
GRAPHHAUSWERTUNG

ERMITTLUNG DER STIMMUNGSMACHER IM BUNDESTAG
MARKUS CHRISTOPHER GLUTTING, MARIE BITTIEHN, MIRIAM LISCHKE

18. Februar 2021



Deutscher Bundestag

Betreut von Prof. Dr. Thomas Hoppe
Informationssysteme
M.Sc. Angewandte Informatik
Hochschule für Technik und Wirtschaft
Treskowallee 8, 10318 Berlin, Deutschland

Inhaltsverzeichnis

| | |
|--|----------|
| Abbildungsverzeichnis | 1 |
| Tabellenverzeichnis | 1 |
| 1 Einleitung | 2 |
| 1.1 Hintergrund | 2 |
| 1.2 Problemstellung | 2 |
| 1.3 Zielsetzung | 2 |
| 2 Grundlagen | 3 |
| 2.1 PageRank | 3 |
| 2.2 Entwicklungsframeworks, -Tools und -Konzepte | 4 |
| 3 Anforderungsanalyse und Konzept | 6 |
| 3.1 Architektur | 6 |
| 3.2 Schnittstellen | 6 |
| 4 Implementierung | 7 |
| 5 Fazit | 8 |
| 5.1 Zusammenfassung | 8 |
| 5.2 Lernziele | 8 |
| 5.3 Ausblick | 8 |
| Literaturverzeichnis | i |
| Glossar | ii |

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Der folgende Teilabschnitt der Ausarbeitung beschäftigt sich mit dem sechsten Teilprojekt: die Graphauswertung. Die Projektgruppe, welche an dem Teilprojekt gearbeitet hat, besteht aus Markus Glutting, Miriam Lischke und Marie Bittiehn.

1.1 Hintergrund

Das Teilprojekt “Graphauswertung” baut auf den Ergebnissen der Teilprojekte “Interaktion zwischen Personen” (Gruppe 4) und “Interaktion zwischen Fraktionen” (Gruppe 5) auf. Genauer formuliert, besteht die Aufgabenstellung darin, die Graphen, welche von Gruppe 4 und 5 erstellt werden, auszuwerten und die Ergebnisse der Auswertung der nachfolgenden Gruppe 8 für ihre Benutzeroberfläche zur Verfügung zu stellen.

1.2 Problemstellung

Durch die Auswertung der Graphen sollen die sogenannten “Stimmungsmacher” im Bundestag ermittelt werden. Unter einem Stimmungsmacher ist im vorliegenden Kontext eine Person gemeint, welche viel mit vielen verschiedenen Personen redet und somit eine Stimmung verbreitet. Ob diese verbreitete Stimmung positiv oder negativ ist, ist dabei nicht entscheidend.

Neben der Ermittlung von Stimmungsmachern sollen ebenfalls simple mathematische Analysen auf den Graphen durchgeführt werden. Dadurch soll eine Gesamtbetrachtung der Sitzungen einer Wahlperiode ermöglicht werden, ebenso wie die Option die Sitzungen in Vergleich zueinander stellen zu können.

1.3 Zielsetzung

Um Stimmungsmacher zu ermitteln, soll der PageRank-Algorithmus (siehe Kapitel 2) verwendet werden. Dies bietet sich an, da Stimmungsmacher gleichbedeutend sind zu Personen, welche viele Nachrichten mit einem positiven und/oder einem negativen Sentiments empfangen bzw. versenden. Mithilfe des PageRank-Algorithmus werden eben diesen Personen bzw. Knoten im Graphen hohe Ränge vergeben, wodurch sie identifiziert werden können.

Für die mathematischen Analysen sollen Berechnungen auf den gesamten Graphen durchgeführt und statische Größen wie bspw. der Median oder ein Quartil der Sentiments berechnet werden.

TODO: Lernziele darstellen

2 Grundlagen

In diesem Kapitel werden die notwendigen Kenntnisse für das Verständnis der nachfolgenden Kapitel vermittelt. Zuerst wird der PageRank-Algorithmus erläutert mit Fokus auf eine effizientere Berechnung der Ränge mithilfe von Matrizenmultiplikationen. Nachfolgend wird auf die wichtigsten verwendeten Entwicklungsframeworks, -Tools und -Konzepte eingegangen.

2.1 PageRank

Der PageRank-Algorithmus ist ein Algorithmus zur Gewichtung von Knoten innerhalb eines Netzwerks anhand der Anzahl ihrer eingehenden Beziehungen. Dabei geht in die Berechnung des PageRanks eines Knoten, neben der Anzahl an eingehenden Beziehungen, ebenso der PageRank der auf ihn verweisenden Knoten mit ein.

Aufgrund der daraus entstehenden direkten Abhängigkeit des PageRank eines Knoten von den PageRanks der auf ihn verweisenden Knoten, ist vor allem bei großen Netzwerken eine genaue Berechnung aller PageRanks nicht immer in absehbarer Zeit möglich. Aus diesem Grund werden die Werte für die PageRanks bei der Berechnung meist iterativ angenähert. Dabei wird allen Knoten ein einheitlicher Startwert als PageRank vergeben. Meist wird als Startwert der Wert $1/N$ verwendet, wobei N die Anzahl aller Knoten des Netzwerks ist. Danach wird der PageRank für alle Knoten mehrfach berechnet und so die Werte iterativ angenähert.

Eine Möglichkeit dieser iterativen Annäherung ist ein mehrfaches Iterieren über das gesamte Netzwerk und die rekursive Berechnung des PageRanks für jeden Knoten. Da im vorliegenden Anwendungskontext mit mehreren hundert Knoten und zig Tausend von Beziehungen zwischen den Knoten gerechnet werden kann, wäre das Iterieren über alle Knoten und Beziehungen nicht sehr effizient. Stattdessen können allerdings Matrizenmultiplikationen und eine Eigenvektorberechnung für die PageRank-Berechnung verwendet werden.

Das Netzwerk, welches aus mathematischer Sicht als gerichteter Graph betrachtet werden kann, kann als eine quadratische, stochastische Matrix abgebildet werden [1]. Stochastisch bedeutet in diesem Kontext, dass die Summen der Spalten der Matrix alle 1 betragen [1]. Wenn im Graphen von einem Knoten j eine Beziehung zum Knoten i besteht, dann wird in der Matrix an der Stelle ij der Wert $1/d^+(j)$ eingetragen, wobei $d^+(j)$ für den Ausgangsgrad von j steht [1]. Wenn keine Beziehung besteht, wird an der Stelle ij eine 0 eingetragen [1]. Für den Fall, dass ein Knoten gar keine ausgehenden Beziehungen besitzt, wird in seiner gesamten Spalte in der Matrix

der Wert $1/N$ eingetragen [1]. Dadurch wird verhindert, dass der PageRank in Sackgassen-Knoten gewissermaßen “versickert”.

Wurde der Graph nun als Matrix abgebildet, so kann der PageRank iterativ mithilfe von Matrixmultiplikationen berechnet werden. Dabei wird der PageRank als Vektor der Länge N dargestellt, meist befüllt mit dem Startwert $1/N$ für alle Knoten. Statt einer gesamten Iteration über das Netzwerk und den rekursiven Berechnungen neuer PageRanks für alle Knoten, muss nun nur eine simple Multiplikation der Matrix mit dem Vektor durchgeführt werden. Das Ergebnis der Multiplikation ist wieder ein Vektor und dieser enthält die neuen PageRanks aller Knoten. Nach mehrfacher Multiplikationen der Matrix mit dem PageRank-Vektoren ändern sich die Werte innerhalb des Vektor nicht mehr. Es wurde iterativ ein dominanter Eigenvektor der Matrix berechnet, welcher gleichzeitig den endgültigen PageRanks der Knoten des Netzwerks entspricht [1].

Auf gleiche Art und Weise kann der sogenannte “Reverse PageRank” berechnet werden. Bei dem Reverse PageRank handelt es sich um den PageRank berechnet auf dem Reverse-Graphen des Originalgraphen [2]. Dies bedeutet, dass der Reverse PageRank nicht basierend auf den eingehenden Beziehungen der Knoten berechnet wird, sondern auf den ausgehenden Beziehungen.

Um im ersten Schritt den Reverse-Graphen des Originalgraphen zu erhalten, werden die Richtungen der Beziehungen zwischen den Knoten des Originalgraphen umgedreht [2]. Danach wird der so erzeugte Reverse-Graph ebenfalls als Matrix abgebildet und die PageRank-Werte der Knoten werden mithilfe von Matrizenmultiplikationen berechnet. Die dadurch ermittelten Werte für den Reverse PageRank ähneln den Ergebnissen des klassischen PageRank dahingehend, dass die einzelnen Werte gleich sind, aber anders verteilt. Dies ist dadurch zu erklären, dass bei der Berechnung des Reverse PageRank zwar die Richtungen der Beziehungen umgedreht werden, jedoch keine Beziehungen entfernt oder hinzugefügt werden. Aus diesem Grund sind die Werte gleich, aber durch das Umdrehen der Beziehungen sind sie anders verteilt.

2.2 Entwicklungsframeworks, -Tools und -Konzepte

Vor der Wahl eines Entwicklungsframeworks galt es zunächst zu entscheiden, welche Programmiersprache verwendet. Innerhalb der Gruppe war die Python-Bibliothek numpy bekannt, welche gut zur performanten Berechnung von Matrizen geeignet ist. Daher fiel die Entscheidung auf die Programmiersprache Python. Zur Bereitstellung von HTTP-Schnittstellen für die Benutzeroberfläche wurde das Microframework Flask verwendet. Flask zeichnet sich dadurch aus, dass es einen sehr schlanken Kern hat, der nach Bedarf erweitert

werden kann. Dadurch unterscheidet es sich stark von anderen Frameworks wie bspw. Django, welches stattdessen eher darauf abzielt, ein vollständiges Framework für Webanwendungen zu sein.

Der Zugriff auf die Neo4j Datenbanken erfolgt mithilfe des Neo4j Python Treibers, welcher im Python Paket neo4j bereitgestellt wird. Es wurde kein Object Graph Mapper (z.B. neomodel) verwendet, da eines der Lernziele darin bestand, die Abfragesprache Cypher kennenzulernen.

3 Anforderungsanalyse und Konzept

Die allgemeinen Anforderungen ergeben sich aus der Aufgabenstellung:

- Ermittlung von Stimmungsmachern im Bundestag
- Mathematische Analysen bezüglich des Sentiments im Bundestag

Zur Identifikation detaillierter Anforderungen fand im frühen Projektverlauf eine Absprache mit Gruppe 8 statt, welche die Nutzeroberfläche entwickelt. Im Rahmen dieses Meetings wurde festgelegt, dass folgende Funktionalitäten notwendig sind:

- Abfrage von Stammdaten:
 - Sitzungen
 - Fraktionen
 - Abgeordnete
- Abfrage des Graphen für Fraktionen und Abgeordnete. Die Übertragung des Graphen erfolgt durch gleichzeitige Übertragung der Beteiligten Knoten (Abgeordnete bzw. Fraktionen) und der Nachrichten, die zwischen diesen verschickt wurden. Dabei werden Nachrichten zwischen zwei beteiligten Knoten aggregiert, um die übertragene Datenmenge zu reduzieren.
- Mathematische Analysen bezüglich des Sentiments im Bundestag. Als Ziel wurde die Darstellung von Boxplots ausgegeben. Daher wurde festgelegt, dass Minimum, unteres Quartil, Median, oberes Quartil und Maximum berechnet werden. Zum Vergleich des Sentiments innerhalb verschiedener Sitzungen soll die Möglichkeit der Filterung nach Sitzung vorgesehen werden.

3.1 Architektur

TODO: Diagramm + Beschreibung

3.2 Schnittstellen

- 1) HTTP Endpoints für Nutzeroberfläche
- 2) Datenbankzugriff auf Neo4j Datenbanken

4 Implementierung

5 Fazit

5.1 Zusammenfassung

5.2 Lernziele

5.3 Ausblick

Literaturverzeichnis

- [1] Taher Haveliwala. *Efficient Computation of PageRank*. Technical Report 1999-31. Stanford InfoLab, 1999. URL: <http://ilpubs.stanford.edu:8090/386/> (besucht am 09.02.2021).
- [2] Li-Tal Mashiach und Ziv Bar-Yossef. „Local approximation of pagerank and reverse pagerank“. In: *Proceedings of the 17th ACM conference on Information and knowledge management*. Okt. 2008, S. 279–288. URL: <http://www.cs.technion.ac.il/users/wwwb/cgi-bin/tr-info.cgi/2008/MSR/MSR-2008-09> (besucht am 18.02.2021).

