Modellierungsbeschreibung: Datensuche für Linked Data

1 Allgemeines

Das Semantic Web und seine dazugehörigen Komponenten wachsen jeden Tag. Dazu gehört somit auch Linked Open Data. Um diese Daten effizient nutzen zu können benötigt man eine Suchmaschine. Diese wurde für Linked Data durch das Open-Source-Project "Tapioca" bereits implementiert. Allerdings gibt es einige Punkte, in denen Tapioca verbessert werden kann.
Im Rahmen des Softwaretechnik-Praktikums soll die Suchmaschine verbessert werden. Dabei wird eine benutzerfreundliche Webseite erstellt, auf der man eine Schlagwortsuche oder eine Suche durch das Hochladen einer RDF-Datei starten kann.

2 Produktübersicht

Das Produkt besteht aus einer Webseite, einem Indexgenerator, einem Benchmarking Tool und einem Meta Data Extraction Tool.
Das Webinterface wird dabei auf einem Tomcat-Server aufgesetzt, da eine Kommunikation zwischen der Webseite und einem Java-Programm stattfinden soll. Die Suche wird, wie schon bemerkt, entweder durch Schlagwörter, oder durch eine RDF-Datei möglich sein.

3 Grundsätzliche Struktur- und Entwurfsprinzipien

3.1 Generelle Idee

Das Projekt lässt sich insgesamt in drei kleinere Teilprojekte gliedern, aus diesem Grund haben wir uns entschieden drei Teilprojekte in unserem Git-Repository zu erstellen.

Für die Benutzer/innen der Suchmaschine steht die Webseite im Vordergrund. Diese sollte somit einfach und intuitiv bedienbar sein. Es wird Wert auf eine Oberfläche gelegt, die einen nicht "überrumpelt", aber dennoch viel Funktionalität liefert.

Das Meta Data Extraction Tool wird von den Benutzer/innen der Webseite benötigt, um die Metadaten aus ihren Dateien zu filtern. Das macht das Vergleichen ihrer Datensätze einfacher.

Der Indexgenerator wird von dem/der Administrator/in verwendet um in regelmäßigen Abständen den Index der Suchmaschine zu erstellen und zu aktualisieren. Dieser ist daher für die Nutzer/innen nicht sichtbar. Der Indexgenerator wird am Ende des Projektes noch um eine Benchmarking-Funktionalität erweitert, womit es möglich wird, die Qualität des erstellten Index zu messen.

3.2 MVC-Architektur und JSF

Der wichtigste Bestandteil des Projektes ist zweifelsohne die Website mit der Suchmaschine, weil hier die Interaktion mit einer hoffentlich breiten Nutzerschaft stattfindet. Beim Entwurf dieser orientieren wir uns am bekannten "Model-View-Controller" Ansatz. Realisiert wird dieser mit dem "Java Server Faces" Framework, Version 2.2. Dieses überträgt den MVC-Ansatz, genauer gesagt "Model2", auf die Welt der Webentwicklung mit Java. Als "Model" kommen dabei die gewohnten Java-Klassen zum Einsatz, der "View" wird mit sogenannten "Facelets (XHTML)" realisiert und als "Controller" werden "Servlets" und "Managed Beans" verwendet. Letztere bilden die Schnittstelle zwischen der Ansicht im Webbrowser und der Geschäftslogik in Form einer Java-Klassenhierarchie, denn sie sind für den Datenaustausch zwischen diesen beiden Schichten zuständig.

3.3 Tapioca, Topic Modelling und Jena

Das ganze Projekt ist im wesentlichen ein Refactoring vorhandenen Codes. Dafür verwenden wir das öffentliche Tapioca Repository von Michael Röder auf Github. Hauptbestandteil unserer Arbeit ist die anwendungsorientierte Umgestaltung und ggf. Erweiterung der dort implementierten Funktionalitäten. Diese Klassen können wir verändern.

Im Gegensatz dazu stützt sich das Tapioca Projekt selbst wiederum auf zwei andere Projekt, deren Code wir lediglich mitverwenden, aber nicht verändern. Eines davon ist das Topic Modelling Projekt, welches, wie der Name schon verrät, vor allem das Topic Modelling implementiert. Diesen Code modifizieren wir nicht, binden ihn stattdessen über ein Maven Repository in unser Projekt ein.

Die zweite wichtige Quelle unserer Funktionalität ist das Apache Jena Semantic Web Framework. Dieses bietet für Java eine gute Schnittstelle um mit Semantic Web Daten, als RDF-Graphen repräsentiert, arbeiten zu können. Viele Input-/Output-Operationen sowie einfache Datenverarbeitungsschritte verwenden Klassen dieses Frameworks, welche wir ebenfalls nicht modifizieren, sondern wiederum über ein Maven Repository in das Projekt einbinden.

3.4 Elasticsearch

Um die Suchmaschine noch nützlicher zu machen, soll sie neben der Möglichkeit zu einem gegebenen RDF-Datensatz thematisch ähnliche Datensätze via Topic Modelling zu suchen, auch das Feature bieten, direkt mittels Schlagworten nach passenden Datensätzen zu suchen. Um dieses Feature nutzen zu können, wird ein Elasticsearch Server installiert und mittels REST-API an die eigene Java Applikation angebunden. Elasticsearch ist ein auf Apache Lucene basierendes Open Source Projekt, welches die Volltextsuche und -analyse ermöglicht.

3.5 Pakete und die Wiederverwendbarkeit von Klassen

Die Java Klassen werden wie üblich in Pakete aufgeteilt, was die Übersichtlichkeit erhöht und die Importierbarkeit erleichtert. Das ist für unser Projekt aber vor allem deswegen sinnvoll, weil viele der Tapioca Klassen mehrmals verwendet werden können. So muss zum Beispiel die Suchfunktion der Website viele Dinge können, die auch der Indexgenerator können muss, auch wenn es am Ende zwei separate Programme sind, etwa ein Output File aus der Meta Data Extraction einlesen und weiter verarbeiten oder ein Corpus Dokument auf einen Themenvektor abbilden.

4 Struktur- und Entwurfsprinzipien einzelner Pakete

4.1 Webseite

Die Webseite ist die zentrale Anlaufstelle für den Nutzer. Diese sollte, wie bereits erwähnt, einfach und intuitiv bedienbar sein. Das Eingabefeld kann der Nutzer benutzen, um ein Schlagwort einzugeben, oder eine Datei per Drag-And-Drop hochzuladen um so danach zu suchen. Außerdem wird es ein Menü geben, welches weitere Informationen und Funktionen bereitstellt. Das Menü liegt in der oberen rechten Ecke und teilt sich in drei Punkte auf. Der Menüpunkt "Readme" liefert weiterführende Informationen zur Benutzung der Suchmaschine. Im Punkt "Tools" werden für Benutzer Werkzeuge, wie das Meta Data Extraction Tool, zur Verfügung stehen. Als letzten gibt es den "Contact", welches das Impressum der Seite darstellt.
Nach der Suche gelangt der Nutzer auf eine Ergebnisseite. Auf dieser findet man sowohl das Menü, als auch eine Suchleiste wieder, sodass der User nicht auf die Homepage zurück muss, um eine weitere Suche zu starten.
Die Ergebnisse bestehen aus einem Kopf (Link zur Datei), der Ähnlichkeit (zwischen 0 und 1) und dem Körper mit einem kleinen Auszug aus der Datei.
Da dieses das erste Paket ist, ist die eigentliche Suchfunktion noch nicht eingebaut. Stattdessen liefert die Ergebnisseite vorprogrammierte Suchergebnisse, welche die Suchanfrage in den Köpfen stehen haben.

4.2 Meta Data Extraction

Das Meta Data Extraction Tool ist eine Anwendung, die der Benutzer vor der Suche auf seinen eigenen Daten benutzen sollte.
Die eingegebenen RDF-Daten können dabei die Formate `JSON-LD`, `N-TRIPLES`, `N3`, `RDF/JSON`, `RDF/XML`, `RDF/XML-ABBREV` oder `TURTLE` haben. Dabei kann der Nutzer sogar entscheiden welches Ausgabeformat die resultierende Datei haben soll. Man kann die Dateien also ineinander umwandeln.
Als erstes werden VoID-Information extrahiert. Dabei wird ein VoID-Model erstellt. Danach werden die Labels der RDF-Datei erfasst und zum Model

hinzugefügt. Nach diesen Schritten wird das Model in eine Datei mit entsprechenden Ausgabeformat geschrieben.

4.3 Document Generation

Dieses Paket ist Teil des Indexgenerators. Seine Aufgabe besteht darin, die eingelesenen Metadaten über RDF-Datensätze in einen Textkorpus umzuwandeln, wobei der jeweilige Datensatz durch eine Menge von zu ihm passenden Schlagworten repräsentiert wird. Diese Funktionalität wird auch für die durch Hochladen von Metadaten ausgelöste Suche genutzt. Es kann zwischen verschieden Zählweisen der "Classes" und "Properties" ausgewählt werden.

4.4 Indexgenerator

Der mittels Document Generation erzeugte Korpus wird mit Hilfe eines Topic Modelling Verfahrens in ein entsprechendes Themenmodell umgewandelt und somit ein Index erstellt. Diese Funktionalität vervollständigt also das zweite eigenständige Programm unseres Projektes, den Indexgenerator. Die für die Modellerzeugung verwendete Themenanzahl kann als Parameter übergeben und somit manuell festgelegt werden. Angemerkt sei noch, dass durch die Verwendung von Java Interfaces die Option geschaffen wird, das standardmäßige Topic Modelling Verfahren, nämlich Latent Derichlet Allocation, unkompliziert durch ein anderes zu ersetzen.

4.5 Model Generation

Der mittels Document Generation erzeugte Korpus wird mit Hilfe eines Topic Modelling Verfahrens in ein entsprechendes Themenmodell umgewandelt und somit ein Index erstellt. Diese Funktionalität vervollständigt also das zweite eigenständige Programm unseres Projektes, den Indexgenerator. Die für die Modellerzeugung verwendete Themenanzahl kann als Parameter übergeben und somit manuell festgelegt werden. Angemerkt sei noch, dass durch die Verwendung von Java Interfaces die Option geschaffen wird, das standardmäßige Topic Modelling Verfahren, nämlich Latent Derichlet Allocation, unkompliziert durch ein anderes zu ersetzen.

4.6 Elasticsearch

Die Schlagwortsuche wird mit Elasticsearch Version 1.4 realisiert. Zur Anbindung an unsere Webanwendung verwenden wir die von Elasticsearch bereitgestellte "Java API". Mittels dieser werden beim Start des Servers alle nötigen Metadaten aus einem N-Triples-File ausgelesen und in den Suchmaschinenindex eingetragen. Dabei werden pro Datensatz nur drei Werte abgelegt: der Titel, seine URI und die Beschreibung. Eine Suchanfrage in unserer Webanwendung wird mit der Java API entsprechend weitergeleitet und die 20 besten Ergebnisse, so denn vorhanden, werden zurückgeliefert und schließlich auf der Ergebnisseite gerendert und dargestellt.

4.7 Inferencer

Der Inferencer ist Teil der Suchmaschine. Lädt ein Benutzer oder eine Benutzerin Metadaten über einen RDF-Datensatz hoch, so sucht der Inferencer dazu ähnliche Datensätze aus dem Index heraus und berechnet zu diesen gemäß eines vorgegebenen Verfahrens, "Cosine Similarity of Vectors", jeweils die Ähnlichkeit zum Eingabedatensatz.
Hier wird das Topic Model, welches während der Model Generation erstellt wurde, benutzt, um zu jedem Dokument im Corpus einen entsprechenden Vektor zu erstellen. Auch der in der Suchmaschine hoch geladene Datensatz bekommt einen Vektor. Das oben genannte Verfahren wird dann auf alle Vektoren angewandt. Die Dokumente werden dann nach der Ähnlichkeit sortiert und auf der Webseite ausgegeben.

5 Datenmodell

5.1 Vorbereitung

Bevor man Suchanfragen an die Suchmaschine stellen kann, muss einiges an Vorarbeite geleistet werden. Wir besitzen einen großen Satz an RDF-Dokumenten, die für den späteren Index genutzt werden sollen. Bevor das möglich ist muss mit Hilfe des Meta Data Extraction Tools Informationen aus dem Corpus extrahiert werden. Basierend auf der Meta Data wird ein neuer Corpus erstellt, welcher für jede RDF-Datei jeweils den Namen, die URI, die Beschreibung und den eigentlich Text speichert. Dieser Corpus wird dann wiederum verwendet, um das Topic Model zu erstellen. Dieses Model ist wichtig, da für jedes RDF-Dokument ein Themenvektor erstellt wird und der Inferencer auf Grund dieser Vektoren die Ähnlichkeit zur Eingabe berechnet. Eine bildliche Darstellung des kompletten Datenverlaufs ist im Anhang (A1) zu finden.

5.2 Suche per Dokument

Der Benutzer besitzt ein RDF-Dokument. Um damit die Suche zu starten, muss er erst mit dem Meta Data Extraction Tool die Meta Data aus dem Dokument generieren. Die entstandene Datei wird dann hochgeladen. Es wir mit dem entsprechenden Topic Model einen Themenvektor erstellt und über einen speziellen Algorithmus die Ähnlichkeit berechnet und sortiert auf der Webseite angezeigt.

5.3 Suche per Schlagwort

Der Benutzer weiß ungefähr wonach er suchen möchte, hat aber noch keine entsprechende Datei. Er kann nun auf der Webseite ein Schlagwort eingeben. Wir haben auf Grundlage der Titel und der Beschreibungen einen Index erstellt. Um die effektive Suche kümmert sich dann das Framework Elasticsearch.

6 Glossar

MVC steht für Model-View-Controller und ist ein Architekturmodell der objektorientierten Programmierung. Es gibt eine strikte Rollenverteilung und teilt ein Programm in ein Datenmodell (model), der Datenpräsentation (view) und der Programmsteuerung (controller) auf.

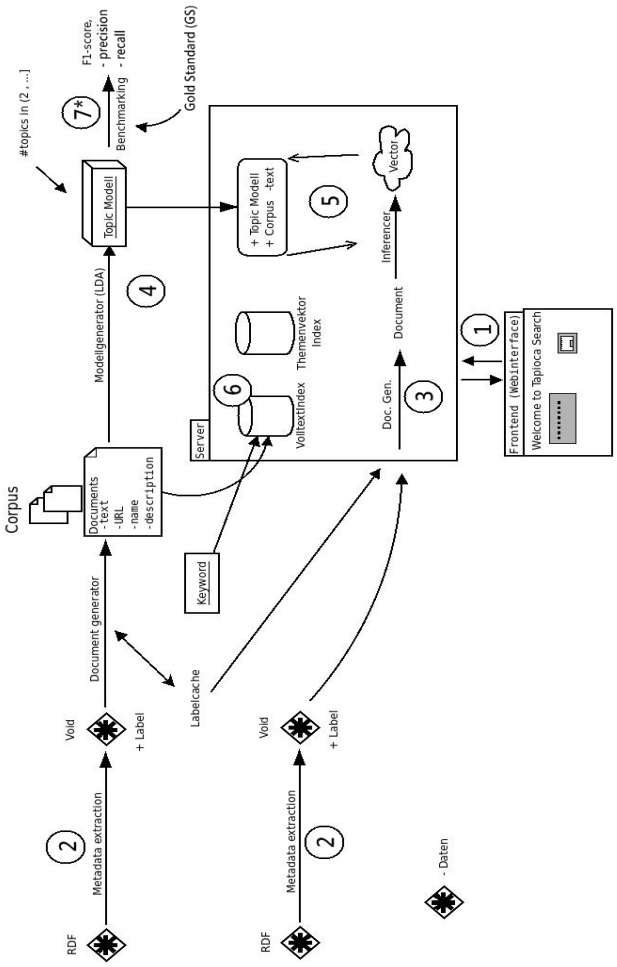
RDF steht für Resource Description Framework und erweitert das Web um die Möglichkeit Inhalten miteinander zu verbinden. In RDF-Dateien werden Aussagen über Ressourcen getroffen, wobei Ressourcen eindeutig bezeichnete Dinge der Welt sind. Dabei werden durch einen Graphen (bzw. Tripel) Ressourcen mit anderen Ressourcen verbunden.

Metadata beinhalten Merkmale über andere Daten. So werden oft größere Datensammlungen (Dokumente, Bücher, etc.) beschrieben.

Corpus bezeichnet die Ansammlung von Daten. Das kann eine Sammlung von Wörtern, Sätzen, Kapitel, Bücher, Dokumente, etc. sein.

Topic Modelling ist das Verfahren, welches wir zum Aufteilen von Wörter in verschiedene Themen benutzen. Dabei werden alltägliche Stopwörter, wie “der“, “die“, “das“ etc. herausgefiltert, da diese das Ergebnis negativ beeinflussen.

Anhang A1


#topics in (2 ... n)
F1-score,
- precision
- recall
Benchmarking
Gold Standard (GS)
7*
Topic Modell
ModelGenerator (LDA)
4
+ Topic Modell
+ Corpus -text
5
Vector
Inferencer
Themenvektor
Index
Volltextindex
6
Server
Doc. Gen.
3
Document
1
Frontend (WebInterface)
Welcome to Tapioca Search
Corpus
Documents
-text
-URL
-name
-description
Keyword
Document generator
LabelCache
Void
+ Label
2
Metadata extraction
RDF
- Daten