Programmdokumentation: AnguillaSearch

Einleitung

Im Rahmen des Moduls „Grundpraktikum Programmierung“ meines Bachelorstudiums in Informatik an der Fernuniversität in Hagen habe ich eine Suchmaschine mit der Programmiersprache Java programmiert.

Aus Komplexitätsgründen wird hier auf die Implementierung eines Webservers sowie eines Graphical User Interfaces verzichtet und sich lediglich auf den Crawler, die Indexierung und den Algorithmus zum Bewerten der gecrawlten Websites konzentriert.

Im Folgenden werde ich auf den grundlegenden Programmaufbau eingehen.

Vector

Die Datenstruktur „Vector“ repräsentiert einen Vektor aus Beziehungen zwischen Token und ihren dazugehörigen TF-IDF-Werten.

Genutzte Datenstrukturen innerhalb der Klasse Vector

- `Map<String, Double> TokenMetrics`
  - Speichert im Schlüssel einen Token (String) und den jeweils dazugehörigen TF-IDF-Wert (Double).

Methoden innerhalb der Klasse Vector

- `addToken(String token, double weight)`
  - Fügt einen Token mit einem dazugehörigen Gewicht zu der Vector-Datenstruktur hinzu.
- `getMetrics(String token)`
  - Nimmt einen Token als Parameter entgegen und gibt seinen zugehörigen Wert zurück.
- `getAllMetrics()`
  - Gibt den gesamten Vektor wieder.

TFIDF

Die Klasse „TFIDF“ wird zum Berechnen des TF-IDF-Werts genutzt.

Ich habe mich dafür entschieden, hierfür eine separate Klasse zu implementieren, da auf die Berechnung des TF-IDF-Werts in mehreren Klassen zugegriffen wird.

ReverseIndex

Die Klasse `ReverseIndex` implementiert die Datenstruktur des Reverse-Index und stellt eine Methode zur Berechnung des TF-IDF-Wertes bereit.

Genutzte Datenstrukturen innerhalb der Klasse ReverseIndex

- `Map<String, Map<String, Double>> reversedIndex`
  - Repräsentiert den Reverse-Index. Die äußere Map nutzt als Schlüssel einen Token (String), also ein Wort, das in der Suche vorkommt. Die innere Map nutzt als Schlüssel die Dokumenten-ID (String) und als Wert den TF-IDF-Wert (Double).

Methoden innerhalb der Klasse ReverseIndex

- `addToken(String token, String docID, double tfidf)`
  - Fügt einen Token, die zugehörige Document-ID und den TF-IDF-Wert hinzu.
- `getReverseIndex()`
  - Gibt die gesamte ReverseIndex-Datenstruktur zurück.
- `getTokenInfo(String token)`
  - Gibt Informationen zu einem Token zurück.
- `tokenizeContent(String content)`
  - Zerlegt den Inhalt eines Strings in Token und wendet Lemmatisierung an.
- `processDocuments(List<JsonObject> documents)`
  - Dient Testzwecken in ReverseIndexTests.java.
- `searchQuery(String searchQuery, boolean useCosineSimilarity, JsonObject json)`
  - Führt eine Suchanfrage aus und bewertet die Ergebnisse.

PageRank

Die Klasse `PageRank` implementiert den PageRank-Algorithmus, welcher die Wichtigkeit der gecrawlten Websites bewertet, wodurch eine effektive Suche gewährleistet werden kann.

Genutzte Datenstrukturen innerhalb der Klasse PageRank

- `Map<String, Double> pageRankValues`
  - Speichert die berechneten PageRank-Werte für jede URL.

Methoden innerhalb der Klasse PageRank

- `calculate()`
  - Zunächst wird das jeweilige Testnetz gecrawlt und die Anzahl der gecrawlten Websites in der Variable `crawledPages` gespeichert. Alle Seiten im Testnetz erhalten einen initialen PageRank-Wert, welcher iterativ für jede Seite neu berechnet wird. Ist `rankDifferenz ¡ epsilon`, wird die Iteration gestoppt.

PageData

Die Klasse `PageData` beinhaltet Setter- und Getter-Methoden für die geparsten Inhalte der Websites, wie beispielsweise den Titel, Headings, Document-ID oder den PageRank.

ForwardIndex

Die Klasse `ForwardIndex` implementiert die Datenstruktur des Forward-Index.

Genutzte Datenstrukturen innerhalb der Klasse ForwardIndex

- `Map<String, Vektor> forwardIndexValues`
  - Bildet auf eine Document-ID einen Vektor ab, welcher den ForwardIndex beinhaltet.

Crawler

Die Klasse `Crawler` definiert einige Methoden, welche für das Crawlen der Websites aus einem gegebenen Testnetz notwendig sind.

Relevante Datenstrukturen innerhalb der Klasse Crawler

- `Queue<String> queueURL`
  - Eine Warteschlange, in welcher sich die Links befinden, die noch von dem Crawler besucht werden müssen.
- `Set<String> visitedURLs`
  - Eine Menge, um vom Crawler bereits besuchte bzw. gefundene Links zu verwalten und Duplikate zu vermeiden.
- `Set<String> allLinks`

  - Eine Menge zur Darstellung der Anzahl an Links innerhalb einer Website.

- `List<PageData> crawledPages`
  - Die gecrawlten Websites werden hier aufgelistet.
- `ForwardIndex forwardIndex`
  - Eigens implementierte Klasse, welche einer gegebenen Website ihren tokenisierten Inhalt zuordnet.
- `ReverseIndex reverseIndex`
  - Eigens implementierte Klasse, welche einem gegebenen Begriff bzw. gegebenen Begriffen das bzw. die zugehörigen Dokumente zuordnet.
- `int totalLinkCount`
  - Zählervariable zum Aufzählen aller existierender Links in einem gegebenen Testnetz.

Methoden innerhalb der Klasse Crawler

- `getSeeds(JsonObject jsonFile)`
  - Implementiert einen grundlegenden Prozess beim Crawlen der Websites. Hierbei werden die im jeweiligen Testnetz vorhandenen Seed-URLs gesucht und diese in die Warteschlange eingefügt, sodass sie in einem nächsten Schritt abgearbeitet werden können.
- `getFurtherUrl(String url)`
  - Überprüft, ob diese URL auf weitere Websites verlinkt, und fügt diese Links der Warteschlange hinzu.
- `getAllLinks(String url)`
  - Gibt die Anzahl aller Links, auf die die gegebene URL verlinkt, als ganzzahligen Wert zurück.
- `getAllLinksFromURL(String url)`
  - Gibt alle von dieser URL verlinkten Websites der Datenstruktur `Set<String> allLinks` zurück.
- `getAllLinksFromJSON(JsonObject jsonFilePath)`
  - Gibt alle im entsprechenden Testnetz gefundenen Links als ganzzahligen Wert zurück.
- `crawl(JsonObject jsonFilePath)`
  - Führt den eigentlichen Crawling-Vorgang durch.
- `getVisitedWebsited(JsonObject jsonFilePath)`

  - Gibt die Anzahl der besuchten Websites zurück.

- `getCrawledPages()`
  - Gibt eine Liste der gecrawlten Websites zurück.
- `getForwardIndex()`
  - Gibt den Forward-Index zurück.
- `getReverseIndex()`
  - Gibt den Reverse-Index zurück.
- `calculateTFIDF()`
  - Berechnet den TF-IDF-Wert jeder gecrawlten Seite.
- `rankWebsites(Vektor queryVektor)`
  - Bewertet Websites mit Hilfe einer Kombination des PageRank-Algorithmus sowie der Cosinus-Ähnlichkeit.