Projektangebot

SWP 12-2

Colanut – LIMES

Zielsetzung

Die Zielsetzung des zu bearbeitenden Projekts ist die Erstellung einer Web-GUI für die bereits bestehende Applikation LIMES. Sie soll dazu dienen, das Arbeiten mit LIMES einfacher und übersichtlicher zu gestalten. Im Rahmen dessen war eine Auseinandersetzung mit mehreren verschiedenen Frameworks notwendig, wobei sich auf Grund seiner Vielseitigkeit das „Play! Framework“ als beste Option zur Umsetzung des Projekts herausgestellt hat.

Es wird bei der Entwicklung dieser GUI auf die Umsetzung eines User-Managements Wert gelegt. Weitere Schwertpunkte liegen auf dem Speichern und Bearbeiten von "LinkSpecs", die Verfügbarkeit der Ergebnisse, welche durch das Ausführen von „LinkSpecs“ erzeugt werden, als Datei, sowie das Einbinden der LIMES-Module zum active- und batchlearning der „LinkSpecs“.

Weitere wichtige Probleme sind die Erhaltung der Erweiterbarkeit, Modularität und Anpassungsfähigkeit der GUI. Dazu ist es nötig, das Gesamtprojekt in mehrere Teile zu Gliedern, welche getrennt voneinander entwickelt und implementiert werden können.

Folgend werden die größeren Arbeitspakete, in welche das Projekt untergliedert wird, und deren weiterhin zu bearbeitenden Teilaspekte (oder Problemstellungen) näher beleuchtet, sowie alle nötigen und optionalen Funktionen, betreffend dieser Teilaspekte, angeführt und auf deren Umsetzung eingegangen. Weiterhin wird geklärt, wie die Erweiterbarkeit, Modularität und Anpassungsfähigkeit aufrecht erhalten werden kann.

Als Milestones sollen folgende Punkte verwendet werden:

- Für die graphische Gestaltung die Sicherung der Übersichtlichkeit und der Funktionalität
- Für das User-Management die Möglichkeit, sich zu registrieren, einzuloggen und LinkSpecs zu speichern
- Für das Bearbeiten und Speichern der LinkSpecs die korrekte Funktion des GUIs
- Für das Bereitstellen der Matches als Datei die Möglichkeit, eben diese auf dem Computer abspeichern zu können
- Für das Learning die Optimierung und erzeugen von optimierten LinkSpecs

Gliederung

Das Gesamtprojekt soll in folgende Arbeitspakete unterteilt werden:

1. Grafische Gestaltung / Design
    1.1. Grafische Erstellung von „LinkSpecs“

2. User-Management
    2.1.Registrierte und anonyme User
    2.2.Rechteverwaltung
    2.3.Sichtbarkeit von selbst erstellten „LinkSpecs“ (öffentlich/privat)

3. Bearbeiten und Speichern von „LinkSpecs“
    3.1.Hochladen als z.B. CSV-Datei

4. Bereitstellung der gefundenen „Matches“ als Datei

5. Lernen von „LinkSpecs“
    5.1.Active-Learning
    5.2.Batch-Learning

1. Grafische Gestaltung / Design

Die Grafische Gestaltung der zu realisierenden Oberfläche ist ein sehr zentraler Aspekt des Projekts, da es sowohl für die Eingabe als auch für die Ausgabe der Daten wichtig ist, eine übersichtliche und nützliche Oberfläche zu gestalten.

Um dem Benutzer eine gewisse Flexibilität in der Bearbeitung der LinkSpecs zu gewährleisten, sollen die verschiedenen Arbeitsschritte in Tabs dargestellt werden. Dabei soll darauf geachtet werden, dass die Tabs übersichtlich gehalten werden.

Der erste dieser Tabs soll dabei für die Eingaben der Daten verwendet werden, die LIMES als Grundlage der LinkSpecs verwendet: Source Point und Target Point. Ebenso werden in diesem Tab die Parameter der zu erstellenden Config-Datei, wie z.B. Pagesize und Restrictions, eingegeben. Alternativ zum erstellen von neuen Config-Dateien soll auch eine Möglichkeit gegeben werden, bereits bestehende LinkSpecs oder Quellen in Form von CSV-Dateien zu laden, um diese dann bearbeiten zu können.

Ein weiterer Tab soll dann der eigentlichen Bearbeitung bzw. Erstellung der LinkSpecs dienen. In diesem Tab wird die grafische Oberfläche angezeigt, mit dem der Benutzer die LinkSpecs erstellt. Diese Oberfläche ist im Prinzip eine verbesserte Version der ColaNut Oberfläche. Ebenso werden weiterhin Source- und Target Point angezeigt, um dem Nutzer ein schnelleres Wiedereinfinden zu ermöglichen, sollte er eine Pause eingelegt haben.

Ein dritter Tab soll nun die gefundenen Matches zur Verfügung stellen. Diese Matches, welche durch die LinkSpecs fest vorgegeben sind, sollen dabei wahlweise angezeigt oder direkt gespeichert werden. Näheres zur Bereitstellung der Matches wird in Teil 4 behandelt.

Ebenso soll ein letzter Tab erstellt werden, welcher das Lernen von LinkSpecs ermöglichen soll. Dieses Tab wird in Teil 5 näher bestimmt.

Unabhängig von den Tabs soll die GUI auch eine Möglichkeit enthalten, sich einzuloggen, um LinkSpecs speichern zu können. Dies wird mithilfe typischer Textfelder realisiert, in denen die Nutzer ihre Login-Daten und das Passwort eingeben können, um jederzeit LinkSpecs laden, speichern oder erstellen zu können.

Bei der gesamten Implementierung der GUI soll darauf geachtet werden, dass das GUI sowohl modularisierbar als auch erweiterbar ist. Dies soll mithilfe übersichtlicher und gut dokumentierter Quelltexte realisiert werden, womit einem anderen Entwickler eine

schnelle Einarbeitung in eben diese Quelltexte ermöglicht werden soll. Ebenfalls sollen Methoden und Klassen so allgemein wie möglich gehalten werden.

2. User-Management

Das User-Management ist eines der größeren zu realisierenden Arbeitspakete. Es lässt sich wiederum in 3 Teilgebiete unterteilen.

Zum ersten die Möglichkeit sich als Benutzer zu registrieren und eigene „LinkSpecs“ zu erstellen bzw. bestehende zu bearbeiten, oder anonym vorgefertigte „LinkSpecs“ auszuführen.

Als zweites eine Rechteverwaltung zu entwickeln, um festzulegen, welche Funktionen und Möglichkeiten angemeldeten Usern vorbehalten sind und drittens selbst erstellte „LinkSpecs“ für alle User, ob registriert oder nicht, oder nur bestimmten User-Gruppen zugänglich zu machen.

Die ersten beiden Probleme bedingen sich in gewissem Maße gegenseitig und sie erhalten das Hauptaugenmerk bei dem User-Management. Das bedeutet sie werden als Notwendige Funktionen betrachtet. Die Sichtbarkeit der „LinkSpecs“ ist ein optionales Feature.

Umzusetzen ist zum einen die Rechteverwaltung durch das im „Play! Framework“ enthaltene „authorisation-module“ Deadbolt, in welchem man vorhandenen Rollen (anonym/registriert) per Java-Klassen Rechte zuweisen kann, um bestimmte Funktionen (per Annotation markiert) zu sperren, oder freizuschalten. Im Zusammenspiel mit dem Secure-modul zur Authentifizierung von Usern (welches ebenfalls über Annotationen arbeitet und eine Security-Klasse mit Passwort und Username Abfrage benötigt) , ist es möglich eine Log-in Routine zu erstellen, welche sicher stellt Funktionen wie zum Beispiel das erstellen von „LinkSpecs“ für anonyme User zu verbieten.

Um diese Log-in Routine erst zu ermöglichen wird allerdings noch die Möglichkeit benötigt sich registrieren zu können. Dies wird durch eine Java-Funktion ermöglicht, die nach Namen und zu erstellendem Passwort fragt und diese in einer Datenbank (vornehmlich Derby, durch die Plattformunabhängigkeit) ablegt. Dies Derby –DB kann über JDBC direkt mit Java-Code angesprochen werden. Diese Datenbank müsste allerdings schon vorgefertigt auf dem Server vorliegen, oder sie wird als Routine beim ersten Durchlauf der Applikation per Java-Code generiert. In dieser relationalen

Datenbank werden folgende Spalten zu finden sein: UserID (als Integer), Login (als String oder varchar), Hash (als String oder varchar welcher das Passwort enthält), dem „LinkSpec“ welcher als CLOB gespeichert wird und ein Access (als Bool oder alternativ, falls Benutzergruppen festgelegt werden, varchar, welcher die Sichtbarkeit des „LinkSpecs“ angibt). Durch diese Handhabung wird die erstellte Datenbank zwar groß im Umfang, aber bleibt dennoch mit den Nötigen Constraints sicher.

3. Bearbeiten und Speichern von „LinkSpecs“

Die „LinkSpecs“, ob hochgeladen oder selbst auf der grafischen Oberfläche erstellt, werden in der Datenbank als CLOB-Daten gespeichert. Dazu ist es nötig die XML Dateien in das CLOB-Format zu konvertieren. Wichtig dabei ist, dass die Default-Größe des CLOB-Feldes 2 GB beträgt, falls keine spezielle Größe angegeben ist.

Um die Daten zu konvertieren ist es möglich „java.sql.clob“ (aus JDBC 2.0) zu nutzen und dann per StreamReader (oder ähnlichem) die XML einzulesen und zu speichern. Für das bearbeiten der als CLOB vorliegenden Datei, ist es nötig ein encoding mit dem festgelegtem Standard „ISO-8859-1“ durch zuführen, was durch eine Java-Klasse mit den nötigen Readern und Writern ermöglicht werden kann. Das Hochladen von „LinkSpecs“ ist eine Optionale Funktion. Die „LinkSpecs“, welche in verschiedenen Dateiformaten vorliegen können, müssen, um gespeichert oder überhaupt ausgeführt werden zu können, in das XML-Format konvertiert werden.

Da es mannigfaltige Möglichkeiten gibt, in welchem Format die Rohdaten vorliegen, muss man in Anbetracht des Umfangs sich vorläufig auf wenige Formate beschränken und weitere Konverter später als Plug-ins hinzu zufügen. Ein wichtiges Format ist in diesem Fall das CSV-Format, auf welchem das Augenmerk des Konverter Entwurfs liegt. Um diese Umwandlung vorzunehmen ist es nötig OpenCSV und XStream (beides externe Java-Bibliotheken) zu nutzen um zum einen die CSV zu parsen und zum anderen die XML zu parsen/serialisieren. Die CSV wird so von einem CSVReader eingelesen, konvertiert und von einem XStream als XML geschrieben, wobei durch XStream eine „Feinabstimmung“ des fertigen Files vorgenommen werden kann.

Der Upload einer Datei an sich kann durch die Java-Bibliothek „FileUpload“ von Apache

realisiert werden. Diese Bibliothek stellt Möglichkeiten bereit um eine „RFC 1867“ Anfrage zu bearbeiten. Die direkte Bearbeitung der „LinkSpecs“ geschieht innerhalb der grafischen Oberfläche, welche unter Punkt 1 näher erläutert wird. Dazu müssen die XML Daten wieder aufbereitet werden, damit sie richtig in der Oberfläche dargestellt werden können. Um dies nun zu ermöglichen kann der DOM-parser verwendet werden, da dort die Möglichkeit geboten wird, die eingelesenen Daten auch zu verändern. Es hängt allerdings von der Größe der XML-Files ab, welcher Parser bzw. Parser-API verwendet werden kann.

4. Bereitstellen von gefundenen „Matches als Datei

Die Ausgabe von Limes sind die gefundenen Matches, welche in der Datei „accepted.nt“ gespeichert werden. Diese Datei wird durch den LinkSpec fest vorgegeben. Bei selbst erstellten LinkSpecs (z.B. nicht durch das WebGUI) wird der <File></File>-Part der Acceptance Condition und der Review Condition mit der geforderten Dateibezeichnung überschrieben.

Im Kopfteil der accepted.nt-Datei befinden sich die genutzten Präfixe und im Hauptteil die gefundenen Matches. Dabei wird der durch Limes vorgegebene strukturelle Aufbau der Matches nicht verändert. Somit werden sie in folgender Form gesichert:

<URI-Link_Source> Relation <URI-Link_Target>

Der Benutzer hat nun die Möglichkeit diese Datei auf dem eigenen Rechner abzuspeichern oder sie sich im Browser anzusehen. Eine öffentliche oder private Speicherung dieser Daten in einer Datenbank ist im jetzigen Verlauf des Projektes nicht vorgesehen, kann jedoch in einer später erfolgenden Erweiterung implementiert werden.

Des weiteren wird eine „review.nt“-Datei erzeugt, welche neben den verwendeten Präfixen auch ein Beispiel-Match enthält. Der Inhalt dieser Datei wird dem Benutzer als Feedback/Ergebnis vorgestellt. Die Präsentation findet als abschließender Vorgang der Limesausführung statt und soll dem Benutzer die Möglichkeit geben das Ergebnis des Instanz-Linkings zu sehen und gegebenenfalls zu überprüfen. Bei Unstimmigkeiten kann das Ergebnis des LinkSpecs mittels verschiedener Learning-Feature verbessert werden. Dieses wird in Teil 5 näher bestimmt.

5. Lernen von LinkSpecs:

Allgemein:

Das Lernen von LinkSpecs erfolgt über einen eigenen Tab in der WebGUI. Diese Funktionalität steht nur registrierten Nutzern zur Verfügung, keinen Gästen. Am Anfang hat der Nutzer die Möglichkeit sich für eine Learning-Methode zu entscheiden (Active oder Batch Learning). Die Auswahl erfolgt dabei über Radio-Button.

Unabhängig von der genutzten Methode wird der Benutzer anschließend aufgefordert den Ausgangs-LinkSpec anzugeben. Diese kann entweder selbst hochgeladen werden oder man nimmt eine abgespeicherte aus der (>Bearbeiten und Speichern von LinkSpecs). Wie bei der graphischen Oberfläche wird auch hier viel Wert auf die Erweiterbarkeit und Modularisierung gelegt. Dabei wird versucht Funktionen so allgemein wie möglich zu halten um die Hinzunahme weiterer Lernmethoden zu ermöglichen.

5.1 Active Learning

Beim Active Learning kann der Nutzer die Anzahl der Iterationen über ein Textfeld angeben, wobei verschiedene Voreinstellungen möglich sind. Des weiteren kann er noch die Anzahl der Matches angeben, die nach jedem Durchlauf zu überprüfen sind. Nach der Übergabe des LinkSpecs erscheint eine erste Liste von Matches zwischen Instanzen aus Source und Target. Der Nutzer kann nun über Radiobuttons bei jedem potentiellen Match auswählen, ob es sich dabei um ein Match handelt oder nicht (match/non-match). Dieser Vorgang wird der Anzahl der Iterationen entsprechend wiederholt.

Als Ergebnis wird dem Nutzer der angepasste LinkSpec (XML-Datei) übergeben, welche er lokal oder auf dem Server (privat oder auch öffentlich) speichern kann (>Bearbeiten und Speichern von LinkSpecs). Exakteres zur Implementierung des Benutzeroberfläche für das Active Learning erfolgt später, sobald die genauen Schnittstellen feststehen.

5.2 Batch Learning

Ähnlich dem Active Learning kann der Nutzer hier die Anzahl der Matches angeben, die er zu überprüfen wünscht. Jedoch wird eine Mindestanzahl vorausgesetzt (Ein optimaler Wert wird noch festgelegt). Da das Batch Learning nicht auf Iterationen beruht, müssen keine weiteren Schritte beachtet werden. Auch hier erhält der Nutzer nach Abschluss des Lernens den optimierten LinkSpec, den er wiederum lokal oder in der Datenbank speichern kann, wie in Teil 3 beschrieben.