A4: Erklärungen zu Solr und Programmen

Solr

Der Solr-Server befindet sich im Ordner *solr-5.1.0* auf dem mit abgegebenen USB-Stick. Man startet den Server, indem man in der Kommandozeile den Befehl „java –jar start.jar“ im Ordner solr-5.1.0/server/ ausführt.

AdvancedQuery: Bestimmung der Scores in allen Feldern

Die Klasse, mit der in dieser Arbeit die meisten Querys an Solr gesendet werden, ist die *AdvancedQuery*-Klasse. Auf ihre Funktionsweise wird hier im Detail eingegangen, weil diese bestimmt, wie die Scores berechnet werden und das hat einen Einfluss auf der Ergebnisse der Evaluation. Eine „normale“ Query mit der solrj-API (die eine Java-Schnittstelle zu Solr bietet) heißt *SolrQuery*.

Die *AdvancedQuery* bietet folgende Funktionalität:

1. Man kann für ein Feld mehrere Querys angeben; die damit jeweils erzielten Scores werden einzeln abgefragt aufsummiert.
2. Man kann mehrere zusätzliche Felder angeben, deren Werte abgefragt werden, ohne einen Einfluss auf den Score zu nehmen.
3. Es ist möglich, die Ergebnisse zu einem anderen Feldnamen zu speichern als den eigentlich abgefragten.

Beispiel:

Bei Topic 161 geht es um die Rolle des IDE-Gens in der Alzheimer-Erkrankung. Man erstellt dafür ein *AdvancedQuery*-Objekt und sagt, dass man die Teilquerys „ide“ und „alzheimer“ jeweils in folgenden Feldern durchführen möchte (Funktion 1):

language_worddelimiter_rank_lm_1_abstract

language_worddelimiter_rank_lm_1_title

language_worddelimiter_rank_lm_2_abstract

language_worddelimiter_rank_lm_2_title

Darüber hinaus möchte man für dieselben Felder die Werte für die Querys „insulin protease“ und „dementia“ erhalten, den Einfluss dieser Werte auf die Ergebnisse jedoch separat bestimmen können, indem man einen anderen Feldnamen angibt, unter dem das gespeichert wird (Funktion 3).

Außerdem will man den PageRank für jedes Dokument abfragen, das man findet (Funktion 2).

Die *AdvancedQuery* würde dann eine „normale“ Query für „ide“ im Feld „language_worddelimiter_rank_lm_1_abstract“ an Solr senden und für alle Ergebnisse den Score sowie den Wert für den PageRank speichern. Dann würde sie die Query „alzheimer“ im selben Feld durchführen und dasselbe machen; wenn sich ein Dokument bereits in den Ergebnissen befindet, dann wird der neue Score addiert.

Das Gleiche passiert nun mit den Querys „insulin protease“ und „dementia“, nur dass die Ergebnisse hierfür unabhängig von denen von „ide“ und „alzheimer“ zwischengespeichert werden.

Das wird für alle Felder wiederholt, die Ergebnisse der Felder sind unabhängig voneinander.

Am Ende könnte man die Ergebnisse anhand von bestimmten Regeln sortieren, indem man Gewichte für die verschiedenen Felder angibt, oder man lässt sich die Ergebnisse im .csv-Format ausgeben.

Wichtig ist insgesamt, dass keine Query nach „role of ide in alzheimer’s disease“ oder „role ide alzheimer“ oder „ide alzheimer“ durchgeführt wird, sondern zwei Querys: eine nach „ide“ und eine nach „alzheimer“. Die Scores werden dann addiert. Das kann andere Scores ergeben als eine Query nach beiden Termen auf einmal.

Erstellte Programme

Die entwickelten Java-Programme befinden sich auf dem USB-Stick, der mit abgegeben wurde, sowie auf dem GitHub-Repository https://github.com/JonBrem/PubMed-Bachelorarbeit. Der Code kann nicht ausgedruckt abgegeben werden, da etwa 6500 Zeilen Code erzeugt wurden. Es würde außerdem keinen Sinn machen, weil darüber hinaus die mehrere GB große Kollektion sowie der Solr-Server notwendig sind, um die Programme ausführen zu können.

Es gibt kein UI für die Programme. Es wurde auf Codequalität geachtet und es sind Kommentare zur Unterstützung vorhanden, jedoch wurde ebenfalls berücksichtigt, dass die Programme in den meisten Fällen etwas sehr Spezielles durchführen müssen.

Es deswegen nicht möglich, Parameter an die Programme zu übergeben, sondern die Konfiguration erfolgt in der *main*-Methode im Code der Dateien.

Um eine echte Suchmaschine oder Such-API zu entwickeln, wären nur die Dateien im Package *de.ur.jonbrem.pubmed.advanced_querying* sowie

de.ur.jonbrem.pubmed.solrconnection notwendig. Der Vollständigkeit halber wird noch auf ein paar weitere Programme bzw. Klassen verwiesen, die zentrale Punkte der Arbeit darstellen:

- Package *de.ur.jonbrem.pubmed.indexing*:
    - „Indexer“ und „MeshIndexer“ erstellen die Kollektion, indem sie die XML-Dateien lesen und die Informationen an den Solr-Server schicken.
    - Im Subpackage „citations“: die Klasse „HighwireCitationFinder“ beinhaltet die Funktionalität, die Links innerhalb der Kollektion zu bestimmen.
    - Im Subpackage „pagerank“ werden mit dem Programm „CitationRankBuilder“ die PageRank-Scores der Dokumente berechnet.
- Package *de.ur.jonbrem.pubmed.test.machine_learning.own_learning*:
    - Das Programm „FastOptimization“ beinhaltet den Machine Learning-Algorithmus.

Die eigentliche Berechnung der Ergebnisse findet aber im „CalculatorThread“ statt.