente (Backend), implementiert in Java, sowie einer Client-Komponente (Frontend), implementiert in JavaScript unter Verwendung verschiedener, unten aufgeführter Bibliotheken. Beide Komponenten betrachten wir abschließend als „Gesamtapplikation“, welche die folgenden, teils in vorhergehenden Übungen implementierten Funktionalitäten beinhalten soll:

1. Einlesen von Parlamentsdebatten des Deutschen Bundestages der 19. sowie der aktuellen Legislaturperiode im XML-Format.
2. Analysieren, Aufbereiten und datenbankgestütztes Abspeichern der abgefragten Parlamentsreden.
3. Abbildung der Parlamentsdebatten sowie der Abgeordneten-Stammdaten, gemäß Aufgabenbeschreibung.
4. Verarbeitung der Parlamentsreden mittels angegebener NLP-Verfahren.
5. Datenbankgestütztes Abfragen und Auswerten der Ergebnisse.
6. Implementierung eines RESTful Webservice zur Nutzung und Visualisierung der Parlamentsreden für das Frontent.
7. Implementierung einer komplexen Visualisierungskomponente unter Verwendung von **d3.js**[3] gemäß Aufgabe 3.1.
8. Administration der Parlamentsdebatten unter Verwendung eines benutzer- und gruppengesützten Zugriffsmanagmentsystems (vgl. 3.2)
9. Visualisieren sowie Exportieren in ein LaTeX-basiertes Parlamentsprotokoll (vgl. 3.3).

Das Backend soll als RESTful-Webservice unter Verwendung von Java Spark implementiert werden. Dieses Backend stellt dem Frontend alle Methoden und Funktionen zur Verfügung, um mit den Parlamentsprotokollen gemäß der Aufgabenbeschreibung zu interagieren. Zum Testen und Dokumentieren der REST-Abfragen verwenden Sie `swagger.io`. Das Backend ist nach objektorientierten Standards in Java zu erstellen, wie in den vorherigen Übungen erprobt. Die Verwendung von Interfaces bzw. abstrakten Klassen ist obligatorisch und zwingend erforderlich. Sorgen Sie in allen Komponenten für eine möglichst hohe Datenkapselung und vermeiden Sie wiederkehrenden bzw.

[3] https://d3js.org/

duplizierten Quellcode.

Betrachten Sie die Parlamentsdebatten basierend auf den Reden; modellieren Sie die Datenstrukturen dazu so, dass die Reden die zentralen Bestandteile sind: Jede Rede gehört zu einer Sitzung; jede Rede hat Kommentare; jede Rede hat einen Redner, etc. Für Redner ist eine eigene Datenstruktur zu definieren, die es erlaubt, komplexere Inhalte abzubilden. Speichern Sie die Parlaments-Daten in der Ihnen zur Verfügung stehenden MongoDB ab und analysieren Sie die Daten mit NLP-Methoden zwecks der zu realisierenden Visualisierungen samt der benötigten Analysen. Visualisieren Sie Ihre Ergebnisse mittels **d3.js** gemäß der in Aufgabe 3.1 definierten Anforderungen. Erweitern Sie Ihre Applikation um ein Benutzer-, Gruppen- und Feature-Management, das es ermöglicht, die Parlamentsdebatten zu betrachten, zu analysieren bzw. zu editieren und zu exportieren. Beachten Sie auch das bereitgestellte Use-Case-Diagramm, das die Funktionalitäten des Programms beschreibt. Im Zuge des Abschlussprojekts werden Sie dieses Diagramm ggf. erweitern müssen.

Fragen...

...zur Umsetzung stellen Sie bitte frühzeitig, damit diese schon in Ihre Projektplanung einfließen können.

Hierfür findet am **Mittwoch, 11.01.2022, 10.00 Uhr**, eine allgemeine Frage-Stunde via Zoom statt.

Achten Sie auf ein geeignetes Error-Handling und ein aussagekräftiges Feedback für den Benutzer bei Fehleingaben. Dokumentieren Sie Ihr Projekt sowie die Projektbearbeitung ausführlich (Aufgabe 4) und präsentieren Sie Ihr Ergebnis als Gruppe im Tutorium.

Aufgaben

Zur Durchführung lesen Sie zunächst alle Aufgaben gründlich durch, damit Sie Zusammenhänge und Abhängigkeiten nicht übersehen.

1 Projektplanung und Design — 10 Punkte

Erstellen Sie alle nachfolgenden Diagramme unbedingt unter Verwendung von geeigneten Softwarelösungen (z.B. LaTeX)!

(a) Definieren Sie einen Zeitplan für ihr Projekt, der auch aufzeigt, wer in Ihrer Gruppe welche Aufgaben übernimmt bzw. übernommen hat. Dokumentieren Sie dies anhand eines *Gantt*-Diagramms mit Milestones und Aufgabenpaketen. Das Projektende ist der **17.02.2023**. Die Milestones sind auch im GitLab-Projekt zu hinterlegen und zu pflegen. Der Projektverlauf ist einzuhalten und evtl. Abweichungen sind zu dokumentieren. **Eine kontinuierliche Nutzung von GitLab ist daher obligatorisch.**

(b) Erstellen Sie ein Package- und Klassendiagramm für Ihr Backend und *erweitern* Sie das gegebene Use-Case-Diagram.

(c) Erstellen Sie ein Mock-up für Ihre Visualisierung bzw. das User-Interface.

2 Backend — 12 Punkte

Entwickeln Sie das Backend ihrer Applikation als RESTful Java-Applikation[4] und implementieren Sie die nachstehenden Anforderungen zur Abbildung der Datenstrukturen. Dies ist ggf. durch die Anforderungen in Aufgabe 3.1 – 3.3 zu ergänzen. Ihr Backend ist das Rückgrat ihrer gesamten Applikation und beinhaltet alle Methoden und Funktionen für die Aufbereitung, Analyse, Verwaltung und Formatierung der Daten.

(a) Implementieren Sie eine Datenstruktur basierend auf der in Aufgabe 1 definierten Diagramme. Verwenden Sie hierzu objektorientierte Konzepte aus der Programmiersprache Java (Interfaces / abstrakte Klassen, sowie deren Implementierungen).

(b) Implementieren Sie eine parametergestützte Klasse `MongoDBHandler`[5], die die Kommunikation mittels der MongoDB gewährleistet. Dies beinhaltet die Operationen *Erstellen*, *Lesen*, *Updaten*, *Löschen*, *Aggregieren* und *Zählen*.

- Die Nutzung von Hilfsbibliotheken, welche die Datenstrukturen selbstständig in MongoDB abbilden, ist nicht gestattet.

(c) Laden Sie die Parlamentstexte (Wahlperiode 19 + 20) direkt vom Bundestag[6] herunter (via Java) und importieren Sie die Protokolle in ihre Datenbank. Verwenden Sie die Protokolle ab Beginn der 19. Legislaturperiode (einschließlich) bis *heute*[7]. Eine Routine prüfe vorab, ob die Debatten schon vorhanden sind.

- Das manuelle Nachladen der Protokolle soll über das Frontend möglich sein. Vor dem Verarbeiten ist zu prüfen, ob das heruntergeladene Protokoll bereits existiert.
- Der Fortschritt der einzelnen Verarbeitungen ist als Fortschrittsanzeige im Frontend zu visualisieren.

(d) Bilden Sie die Abgeordneten der 19. sowie der aktuellen Wahlperiode vollständig ab und erweitern Sie die Abfragen über die Parlamentsreden hinaus:

1. Alle Abgeordneten inkl. ihrer Meta-Daten sind in einer eigenen Collection abzulegen.
2. Fügen Sie, falls vorhanden, ein Foto der Abgeordneten ein und verwenden Sie hierfür die Bilddatenbank des Bundestages[8]. Die Abfragen sind in Java zu implementieren und während des Parsens zu erfassen. Bitte beachten Sie, dass jedes Bild eine Menge von Meta-Daten hat, die ebenfalls abzubilden sind.

(e) **Natural Language Processing**

(1) Implementieren Sie Klassen und Methoden zur objektorientierten Vorverarbeitung von Parlamentsreden.

(2) Implementieren Sie Ihre Applikation so, dass alle eingelesenen Debatten, unmittelbar nachdem diese abgerufen wurden, mittels NLP-Methoden unter Verwendung einer UIMA-Pipeline-Architektur vorverarbeitet werden. Verwenden Sie die nachstehenden Vorverarbeitungsmethoden[9]:

- Token, Sentences, POS, Dependency, Named Entity Recognition (**SpaCyMultiTagger3**)
- DDC (**LabelAnnotatorDocker**)
- Sentiment (**GerVaderSentiment**)

[4] Die Verwendung von Maven ist verpflichtend!
[5] Die Zugangsdaten bekommen die Gruppen in ihre GitLab-Projekte kopiert.
[6] https://www.bundestag.de/services/opendata
[7] Heute bedeutet immer den jeweils aktuellen Tag, also maximal das Projektende am 17.02.2023.
[8] https://bilddatenbank.bundestag.de
[9] Die Adressen für die Vorverarbeitung finden Sie im OLAT-Kurs.

(3) Speichern Sie die Ergebnisse in geeigneter Form zur späteren Abfrage in Ihrer MongoDB-Datenbank und serialisieren Sie das gesamte Ergebnis der Analyse.

(f) RESTful Webservice

(1) Implementieren Sie eine RESTful-Schnittstelle basierend auf Java Spark[10] und definieren Sie *Routen* und *Parameter* gemäß den Aufgaben 3.1 bis 3.3.

(2) Dokumentieren Sie die REST-Methoden und machen diese unter Zuhilfenahme von *swagger.io*[11] verfügbar.

Bonus

Implementieren Sie basierend auf einem Websocket ein Chat-System innerhalb der Applikation, und zwar in Zusammenhang mit Aufgabe 3.2 – siehe unten.

3 Frontend

Implementieren Sie das gesamte Frontend unter Verwendung der *FreeMarker*-Template-Engine. Innerhalb der Templates können Sie Frameworks[12] Ihrer Wahl verwenden. Über folgende Komponenten soll Ihr Frontend verfügen:

3.1 Visualisierung und Abfrage von Parlamentsdebatten — 14 Punkte

(a) Implementieren Sie die nachfolgenden Visualisierungen unter Verwendung der Visualisierungsbibliothek d3.js:

(1) Verteilung der Token als Line-Chart.

(2) Verteilung der POS als vertikale Bar-Chart.

(3) Verteilung der Sentiments als Radar-Chart.

(4) Verteilung der *Named Entities* als multiple Line-Chart sortiert nach Named Entity-Typ:

- genannte Personen,
- genannte Orte,
- genannte Organisationen.

(5) Verteilung der Redner als Bar-Chart mit Redner-Bild als „Hover-Effekt“.

(6) Visualisieren Sie die Ergebnisse der Abstimmungen. Bei namentlichen Abstimmungen, visualisieren Sie auch die Namen der Abgeordneten.

(b) Alle Visualisierungen sind zeitbezogen zu parametrisieren:

- Beginn- und End-Zeitraum (frei auswählbar);
- gruppiert nach Zeitabschnitten (Woche, Monat)
- wählen Sie hierzu ein geeignetes Mittel (z. B. Slider, Combo-Box, Kalender-Feld).

(c) Implementieren Sie eine generische Suchfunktion, welche sowohl über die Tastatur als auch über einen „Such-Button“ ausgelöst werden kann.

(1) Jede Suchanfrage erzeugt ein neues internes „Panel“, das als Titel die gesuchte Zeichenkette enthält. Das Panel selbst beinhaltet dann als Ergebnis die zuvor genannten Visua-

[10] https://sparkjava.com/
[11] https://swagger.io/
[12] jQuery, etc.

lisierungen, mit den in der Suchabfrage enthaltenen Inhalten.

(d) Implementieren Sie eine Volltext-Visualisierung für eine Parlaments-Rede, die folgende Informationen abbildet:

1. Alle im Text gefundenen Named Entities werden farblich hinterlegt und eine Legende kennzeichnet den Typ der Annotation (also Person, Ort, Organisation).
2. Für das Ende eines jeden Satzes wird über ein Informations-Icon der Sentiment-Wert des Satzes angezeigt.
3. Der Sprecher*innenname, Informationen zu seiner/ihrer Partei- und Fraktionszugehörigkeit sowie ein Foto (sofern vorhanden) werden im Kopf der Rede angezeigt.
4. Ermöglichen Sie eine Reden-Auswahl
   - über eine Volltext-Suche.
   - über eine Navigation durch die Tagesordnung der Protokolle (Protokoll, Tagesordnung, Reden).
5. Visualisieren Sie die Kommentare an den dazugehörigen Stellen im Text und heben Sie die kommentierende(n) Person(en) (Abgeordnet*inn) sowie ihre Fraktion(en) hervor.

(e) Visualisieren Sie Netzwerke: Implementieren Sie hierzu basierend auf verschiedenen Kontexten (Parlamentsprotokoll, Rede, Wahlperiode) die nachfolgenden Netzwerke und achten Sie auch auf angemessene Interaktionsmöglichkeiten:

(1) Kommentar-Netzwerk: Welche Redner*innen wurden von welchen Abgeordneten*innen kommentiert? Visualisieren Sie das Sentiment als Farbe der Kante (rot: negativ, grau: neutral, grün: positiv).

(2) Rede-Netzwerk: Welche Redner*innen sprechen zu den gleichen Themen? (Topic-Analyse)

(3) Sentiment-Rede-Netzwerk: Verbindung zwischen Reden, Sentiment und Kategorien (Topics).

Bonus

Fallen Ihnen noch weitere sinnvolle Visualisierungen und Filterungsmöglichkeiten ein? Implementieren Sie diese und erhalten Sie dafür Bonus-Punkte.

3.2 Administration von Parlamentsdebatten — 10 Punkte

Daten sind nicht perfekt und auch die Parlamentsprotokolle des deutschen Bundestages sind hier keine Ausnahme. Implementieren Sie innerhalb des Frontends eine Editoren-Umgebung, die die nachstehenden Funktionen abbildet. Dies erweitert auch die Funktionen aus Aufgabe 2. Bitte beachten Sie, dass Sie keine verschiedenen *Seiten* erstellen, sondern die einzelnen Funktionen nur nach erfolgreichem Login, sowie vorhandener Features / Berechtigungen, aktiv sind[13].

(a) Erstellen Sie ein Login, das nach erfolgreicher Anmeldung die Bearbeitung sowie Neuanlage von Parlamentsprotokollen ermöglicht. Dies beinhaltet u.a. die folgenden Funktionen:

- Anlegen / Bearbeiten von Protokollen und Reden
- Zuordnung von Rednern zu Reden
- Zuordnung von Rednern und Fraktionen zu Kommentaren

(b) Implementieren Sie ein Benutzer-, Gruppen- und Rechte-Management, das basierend auf `Features` die folgenden Funktionen ermöglicht:

[13] Die Nutzung von FreeMarker emphiehlt sich sehr!

- Benutzer anlegen / editieren und löschen.
- Vergeben von Rechten, Administration von `Features`.
- Protokolle, Reden, Kommentare anlegen / ändern / löschen.
- Abgeordnete anlegen oder editieren.

(c) Export-Templates können verwaltet, editiert und erstellt werden (vgl. Aufgabe 3.3).

3.3 Export und Präsentation der Parlamentsdebatten — 12 Punkte

Die Präsentation sowie der Export von Parlamentsprotokollen ist ein wichtiger Bestandteil Ihrer Applikation. Sie ermöglichen damit unter anderem, die bestehenden Protokolle informativ und ansprechend auszugeben bzw. zu exportieren. Erstellen Sie hierzu die nachfolgenden Funktionen:

(a) Implementieren Sie einen Protokoll-Export, um aus Ihren Parlamentsprotokollen ein LaTeX-Dokument zu generieren, welches die folgenden Bestandteile umfasst:

- Tagesordnung als Inhaltsverzeichnis.
- Die einzelnen Reden...
  - ... besitzen Angaben, inkl. Bild. des Redners.
  - ... besitzen statistische Angaben der relevanten NLP-Informationen
  - ... visualisieren Kommentare und Zurufe an den entsprechenden Stellen im Text. Sofern ein konkreter Abgeordneter auszumachen ist, wird dessen Bild inkl. Namen neben dem Zuruf visualisiert.
  - Das Ergebnis einzelner Abstimmungen (sofern vorhanden) wird zusätzlich mittels grafischer Elemente (u.a. Tortendiagramm) unterstützt.
  - Die Generierung kann für spezifische Protokolle sowie für alle Protokolle auf einmal erfolgen. Erfolg die Generierung für mehr als ein Protokoll ist jedes Protokoll, gemäß Sitzungs-Index und Datum, durch ein *Kapitel* zu trennen.
  - **Überlegen Sie**, welche Elemente noch für eine adäquate Präsentation nötig sind und fügen Sie diese ein.

(b) Implementieren Sie den Export, wie gewohnt objektorientiert, indem die einzelnen LaTeX-Bausteine mittels einer Methode *toTeX()* über Ihre Klassenstrukturen abgebildet und generiert werden können.

(c) Die einzelnen Bausteine sind via Webinterface zu erstellen und zu verwalten (je nach Berechtigung).

(d) Ihre Webapplikation soll eine Live-Vorschau des Exports implementieren[14]. Die Generierung erfolgt hierbei über das Backend und das Frontend visualisiert das generierte PDF innerhalb des Frontends, ohne vorherigen Download.

(e) Bitte sorgen Sie für ein aussagekräftiges Fehlermanagement.

4 Dokumentation — 10 Punkte

Dokumentieren Sie ihren Quellcode und den Projektverlauf ausführlich und erstellen Sie als Gruppe eine Dokumentation:

(a) Dokumentieren Sie den Quellcode ausführlich unter Verwendung von JavaDoc.

[14] vgl. Overleaf

(1) Dokumentieren Sie, welche Gruppenmitglieder welche Methoden implementiert haben.

(2) Dokumentieren Sie, welche Gruppenmitglieder welche Methoden modifiziert haben.

(3) Beschreiben Sie jede Methode ausführlich.

(4) Verwenden Sie sprechende Variablennamen.

(5) Verwenden Sie `swagger.io` zur Dokumentation der REST-Methoden.

(b) Erstellen Sie ein Benutzer-Handbuch, in dem Sie die Nutzung ihrer Applikation ausführlich erläutern.

Abgabe der Aufgabe

Ihr vollständiges, fertiges Projekt[15] laden Sie bitte bis spätestens zum **17.02.2023** als ZIP-Archiv in OLAT hoch und **pushen** alles zusätzlich in ihr GitLab-Projekt. Laden Sie das Projekt nur „einmal" pro Gruppe in OLAT hoch! Alle Abgaben, die nach diesem Datum hochgeladen werden, werden nicht berücksichtigt.

Die Präsentation der Applikation in den Abschlusstutorien wird von den Tutor*innen festgelegt.

[15] Beinhaltet: Projektquellcode, Projektdokumentation (Quelle + PDF), Benutzerhandbuch.