FernUniversität in Hagen

Documentation Grundpraktikum Programmierung

AnguillaSearch

Verfasser:

Willi Ehrmann

Matrikelnummer:

q9342591

Kurs:

63081-Grundpraktikum Programmierung WS 2024/2025

Abgabedatum:

6. Januar 2025

6. Januar 2025

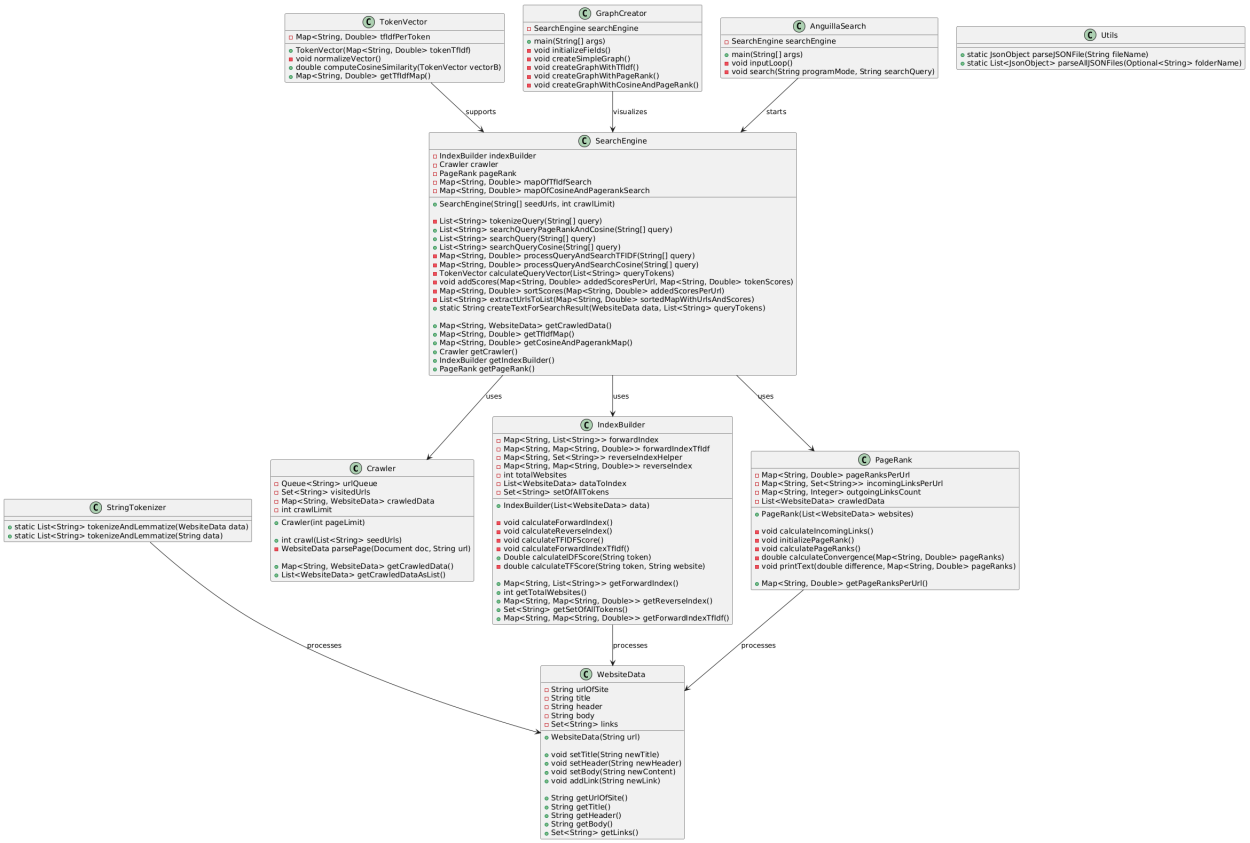
Inhaltsverzeichnis

1 Einleitung

Hier beschreibe ich den Grundlegenden Aufbau und Funktion eines Java Programms welches im Zuge des Grundpraktikums Programmierung erstellt wurde. Das Programm dient dazu Webseiten zu crawlen und die Inhalte zu Indexieren so dass der benutzer danach über die Konsole eine Such ausführen kann und Links zu passenden Webseiten angezeigt bekommt. Hierbei werden 3 verschiedene Ansätze zum Vergleich der Suche mit den Webseiten verfolgt. Einmal werden sogenannte TFIDF Scores für jeden Token jeder Webseite erstellt. Die Summe dieser Token passend zur Query wird addiert und dann werden die Links absteigend sortiert nach Höhe des Gesamtscores ausgegeben. Der zweite Ansatz verfolgt den Vergleich der Kosinus- Ähnlichkeit von Suchvektor und Tokenvektor der Webseiten. Auch hier werden die Links absteigend sortiert und anschließend ausgegeben. Der Dritte Ansatz sortiert die Webseiten im Hinblick auf einer Kombination aus Kosinusähnlichkeit vom Suchvektor und Tokenvektor der Webseiten und Pagerank der jeweiligen Webseiten.

Außerdem hat das Programm noch eine ausführbare Utilityklasse die eine Suche auf einem Webseitennetz bildlich darstellt. Die Knoten sind dabei die Webseiten und die Kanten die Links der Webseiten untereinander. Hier werden 4 verschiedene Bilder erstellt. Einmal ein simples Bild aller gecrawltern Webseiten und der Links untereeinander. Zweitens ein Bild welches die TFIDF Scores passend zur Suche an die Labels der Knoten schreibt. Das dritte Bild zeigt die Knoten in variabler Größe je nach ihrem Pagerank. Außerdem zeigen die Kanten anhand eines Labels wieviel von ihrem PagerankScore sie an die jeweils andere Webseite übertragen. Das vierte Bild Gibt die Knoten mit einer Beschriftung der kombinierten Kosinusähnlichkeitsscores mit Pagerank aus wobei die 3 Knoten mit dem höchsten Score farblich hervorgehoben werden.

Im Folgenden sehen sie ein Klassendiagramm des Programms:



2 Klassen und Pakete

2.1 paket crawler

Die Klasse `Crawler` ist dafür verantwortlich, eine bestimmte Anzahl vorgegebener *Seed* urls) zu besuchen und deren Inhalte auszulesen und in `WebsiteData`-Objekten zu speichern.

Die Klasse `WebsiteData` enthält die Informationen zu einer einzelnen Webseite. Sie enthält den Titel, Header, Body und die Links der Webseite.

2.2 paket indexing

Die Klasse `IndexBuilder` erstellt verschiedene Indizes basierend auf den `WebsiteData` Objekten welche sie vom Crawler übergeben bekommt. Sie berechnet außerdem die TFIDF scores für die Token.

1. **Forward Index**: Weist jeder url eine Liste aller vorkommenden Tokens zu. (nach

Tokenisierung,Lemmatisierung und Entfernung aller Stopwörter und Würter unter Länge 4).

2. **Reverse Index** (Helper & TFIDF Mapping): Für jedes Token wird im Helper gespeichert, in welchen Webistes (urls) es auftaucht Danach wird im Reverse Index noch der jeweilige TFIDF Score auf jede Url gemappt. Token -> (url -> TFIDF)

3. **TF-IDF-Berechnung**: Berechnet für jedes Token in jedem Dokument den TF (Term Frequency) und IDF (Inverse Document Frequency) Score. Daraus entsteht der eigentliche TFIDF Score (TF *IDF)

4. **Forward Index mit TFIDF**: Für jede url werden dort vorkommenden Token auf den jeweiligen TFIDF Score gemapped. url -> (Token -> TFIDF)

2.3 paket searching

Die Klasse `SearchEngine` benutzt die gesammelten Daten (Listen von `WebsiteData`) und benutzt verschiendene Suchverfahren um mit hilfe einer Suchanfrage die relevantesten Ergebnisse zu finden. Es gibt 3 verschiedene Sucharten:

- **TFIDR Suche**: Gewichtung der Relevanz der Suchbegriffe innerhalb jeder Webseite.

- **PageRank**: Berechnung des PageRankwerts für jede url basierend auf deren eingehenden und ausgehenden Links.

- **Kosinusähnichkeit + PageRank**: Kombination, um die relevantesten Seiten hinsichtlich der Suchbegriffe zu finden.

Die Klasse `PageRank` berechnet den PageRankwert zu jeder url anhand der Linkstruktur. Dabei wirdder Algorithmus iterativ angewendet.Solange zwischen zwei Iterationen die Gesamtabweichung der Werte zwischen den Iterationen noch nicht konvergiert ist, werden die PageRank-Scores weiter berechnet.

Die Klasse `TokenVektor` erstellt einen Vektor (Eine Map von Token -> TFIDF Score) von entweder einer url (urlvektor) oder einer Suchanfrage (queryvektor). Dieser wird dann

u.a. dazu benutzt um die Kosinusähnlichkeit zu berechnen. die Vektoren werden normiert so dass für die Kosinusähnlichkeit nur noch das Skalarprodukt gebraucht wird.

2.4 paket util

Die Klasse `StringTokenizer` zerlegt die gecrawlten Inhalte der Websites Titel, Header und Body in einzelne Wörter (Tokens) Und lemmatisiert diese auf ihre Grundform. Anschließend werden Stoppwörter und zu kurze Wörter entfernt außerdem alle Links (Wörter die Zahlen und einen Punkt irgendwo enthalten). Diese Klasse verwendet eine `StanfordCoreNLP` Pipeline zur Tokenisierung und LEmmatisierung.

Die Klasse `Utils` besitzt Hilfsfunktionen um json Dateien einzulesen und zu parsen und bestimmte Arrays daraus zu extrahieren (z.B: seed urls, query urls etc.).

Die Klasse `GraphCreator` ist eine Utilityklasse mit einer Mainmethode. Beim Ausführen erstellt diese aus einer vordefinierten geparsten Liste an seedursl eine Searchengine (mit Crawler und Indexer), erstellt mittels der Bibliothek `JGraphX` verschiedene Graphen welche als Bild in einem Ordner (figures) abgelegt werden. Es werden 4 verschiedene Graphen erstellt mithilfe folgender Methoden:

- `createSimpleGraph()`: Erstellt einen einfachen Graphen, in dem jede gecrawlte url als Knoten dargestellt wird und Kanten die Links zwischen den Seiten abbilden.
- `createGraphWithTfIdf()`: Führt eine Suchanfrage ("flavor") aus und nutzt die aus der `SearchEngine` bereitgestellten TFIDF Werte. Diese Werte werden auf die Knoten beschriftet.
- `createGraphWithPageRank()`: Nutzt den berechneten PageRank jeder url, um die Knotengrößen zu skalieren und die Flussmenge von jedem Knoten zu jedem verlinkten Knoten an die Kante zu schreiben.
- `createGraphWithCosineAndPageRank()`: Kombiniert Kosinusähnlichkeit mit Pagerank, schreibt die Werte in die Knoten und hebt die Top3 der Knoten mit den höchsten Werten farblich hervor.

Zu diesem Zweck nutzt `GraphCreator`:

- `mxGraph` und `mxCircleLayout` aus dem `JGraphX`-Framework
- `mxCellRenderer` zum Erzeugen von `BufferedImage`-Objekten
- `ImageIO` zum Schreiben in `.png`-Dateien

2.5 Hauptklasse

Die Klasse `AnguillaSearch` ist die Hauptklasse mit der main-Methode und dient zum Starten des Programms. Sie Lädt und parst eine json Datei mit seed urls und startet eine `SearchEngine` welche ihrerseits erst einen `Crawler` erstellt die Urls inklusive verlinkter Urls bis zum festgesetzten limit crawlt und danach an einen `IndexBuilder` übergibt welcher die Indizes erstellt. Nachdem die `SearchEngine` komplett initialisiert ist inklusiver aller Indizes wird die Konsole freigegeben und der Benutzer kann mit einer Suchanfrage beginnen oder das Programm beenden. Vor der Suchanfrage muss der Benutzer noch auswählen welchen der 3 Suchmodi er benutzen will (TFIDF, Kosinusähnlichkeit oder Kosinusähnlichkeit in Kombination mit PageRank). Nach einer Sche wird auf der Konsole dann eine nach Relevanz absteigend sortierte Liste an Urls ausgegeben mit einem kurzen Textsnippet aus dem Body der Seite.

3 Datenstrukturen

Eine der Hauptdatenstrukturen ist die `WebsiteData` Klasse diese Speichert den Header, Titel und Body der Seite in einem String und benutzt eine `List<String>` zur Speicherung aller gefundenen Links auf der Seite.

Die zweite wichtige Datenstruktur ist der `TokenVektor` dieser speichert intern eine `Map<String,Double>` und representiert entweder einen Suchvektor oder einen Urlvektor wobei jeweils Token -> TFIDF-Score gemappt ist.

Im Restlichen Programm werden hauptsächlich `Map<String,Double>` zum mappen von urls/Tokens auf gewisse Scores benutzt. Ab und an wird auch eine `Map<String,`

`Map<String,Double»` benutzt um entweder Url -> (Token -> Score) oder Token -> (Url -> Score) zu mappen.

Grundsätzlich wird meistens die Linked-Version einer Map oder eines Sets verwendet damit die Einfügungsreihenfolge erhalten bleibt.

4 Ergänzungen zu den Aufgaben

Aufgabe 3.5 Der Vergleich der einfachen Ähnlichkeitssuche im Vergleich mit der Kosinusähnlichkeit zeigt dass die Suchergebnisse teilweise komplett anders sortiert bzw priorisiert werden.

Für das Netz cheesy3-7fdaa098 Liefert die Suche mit den in der json angegebenen Querytoken burrata, slovakianbryndza und cantal in einer search query für die Suchmethode mit dem einfachen Ähnlichkeitsmaß folgende Liste absteigend sortiert nach Relevanz:

[http://mozzarella-and-edam.cheesy3, http://gouda-and-muenster.cheesy3, http://quark-and-slovakianbryndza.cheesy3, http://asiago-and-brie.cheesy3, http://rich-mozzarella.cheesy3, http://nutty-lancashire.cheesy3]

Die selbe Suche mit der Kosinusähnlichkeit als Maß liefert folgendes Ergebnis:

[http://mozzarella-and-edam.cheesy3, http://gouda-and-muenster.cheesy3, http://rich-mozzarella.cheesy3, http://quark-and-slovakianbryndza.cheesy3, http://nutty-lancashire.cheesy3, http://asiago-and-brie.cheesy3]

Die selbe Suche mit einer Kombination aus Kosinusähnlichkeit und Pagerank:

[http://gouda-and-muenster.cheesy3, http://rich-mozzarella.cheesy3, http://nutty-lancashire.cheesy http://mozzarella-and-edam.cheesy3, http://quark-and-slovakianbryndza.cheesy3, http://asiago-and-brie.cheesy3]

Aufgabe 4.3 Wenn wir das Netz cheesy6-54ae2b2e mit dem neu implementierten Pagerank versuchen zu crawlen landen wir in einer Endlosschleife. Da wir einen Zyklus haben.

Aufgabe 5.3. Eine multiplikative Kombination von Kosinusähnlichkeit und Pagerank sorgt dafür dass der Algorithmus auch für cheesy6-54ae2b2e funktionert.