Arbeitsplan der Projektgruppe dsl-16

Marleen Wagner & Jana Nüßler

February 2, 2016

Contents

1 Projektvision

Im Zuge dieses Softwaretechnik-Praktikums (Datensuche für Linked Data), soll das vom Projekt Owner angefertigte Produkt Tapioca in ein neues System eingebunden werden.
Den Nutzern soll es möglich sein über eine Internetseite eine Browseranfrage zu starten. Diese soll aus Schlagworten oder den extrahierten Metadaten eines RDF-Datensatzes bestehen. Das Extrahieren der Metadaten aus dem Datensatz erfolgt vorher lokal auf dem Rechner des Nutzers.
Bei Schlagworten werden dem Nutzer Datensätze mit entsprechenden Themen ausgegeben. Falls er jedoch einen Datensatz nutzt um die Suche zu triggern,so werden die Metadaten mit denen von vorhandenen Datensätzen abgeglichen und als Ausgabe erscheint eine Menge von thematisch ähnlichen Datensätzen, diese sind absteigend nach der Ähnlichkeiten zur Eingabe sortiert. Der Vergleich erfolgt wie folgt: Zuerst werden die Metadaten des Eingabedatensatzes extrahiert und in einem Dokument zusammengefasst. Daraus wird ein Themenmodell erstellt, welches die Verteilung von Themen pro Dokument modelliert. Zudem wird auch die Wortverteilung pro Thema modelliert. Durch den vom ProjectOwner gestellten Inferencer, lässt sich daraus ein Vektor schaffen, welcher letztlich das Vergleichen mit anderen Vektoren, der vorhandenen Datensätzen erlaubt. Hierbei kommt ein manuell

angefertigter Goldstandard ins Spiel, anhand dessen sich die Qualität der Suche einschätzen lässt (mittels eines F1 Score wird die Übereinstimmung zwischen Goldstandard und durchgeführter Suche ermittelt).
Insgesamt sollen drei Programme und eine Weboberfläche entstehen, genauere Erläuterungen dazu werden bitte "Abschnitt 3 - Designübersicht und Funktionalität" entnommen.

2 Vorraussetzungen

Tapioca ist eine Engine zur Suche von Linked Data Datensätzen. Diese Suchmaschine hat im Hintergrund Teile der LOD-Cloud mit über 1600 RDF-Datensätze zur Verfügung. Für diese Daten wurden Metadaten extrahiert, um so für jeden einzelnen Datensatz ein Dokument zu erstellen, welches den Datensatz beschreibt. Außerdem wird aus den Metadaten ein Themenmodell (Verteilung der Themen) generiert. Daraus kann man die Ähnlichkeit zu anderen Datensätzen bestimmen.
Es gibt jedoch noch einige Problem:

1. Man muss für das Themenmodell eine günstige Anzahl von Themen wählen. Diese Auswahl ist allerdings nicht leicht, da bisher nur mit einem eigenen Goldstandard gearbeitet wurde.

2. Die Eingabe eines Users soll außerdem durch Schlagworte, oder durch ein eigenes VoID-Dokument eines RDF-Datensatzes möglich sein.

3. Ein weiteres Problem stellt die Metadataextraction dar, da diese sehr zeitaufwändig ist.

4. Weiterhin soll das Testen einer Suche mittels vorgegebenem Goldstandard ermöglicht werden. Dazu wird ein Benchmarking-Tool benötigt.

Diese Probleme sollen durch die Arbeit unserer Gruppe beseitigt werden. Dazu soll ein Kommandozeilenprogramm zur Metadataextraction erstellt werden, um diese extern durchführen zu können.
Da die Anzahl von verfügbaren Datensätzen stetig wächst, soll es zur jeder Zeit möglich sein, die Anzahl von Themen zur Themenmodellierung neu zu berechnen, da die Qualität der Suche von der Anzahl der verwendeten Themen abhängt.
Es soll weiterhin eine Volltextindexierung geben, die mit der Beschreibung der Dokumente arbeitet.
Somit wird unser Team eine verbesserte und modularisierte Variante von TAPIOCA erstellen, welche die grundlegenden Konzepte weiter verwendet.

3 Designübersicht und Funktionalität

Das Projekt besteht aus drei separaten Programmen, welche im Folgenden nacheinander kurz beschrieben werden : 1. Programm zur Metadata-Extraction, 2. Indexierungstool, 3. RDF-Datensatz-Suchmaschine. Für diese Suchmaschine wird eine einfach zu bedienende Weboberfläche aufgesetzt über welche die User ihre Suche ausführen können. Die Benutzbarkeit der Weboberfläche wird anschließend mittels SUS Skala evaluiert, genaueres dazu kann dem Testkonzept entnommen werden. In 4. werden noch ein paar Worte zur Modularität verloren und schließlich wird in 5. zum besseren Verständnis noch ein graphisches Programmschema präsentiert.

3.1 Programm zur Metadataextraction

Input: Ein Semantic-Web-Datensatz im RDF-Format, wobei die folgenden Serialisierungsformate unterstützt werden sollen: RDF/XML, N-Triples, Turtle, RDF/JSON. Um das Tool anzusprechen, reicht ein Commandline-Interface aus.
Output: Ein RDF-Dokument mit Metadaten über den Eingabedatensatz. Enthalten sind Informationen über alle im Dokument auftretenden Klassen und Merkmale (Properties), seine Adresse sowie (falls vorhanden) eine menschliche Beschreibung. Dafür wird das VOID-Vokabular genutzt.
Algorithmus: Es wird anhand festgelegter Kriterien bestimmt, welche URIs Klassen und welche Merkmale sind. Außerdem werden manche URIs auch ausgeschlossen, weil sie keine nützlichen Informationen hinsichtlich des Themas enthalten. Ebenso wird die Adresse und die Beschreibung aus dem RDF-Datensatz extrahiert.

3.2 Indexierungs-Tool

Input: Hier gibt es verschiedene Möglichkeiten. Zur Ansteuerung dieses Tools ist ebenfalls ein Commandline-Interface ausreichend.

1. (a) Eingabe einer Menge von Metadaten über RDF-Datensätze (Tool, s.o.), wobei hier ebenfalls die Formate RDF/XML, N-Triples, Turtle und RDF/JSON unterstützt werden sollen,
   (b) Angabe einer bestimmten Variante der Dokument-Erzeugung, (c) Vorgabe der für die Modellerzeugung zu nutzenden Themenanzahl, sowie (4) ggf. Auswahl eines zu verwendenden Topic-Modelling-Verfahrens.
2. Eingabe eines Ähnlichkeitsschwellwertes und eines Goldstandards, auf Basis dessen Performance-Messungen über den generierten Index durchgeführt werden können.
3. Außerdem soll es die Möglichkeit geben, 1. und 2. zu kombinieren und so automatisiert die optimale Themenanzahl zu bestimmen.
4. Eingabe von Metadaten wie bei 1., allerdings wird jetzt kein Index mit Topic-Modelling erzeugt, sondern lediglich einer zur Schlagwortsuche.

Output: Entsprechend gibt es auch hier verschiedene Varianten.

1. Ein File bzw. eine Menge von Files, welche den gesamten Index enthalten.
2. Recall, Precison und F1-Score für die indexierte Datensatzmenge und die entsprechend ausgewählten Parameter
3. Der Index und die Performance-Messwerte, ggf. ein Log-File mit den Messwerten für jeden Iterationsschritt zur Bestimmung der optimalen Themenanzahl.

4. Ein File bzw. eine Menge von Files, welche den gesamten Index enthalten.

Algorithmus: Abhängig vom Szenario sind verschieden Dinge zu tun:

1. Im ersten Fall muss zunächst für jeden Datensatz ein Dokument erzeugt werden, welches Klassen, Merkmalen oder beiden jeweils Label zuordnet und diese in Form von Text in ein Dokument einträgt. Je nach Variante wird dabei auch noch die Vorkommenshäufigkeit der entsprechenden Label berücksichtigt. Anschließend wird mittels eines Topic-Modelling-Verfahrens (z.B. LDA) ein Topic Model erzeugt und jedem Dokument ein Themen-Vektor zugeordnet. Die hierfür verwendete Anzahl von Themen hängt von der Eingabe ab. Außerdem wird jedem Vektor natürlich noch sein Name, seine Adresse und seine Beschreibung zugeordnet.

2. Der eingegebene Goldstandard enthält eine Ähnlichkeitsmatrix für eine Teilmenge der im Index repräsentierten Datensätze. Nun wird auf Basis des in 1. erzeugten Index eine zweite Ähnlichkeitsmatrix über genau dieser Teilmenge generiert. Abhängig vom Schwellwert können nun Precision, Recall und F1-Score berechnet werden.

3. Starte mit einer niedrigen Anzahl von Themen und führe iterativ immer wieder 1. und 2. aus. Erhöhe hierzu bei jeder neuen Indexerzeugung die Themenanzahl so lange, bis der F1-Score in der zugehörigen Messung nicht mehr weiter ansteigt.

4. Hier kann auf schon bestehende Tools, welche auf Apache Lucene aufbauen, zurückgegriffen werden. Selbstverständlich soll auch hier eine Zuordnung zu Name, Adresse und Beschreibung des eigentlichen Datensatzes möglich sein.

3.3 RDF-Datensatz-Suchmaschine

Input: Hier gibt es, wie sich bereits andeutete, zwei Varianten. Die Eingabe erfolgt diesmal über eine ansprechende und benutzerfreundlich gestaltete Webseite.

1. Eingabe von Metadaten über einen RDF-Datensatz, erzeugt mit dem Tool aus 1. oder einem ähnlichen.

2. Eingabe von Schlagworten

Output: Die Ausgabe soll an eine Suchmaschine erinnern. Die Ergebnisse werden mit Überschrift, Link zum Datensatz und Beschreibung angezeigt. Dabei sind die Ergebnisse nach Ähnlichkeit mit dem Eingabedatensatz absteigend geordnet. Der Ähnlichkeitswert wird mit ausgegeben, an der vordefinierten Schwelle wird die Liste allerdings abgebrochen.

Algorithmus: Hier sind nun zwei Szenarien zu betrachten, einmal die Suche mit Metadaten und einmal die Suche über Schlagworte.

1. Zu den eingegebenen Metadaten ist zunächst ein Dokument, wie oben schon beschrieben, zu generieren. Anschließend muss mittels desselben Themenmodells, mit welchem auch der Index schon generiert

wurde, ein Themen-Vektor für das Dokument erzeugt werden. Dieser Themen-Vektor kann nun mit jedem Themenvektor im Index verglichen und ein Ähnlichkeitswert berechnet werden. Auf Basis dessen und mit dem vordefinierten Schwellwert kann nun das Suchergebnis zusammengestellt werden.

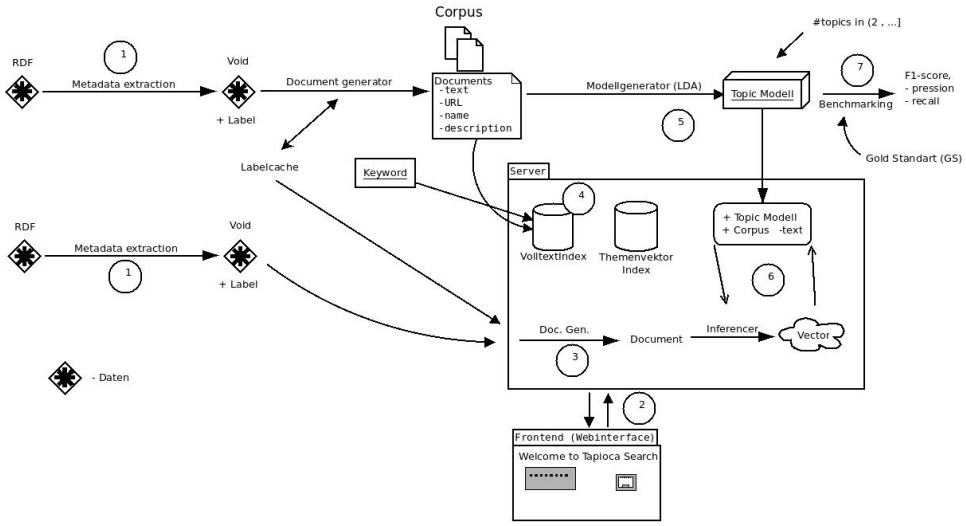
2. Die Umsetzung der Schlagwortsuche wird im wesentlichen einer bereits bestehenden Lösung, basierend auf Apache Lucene, überlassen. Im Ergebnis müssen natürlich auch wieder Name, Adresse und Beschreibung in absteigender Reihenfolge bis zu einem bestimmten Schwellwert angezeigt werden.

3.4 Modularität

Folgende Modularisierung der Programmteile bietet sich an:

- Metadaten-Extraktion, wird bei 1. gebraucht
- Dokument-Erzeugung, wird bei 2. und 3. gebraucht. Hier soll es außerdem möglich sein, verschiedene Varianten auswählen zu können.
- Topic-Model-Erzeugung, wird bei 2. gebraucht, Teile davon aber auch bei 3. Außerdem soll es möglich sein verschiedene Verfahren fürs Topic-Modelling verwenden zu können
- Themen-Vektor-Berechung zu einem Datensatz, wird in 2. und 3. gebraucht und hängt außerdem vom Topic-Modelling-Verfahren ab.
- Indexbildung auf Basis von Topic Model, wird in 2. gebraucht
- Indexbildung auf Basis von Schlagwortsuche, wird in 2. gebraucht
- Suchen über Themen-Vektor-Index, wird in 3. gebraucht, aber auch in 2. fürs Benchmarking.
- Suchen über Schlagwort-Index, wird in 3. gebraucht
- Vergleichen von Suchergebnis mit Goldstandard, wird in 2. gebraucht.
- Ein-/Ausgabemodul zur Kommunikation mit dem Web-Frontend, wird in 3. gebraucht.

3.5 Schematische Darstellung



4 Arbeitspakete

Die Gruppe einigte sich auf Grundlage der in Abschnitt 3 erläuterten Funktionalitäten auf 7 **Releasebündel zur Umsetzung der Muss-Ziele.** Die angegebene Prozentzahl spiegelt jeweils den geschätzten Aufwand bzgl. des Gesamtprojektes wieder

1. Ein gekapseltes Commandline Programm zur MetadataExtraction, also die Vorverarbeitung der RDF-Daten ins VoiD-Format, mit welchem die Suche auf der Webseite getriggert wird *5%*

2. Der TomCat-Server mit statischer Webseite wird aufgesetzt, das Webseiten-Design steht (ohne Funktionalität, lediglich die Kommunikation zwischen Server und Client soll möglich sein) *15%*

3. Die Document Generation/ Indexierung als Teil der Suchmaschine wird realisiert *10%*

4. Der VolltextIndex und die VolltextSuche werden mittels Lucene realisiert. Dadurch kann nun die Schlagwortsuche als Teilfunktionalität auf der Webseite bereits genutzt werden, auch wenn das Suchen mittels VoiD-Format noch nicht möglich ist *15%*

5. Die Topic Model Generation mittels LDA wird realisiert damit diese als Input für die Themenvektormodellierung zur Verfügung steht *20%*

6. Der Inferencer wird eingebaut und errechnet aus dem Topic Model einen Themenvektor, welcher nun zum Vergleich zu den bereits Indexierten Datensätzen genutzt werden kann, der Vergleich wird ebenfalls realisiert *25%*

7. Das Benchmarking-Tool wird in die Indexierung eingebaut, automatisiertes Finden der optimalen Themenanzahl und Messung der Qualität der Suche am Goldstandard werden ermöglicht *10%*

Gesamtaufwand der Muss-Ziele: *100%*

Bei der Anforderungsanalyse mit dem Product Owner kamen folgende Funktionalitäten als **Kann-Ziele** zur Sprache:

- Ein Webcrawler, der ermöglicht, dass nicht nur Ähnlichkeit zwischen dem Eingabedatensatz und den 1680 bereits indexierten Datensätzen auf dem Server errechnet werden, sondern auch im Netz nach passenden Datensätzen gesucht werden kann. *25%*
- Statt der Latent-Dirichlet-Allocation können vom Nutzer auch andere BaseLines zur Berechnung des Themenmodells ausgewählt werden (auch wenn diese vergleichsweise schlechter arbeiten als LDA). Im Tapioca Paper werden drei weitere BaseLines angeführt *etwa 5% pro BaseLine*
- Das MetadataExtraction-Tool kann nicht nur mit RDF-Dateien in verschiedenen Formaten als Eingabe arbeiten, sondern akzeptiert auch einen SPARQL-Endpoint als Eingabe, von welchem die benötigten Metadaten mittels Query angefragt werden. *10%*
- Statt die indexierten Daten bei Programmstart ins Programm zu laden könnte man eine Datenbank nutzen, welche die Themenvektoren speichert, um so über eine Query die nötigen Informationen zu erhalten. Dies ist besonders wichtig, da die Ladezeit bei 1680 Datensätzen zwar noch gering ist, diese Anzahl jedoch schnell wächst und so die Ladezeit des Programms verlängert. Die Gruppe möchte dies eher als Muss-Ziel betrachten, es ist nur noch unklar ob eine Datenbankstruktur für Vektoren existiert, die dies ermöglicht *10%*

Gesamtaufwand der Kann-Ziele: *60%*
Gesamtaufwand der Kann- und Muss-Ziele: *160%*

5 Vorprojekt

Das Vorprojekt soll im Wesentlichen die **Releasebündel 1 & 2** realisieren, somit ein gekapseltes Commandline Programm zur Metadaten-Extraktion und die Aufsetzung eines TomCat-Servers inklusive dem Design der Webseite als Resultat liefern. Dabei sollen erste Kommunikationen zwischen Webseite und Server bereits möglich sein.
Das Commandline Programm aus dem ersten Releasebündel führt eine Datenvorverarbeitung ins VoID-Format durch, und wird im Vorprojekt realisiert, da die VoiD-Daten später als Input für das Hauptprogramm benötigt werden.
Das Designen der Webseite und Aufsetzen des TomCat-Servers stellen sicher, dass die Kommunikation zwischen Webseite und Programm funktioniert, auch wenn die eigentlichen Funktionen noch nicht realisiert sind. So kann Problemen bei Anfragen der Webseite und Antworten des Servers vorgebeugt werden, sodass die spätere Realisierung der Funktionalitäten nicht durch eventuelle Fehler bei der Kommunikation zwischen Webseite und Programm behindert wird.

6 Glossar

- Semantik Web Formate: **RDF** ("Resource Description Framework") versucht Inhalte im Web miteinander zu verbinden und nach semantischen Zusammenhängen zu ordnen. **Turtle** ("Terse RDF Triple Language") ist Serialisierung für RDF-Graphen und ist verbreitet, weil es als benutzerfreundliche Alternative zu RDF/XML gilt. **JSON** ("JavaScript Object Notation") ist ein kompaktes Datenformat in einer einfach lesbaren Textform zum Datenaustauschs zwischen Anwendungen. **N-Triples** is a line-based, plain text format for representing the correct answers for parsing RDF/XML test cases. Die library "JENA" kümmert sich um die Lesbarkeit all dieser Formate und wird in diesem Projekt verwendet
- **VoID**("Vocabulary of Interlinked Datasets") ist ein auf RDF basiertes Schema um verlinkte Datensätze zu beschreiben
- **LOD**("Linked Open Data") bezeichnet im World Wide Web frei verfügbare Daten, die per URI identifiziert sind und so per HTTP abgerufen werden können und ebenfalls per URI auf andere Daten verweisen.
- **URI** ("uniform resource identifier") bestehen aus Zeichenfolgen und dient zur Identifizierung von Ressourcen. Die URI muss nicht zwangsweise im Netzwerk erreichbar sein (vgl. URL)
- Als **Goldstandard** bezeichnet man feste Referenzen, die miteinander verglichen werden, so dass die Konsistenz der Referenzstrukturen gewährleistet werden kann.
- **Benchmarking** Vergleichende Analyse von Ereignissen oder Prozessen auf Basis eines festen Bezugswertes.
- **F1-Score** Vergleichswert zum Goldstandard $\in [0, 1]$, mit "precision" (wie nah liegt Ergebnis am Goldstandard) und "recall" (wieviele vom Goldstandard errechnete Ähnlichkeiten wurden getroffen)
- **LDA**("Latent Dirichlet Allocation") ist ein generatives Topic Modell für Dokumente wie Textkorpora. Es dient dazu, einem oder mehreren Themen von Sätzen oder Wörtern eine Verteilung über Worte zuzuordnen.
- **Topic Model** bieten eine effiziente Möglichkeit, große Mengen von Text zu analysieren. Es gibt viele verschiedene Arten von Topic Models, in diesem Projekt wird die LDA verwendet.
- Ein **Framework** ist ein Programmiergerüst, welches in der Softwaretechnik, vor allem im Bereich der komponentenbasierten Entwicklung verwendet wird.
- **Apache lucene** ist ein Framework zur Volltextsuche und ein Projekt der Apache Software Foundation.
- **Inferencer** erstellt Relation zwischen Ressourcen im Semantik Web.
- **Webcrawler** Eine spezielle Art von Computerprogrammen, das weitgehend autonom das Internet nach bestimmten Daten durchsucht.
- Ein **SPARQL-Endpoint** ist ein SPARQL Protokoll Service und ermöglicht Benutzern (Mensch oder Maschine) eine Wissensbasis via SPARQL anzufragen.