MusicMashup Recommender

ein Linked Open Data Musik-Empfehlungsdienst

Seminararbeit im Seminar

Linked Data Mashup Programmierung

Wintersemester 2014/15

Hasso-Plattner-Institut für Softwaresystemtechnik GmbH

Universität Potsdam

vorgelegt von

Dennis Hempfing
Paul Wille
Karl Wolf

28. Februar 2015

Kurzzusammenfassung

Diese Arbeit gibt einen Einblick in die Funktionsweise des von uns entwickelten Musik-Empfehlungsdienstes MusicMashup, der auf Linked Open Data basiert, und beleuchtet weiterhin die Motivation hinter der Arbeit an diesem System. Im Rahmen des Seminars "Linked Data Mashup Programmierung" bei Dr. Harald Sack (Hasso-Plattner-Institut) haben wir es uns zur Aufgabe gemacht zu zeigen, wie sich unter Nutzung von Linked Data ein vollwertiger Musik-Recommender entwickeln lässt und welche Vor- und Nachteile dadurch entstehen. Dabei stand neben Linked Data die Fragestellung im Mittelpunkt, wie sich das System im Vergleich zu anderen Musik-Recommendern abhebt und welchen Mehrwert es dadurch liefert.

Inhaltsverzeichnis

1 Einleitung

Das Internet ist eine Quelle für alle Arten von Daten – seien es Enzyklopädien wie die Wikipedia oder spezielle Datenbanken für Filme, Bücher, Forschungsergebnisse und viele weitere.
Versucht man nun ein Mashup zu entwickeln, welches verschiedene Datenquellen nutzt, entstehen oftmals Probleme.

- die verwendeten Datenquellen verwenden meistens unterschiedliche Formate. Das heißt, dass vor der Verwendung der Daten, diese auf ein gemeinsames Format geführt werden müssen.
- Auch die Abfrageweise der Daten ist nicht einheitlich. Wird zum Beispiel mit einer SQL-Datenbank und einer API gearbeitet, sind für beide Datenquellen jeweils unterschiedliche Abfrageweise notwendig.
- Die meisten Datenquellen enthalten keine Verweise auf andere thematisch relevanten Datenquellen. Somit ist ein zusätzliches Maß an Recherche erforderlich, um ausgehend von einer Datenquelle eine weitere zu finden.
- Viele Datenquellen werden privat verwaltet und sind nicht öffentlich und somit auch nicht frei nutzbar. Solche Datenquellen können nicht verwendet werden und stellen für die Öffentlichkeit keinen Mehrwert dar.

Um diese Probleme zu lösen wurde das Prinzip von Linked Open Data entwickelt[b][c].

In Kapitel 2 wird zunächst ein kurzer Überblick über die Idee hinter Linked (Open) Data gegeben und die Funktionsweise grob skizziert. Darüber hinaus wird die Funktionsweise anderer Musik-Recommender erklärt und ein Vergleich zu unserem Ansatz gezogen.

In Kapitel 3 wird der von uns verfolgte Ansatz im Detail erläutert und sowohl von konzeptioneller als auch von technischer Seite beleuchtet.

In Kapitel 4 kommt es zur Darstellung und Diskussion der erzielten Ergebnisse und es wird auf die Stärken und Schwächen des Systems eingegangen.

Kapitel 5 ist eine Zusammenfassung der erzielten Ergebnisse und Erkenntnisse. Es folgt ein Ausblick auf mögliche Verbesserungen und Erweiterungen des Systems.

2 Related Work

2.1 Linked Data

Der Begriff "Linked Data" beschreibt eine Menge an Prinzipien, wie Daten im Internet strukturiert, publiziert und verlinkt werden sollten. Entwickelt worden sind diese von Tim Berners-Lee, der sie 2006 in seinem Bericht "Linked Data" (Q1) veröffentlichte. Die Grundidee dabei ist, sich dafür die Architektur des Internets zunutze zu machen. Im Wesentlichen handelt es sich dabei um vier Punkte:

1. Verwende zur Bezeichnung von Objekten URIs.
2. Verwende HTTP-URIs, so dass sich die Bezeichnungen nachschlagen lassen.
3. Stelle zweckdienliche Informationen bereit, wenn jemand eine URI nachschlägt (mittels der Standards RDF und SPARQL).
4. Zu diesen Informationen gehören insbesondere Links auf andere URIs, über die weitere Objekte entdeckt werden können.

Das erste Prinzip besagt, dass URIs nicht nur benutzt werden sollen, um Dokumente im Internet verfügbar zu machen. Jede Form von Objekten oder Entitäten soll eindeutig über eine URI identifizierbar sein, handele es sich um eine Person, einen spezifischen Gegenstand oder allgemein "Dinge".
Das zweite Prinzip spezifiert, dass die im ersten Prinzip angesprochenen URIs HTTP-URIs sein sollen. HTTP hat sich im Internet als universales Übertragungsprotokoll etabliert. Die Verwendung von HTTP-URIs ermöglicht es demzufolge, solche URIs einfach nachschlagbar zu machen.
Das dritte Prinzip schlägt die Benutzung eines einheitlichen Formats für die Publikation von Daten vor, in diesem Fall das "Resource Description Framework" (RDF). Dabei handelt es sich um ein simples graphen-basiertes Datenmodell. Bei SPARQL handelt es sich um eine von SQL inspirierte RDF-Anfragesprache.
Das vierte Prinzip plädiert für die Benutzung von Hyperlinks zur Verlinkung einzelner Ressourcen, um den Zugriff auf weitere thematisch relevante Datenquellen zu ermöglichen. Im Gegensatz zu normalen Hyperlinks, welche keinen Kontext beinhalten, lässt sich bei nach RDF spezifizierten Links ein Kontext angeben. So kann zum Beispiel ausgedrückt werden, dass zwei Ressourcen semantisch äquivalent sind oder dass zwei Personen, die miteinander verlinkt werden, befreundet sind.

2.2 RDF

Wie bereits beschrieben, handelt es sich bei dem "Resource Description Framework" um ein Graph-basiertes Datenmodell zur Strukturierung von Daten. Die Daten werden dabei in Tripeln organisiert und bilden einen gerichteten Graphen. Ein Tripel besteht aus Subjekt, Prädikat und Objekt und bildet die Struktur eines simplen Satzes ab. Solch ein simpler Satz könnte wie folgt aussehen:

```
Led Zeppelin (S) ist (P) eine Band (O).
```

Das Subjekt eines Tripels ist eine URI, die die Ressource eindeutig identifiziert. Beim Objekt kann es sich um ein Literal, zum Beispiel eine Nummer oder einen String, oder um eine URI handeln, die mit dem Subjekt in Verbindung steht. Das Prädikat spezifiziert in welcher Form Subjekt und Objekt miteinander in Verbindung stehen. Thematisch zusammenhängende Prädikate werden in so genannten Ontologien spezifiziert, wobei auch jedes Prädikat über eine URI dereferenzierbar ist.
Der Satz aus dem obigen Beispiel ("Led Zeppelin" ist vom "Typ" "Band") wird in der DBPedia folgendermaßen dargestellt:

http://dbpedia.org/resource/Led_Zeppelin (S)
http://www.w3.org/1999/02/22-rdf-syntax-ns#type (P)
http://dbpedia.org/ontology/Band (O)

Hierbei ist noch zu erwähnen, dass URLs, vor allem solche, die öfter vorkommen, im rdf-Format durch präfixe ersetzt werden können. http://www.w3.org/1999/02/22-rdf-syntax-ns# wird dann zum Beispiel zum Präfix "rdf:"; Ressourcen aus der DbPedia können durch "dbpedia:" dargestellt werden. Das genannte Beispiel sieht dann derart aus:

dbpedia:Led_Zeppelin (S) rdf:type (P) dbpedia-owl:Band (O)

In dieser Ausarbeitung werden wir der Lesbarkeit halber ebenfalls gängige Präfixe für Ontologien benutzen. Diese sind:

- dbpprop: `<http://dbpedia.org/property/>`
- `dbpedia-owl: <http://dbpedia.org/ontology/>`
- `rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>`
- `rdfs:<http://www.w3.org/2000/01/rdf-schema#>`
- `mo: <http://purl.org/ontology/mo/>`
- `foaf: <http://xmlns.com/foaf/0.1/>`
- `owl: <http://www.w3.org/2002/07/owl#>`

2.3 DBpedia

Bei der DBpedia handelt es sich um ein offenes Projekt, dessen Ziel es ist, die in der Wikipedia zusammengetragenen Daten nach der RDF Spezifikation zu strukturieren und unter offener Lizenz zu veröffentlichen (Q4). Sie besteht seit 2007 und ist eines der größten Projekte, das sich mit Linked Data befasst. Aufgrund ihrer Größe, des breiten Spektrums an Informationen, welche abgedeckt werden, und der Anzahl der Verlinkungen zu anderen Datensätzen, bildet die DBpedia einen großen Knotenpunkt im Web of Data und ist demzufolge eine gute Grundlage für die Erstellung eines Linked Data Mashups.

2.4 Existierende Empfehlungsdienste

Seitdem Musik gemacht und verkauft wird, besteht auch immer ein Bedarf danach neue Musik zu entdecken, die einem gefällt. Was früher nur durch das Gespräch mit Freunden oder im Plattenladen möglich war, wird heute auch automatisiert im Internet angeboten.

Man unterscheidet dabei zwischen **inhaltsbasierten** und kollaborativen Empfehlungsdiensten. Bei ersteren steht die Musik selbst im Mittelpunkt; ausgehend von einem Musikstück oder einer Band wird bestimmt, welche anderen Bands und Liedern diesem ähnlich sind, zum Beispiel durch ein ähnliches Genre oder gemeinsame Mitglieder.

Kollaborative Empfehlungsdienste dagegen betrachten nicht die Musik als solche, sondern arbeiten mit Statistiken auf Basis der Interaktionen ihrer Nutzer. Ein einfaches Beispiel stellt gnoosic[1] dar, das nach der Eingabe eines Interpreten weitere vorschlägt, für die man jeweils angeben kann, ob man diese ebenfalls mag.
Der Nutzen dieser Dienste entsteht durch die große Menge an nutzergenerierten Informationen. Durch die Auswertung des Geschmacks vieler Leute lässt sich meist eine präzise Aussage treffen, ob der Hörer von Band A auch Band B schätzt.

Daneben gibt es auch **nicht-automatische Varianten** der Musikempfehlung, zum Beispiel durch Freunde, in Musikgeschäften oder Zeitschriften.

Viele Empfehlungsdienste sind direkt in andere Dienste integriert, so zum Beispiel beim Online-Händler Amazon oder dem Musikstreamingportal Spotify. Diese werten direkt das Kauf- bzw. Hörverhalten ihrer Kunden aus, um damit Empfehlungen zu generieren.
Außerdem existieren spezialisierte Empfehlungsdienste, so zum Beispiel der BibTip-Service des Karlsruher Instituts für Technologie[2] oder Webseiten wie TasteKid[3] und Musicovery[4]. Wiederum andere Dienste richten sich an spezielle Anwendungszwecke wie Jog.fm[5], dass je nach Laufgeschwindigkeit die musikalische Untermalung zum Joggen empfiehlt.
Neben den genannten Beispielen lassen sich für jede Kategorie weitere Anbieter finden, auf die jedoch nicht näher eingegangen werden soll.

Im **Vergleich** zum vorgestellten Ansatz existieren zwei signifikante Unterschiede. Zum ersten gibt keines der anderen Musikempfehlungssysteme Gründe an, warum bestimmte Interpreten empfohlen werden. Zum zweiten verfolgen die anderen Systeme jeweils einen Ansatz gezielt, haben also zum Beispiel Musik, aber keine Liste anstehender Konzerte oder einen Abstract über den Interpreten, aber keine Möglichkeit Musik abzuspielen. MusicMashup gibt Gründe für die Empfehlungen ab und bietet neben der Empfehlungslogik auch weitere Funktionalität an. Wie dies im Detail geschieht, wird im nächsten Kapitel genauer beschrieben.

[1] `http://www.gnoosic.com/`
[2] `http://www.bibtip.com/`
[3] `http://www.tastekid.com/`
[4] `http://musicovery.com/`
[5] `https://jog.fm/`

3 Aufgabenstellung

Wir haben uns zum Ziel gesetzt, einen vollwertigen Musikempfehlungsdienst zu entwickeln, welcher auf Basis von Linked Data arbeitet. Inspiriert ist dies durch den Internetdienst Musicbrainz (Q5). Dabei handelt es sich um eine offene Musik-Enzyklopädie, die zahlreiche Informationen zu Musikern und Bands anbietet. Dazu zählen Informationen wie ein kurzer Abstract, eine Discographie, externe Links, eine Liste aller aktueller und ehemaliger Mitglieder der Band etc. Sucht man auf Musicbrainz zum Beispiel nach der Band "Queens of the Stone Age", sieht man dort, dass Josh Homme das einzige verbliebene Gründungsmitglied der Band ist. Mit einem Klick auf den Namen wird man zur Seite von Josh Homme weitergeleitet. Dort sieht man unter Anderem in welchen anderen Bands er gespielt hat und kann sich wiederum Informationen zu diesen ansehen.

Der Benutzer kann also ausgehend von einem Interpreten weitere entdecken. Genau diese Möglichkeit soll auch MusicMashup bieten. Der Benutzer gibt den Names eines Interpreten ein und bekommt passend zu diesem Empfehlungen. Klickt er eine dieser Empfehlungen an, werden ihm wiederum für diesen Interpret Empfehlungen angezeigt.

Erweitert wird dies durch zusätzlichen Informationen und Funktionen. Die wohl wichtigste Zusatzfunktion ist ein eingebundener Spotify-Player, welcher die Möglichkeit bietet, sich direkt Musik des jeweiligen Künstlers anzuhören. Darüber hinaus wird eine Liste mit anstehenden Konzerten ausgegeben, sowie weiterführende Links, eine Bildergalerie und ein kurzer Abstract über den Artist. Damit der Benutzer nachvollziehen kann, wie er von einem Interpreten zum Nächsten gekommen ist, wird im Header der Seite eine Liste der zuvor betrachteten Interpreten angezeigt.

3.1 Technische Grundlagen des Servers

MusicMashup ist als so genannte Web-App realisiert und präsentiert sich dem Benutzer als interaktive Website, die er aus seinem Browser nutzen kann.
Die programmatische Grundlage von MusicMashup bildet ein in Python geschriebener HTTP-Server basierend auf dem cherrypy-Modul (Q6). Dieser Server wertet die Nutzeranfrage aus, führt die entsprechenden Abfragen durch und liefert die daraus generierten Empfehlungen mittels der Template-Engine mako (Q7) als Website aus.

3.2 Klassenstruktur

Das System besteht aus vier Klassen: *MusicMashupServer*, *MusicMashupArtist*, *MusicMashupParser* und *MusicMashupPagerank* (siehe Abb. x).

Server Die Klasse *MusicMashupServer* stellt den Web-Server bereit und löst bei einem Aufruf der Anwendung weitere Funktionen aus. Wird nach einem Interpreten gesucht, wird für diesen Interpreten ein Objekt vom Typ MusicMashupArtist erstellt. Diesem wird entweder die Suchanfrage oder aber eine aus einer Empfehlung stammende DBpedia-URL übergeben, sollte eine Empfehlung angeklickt worden sein. Die eigentliche Programmlogik

geschieht dann in diesem Objekt. Nachdem diese ausgeführt wurde, wird anschließend das HTML-Dokument gerendert und an den Browser ausgeliefert.

Artist Die Klasse *MusicMashupArtist* dient als Repräsentation eines Artists. Sie implementiert alle Queries, mittels derer Informationen über einen Artist gefunden werden und speichert diese. Darüber hinaus stellt sie Funktionen bereit, die es dem HTML-Template ermöglichen, diese Informationen abzufragen, um sie anschließend anzeigen zu lassen.

Parser Die Klasse *MusicMashupParser* ist eine simple Implementierung eines Parser, der die Informationen, die wir zu den Artists finden, in eine Turtle-Datei (.ttl) schreibt. Bei Turtle handelt es sich um eine Syntax des rdf-Formats, welche eine Repräsentation eines Rdf-Graphen in Textform ermöglicht (Q15). Das Parsen geschieht aus zwei Gründen: erstens liegt nicht jede der Datenquellen in Form von Linked Data vor. Der Parser überführt die Informationenin ein solches Turtle-FIle und ermöglicht so eine Bereitstellung der Daten als Linked Data. Zweitens kann beim Aufruf eines Artists überprüft werden, ob bereits eine Turtle-Datei existiert, in dem die Informationen bereits zu Verfügung stehen. Falls ja, können diese aus der Turtle-Datei geladen werden, was zu einer deutlichen Verringerung der Ladezeit führt.

Pagerank Die Klasse *MusicMashupPagerank* wird einmalig beim Aufruf der Anwendung instanziiert und dient als Schnittstelle zum Dump der DBPedia-Pageranks, welche in einen Graphen geladen werden und somit zum Abfragen bereitstehen.

3.3 Datensätze

Musicbrainz Musicbrainz verwendet IDs, um Artists eindeutig zu identifizieren. Diese IDs wurden von vielen anderen Musikdiensten zur Identifikation von Artists übernommen. Um diese zu erhalten wird eine Anfrage auf DBTune (siehe Q12 Alle anderen Datenquellen brauchen auch noch n Link/Quelle)[d] gestellt. Dabei handelt es sich um ein Linked Data Mapping der Musicbrainz Datenbank. Bei einer erfolgreichen Abfrage erhält das System die MusicbrainzID und über owl:sameAs den Link zur entsprechenden DBPedia Ressource. Da DBTune jedoch nicht vollständig ist (zum Beispiel enthält DBTune keinen Eintrag für die Band „Them Crooked Vultures"), wird als Fallback ein Linked Data Dump der Musicbrainz Datenbank benutzt. Zwar gibt es in vielen Fällen, in denen es keinen Eintrag auf DBTune gibt, einen Eintrag im Musicbrainz Dump, jedoch hält der Dump keinen Verweis auf DBPedia, weswegen er schlussendlich nur als Fallback zum Einsatz kommt.

DBpedia Als Hauptdatenquelle kommt DBPedia zum Einsatz. Von der DBPedia erhält unser System zu einem Interpreten den Abstract, eine Liste der aktuellen und ehemaligen Bandmitglieder sowie ein Thumbnail.[e][f] Um die bereits vorher genannten zusätzlichen Informationen und Funktionen bereitzustellen, musste auf Datenquellen zurückgegriffen werden, die nicht im Linked Data Format vorliegen. Dazu gehören Spotify (Q8) für den Musikplayer, Songkick (Q9) für zukünftige Konzerte und Echonest (Q10). Echonest

dient einerseits als Schnittstelle zu den anderen APIs, andererseits liefert es auch einen sogenannten "Familiarity"-Wert, welcher benutzt wird, um die Relevanz der gefunden Empfehlungen zu bewerten (dazu später mehr).

Commons Weiterhin kommen die DBPedia-Commons (siehe `http://commons.dbpedia.org/`) sowie ein Dump der DBPedia-Pageranks zum Einsatz. DBPedia-Commons stellt die Wikimedia-Commons als Linked Data bereit. Diese werden benutzt, um zu einer Band Bilder in einer Bildergallerie bereit zu stellen, sollten welche vorliegen. Der Dump der DBPedia-Pageranks, welcher von der Semantic-Technologies-Forschungsgruppe des Hasso-Plattner-Instituts bereitgestellt wurde, hält für jede DBPedia-Ressource einen Eintrag über ihren Pagerank. MusicMashup verwendet diesen Dump, reduziert auf Ressourcen mit dem rdf:type Band, MusicalArtist bzw. Artist.

3.4 Funktionsweise

Beim erstmaligen Aufruf von MusicMashup wird dem Benutzer eine einfach gestaltete Startseite präsentiert, auf der mittels eines Eingabefelds nach einem Artist (siehe Glossar) suchen kann, zu dem er sich Empfehlungen geben lassen möchte.

3.4.1 Finden von Informationen zum Künstler

Der vom Benutzer eingegeben Input wird zunächst in Titlecase überführt. Das bedeutet, dass alle Wortanfangsbuchstaben der Eingabe in Großbuchstaben überführt werden, außer es handelt sich um Artikel oder Präpositionen. Wir haben bei unserer Arbeit feststellen können, dass sich somit mehr als 95% aller Künstler zuverlässig finden lassen. Anschließend wird auf DBTune nach einer Ressource gesucht, deren `rdfs:label` mit der Eingabe des Nutzers übereinstimmt und die außerdem als rdf:type den Eintrag `mo:MusicArtist` hat. Wird eine Ressource gefunden, erhält das System die Musicbrainz ID des Artists sowie den Link zur DBPedia Ressource.
Für den Fall, dass sich ein Artist nicht auf DBTune finden lässt, wird im Musicbrainz-Dump nach dem Artist gesucht. Statt `rdfs:label` kommt hier `foaf:name` zum Einsatz. Da im Musicbrainz-Dump kein Verweis auf den entsprechenden DBPedia-Eintrag vorhanden ist, muss auf der DBPedia selbst nach der entsprechenden Ressource gesucht werden.

3.4.2 Suche nach Recommendations

Ist der Link zur DBPedia Ressource des jeweiligen Künstlers vorhanden, kann gezielt nach Recommendations gesucht werden. Dazu werden zunächst alle aktuellen sowie ehemaligen Mitglieder der Band via `dbpprop:currentMembers` oder `dbpedia-owl:bandMembers` bzw. `dbpprop:pastMembers` oder `dbpedia-owl:formerBandMember` abgefragt. Im Anschluss werden für jedes so gefundene Mitglied folgende Relationen überprüft: die Membership-Relation (s.o), die Producer-Relation (`dbprop:producer` oder `dbpedia-owl:producer`), die Composer-Relation (`dbpedia-owl:composer`) und die Writer-Relation (`dbpedia-owl:writer` oder `dbpprop:writer`). Grundsätzlich wird für jede Relation nach Bands oder Künstlern

gesucht, in denen ein aktuelles oder ehemaliges Mitglied der Band zur Zeit spielt, gespielt hat bzw. nach Bands, für die ein aktuelles oder ehemaliges Mitglied der Band produziert oder einen Song geschrieben bzw. komponiert hat.

3.4.3 Auswerten der Recommendations

Wird bei einer dieser Abfragen eine Recommendation gefunden, gibt es zwei Möglichkeiten: ist der gefundene Artist noch nicht in der Liste der recommended Artists enthalten, wird er der Liste samt des Grundes, warum er empfohlen wird, hinzugefügt. Sollte der gefundene Interpret bereits in der Liste enthalten sein, wird diesem Artist lediglich der Grund hinzugefügt, weshalb er gefunden wurde.
Nachdem alle Relationen überprüft wurden, wird anhand eines Voting-Algorithmusses über die Relevanz der einzelnen Recommendations entschieden. Dafür wird der Familiarity-Wert genutzt, den die echoNest-API bereit stellt, sowie die dbpedia-Pageranks. Die echoNest-Familiarity ist ein von echoNest errechneter Wert, der aussagt, wie bekannt die Band ist.

Darüber hinaus wird jeder Relation ein Faktor zugeordnet, der in die Berechnung mit einfließt. Diese Faktoren haben die Aufgabe, die Wichtigkeit der Relationen untereinander zu bewerten. Die Membership-Relation für aktuelle Bands hat den höchsten Faktor, gefolgt von der Membership-Relation für ehemalige Bands, der Producer-Relation und der Writer- und der Composer-Relation, welche gleichwertig sind. Geht diese Relation von einem aktuellen Mitglied der Band aus, so ist der Faktor verdoppelt so hoch wie für ein ehemaliges Mitglied. Am Ende werden der dbpedia-Pagerank, die echonest-Familiarity und der Relationsfaktor miteinander multipliziert. Die Recommendation mit dem höchsten Wert steht am Ende in der Liste der Vorschläge oben und die mit dem niedrigsten Wert unten. Nach Abschluss des Votings wird der Parser gestartet, welcher alle gefundenen Informationen für einen Artist in eine Turtle-Datei schreibt.

Wurden alle Schritte erledigt, kann die Seite gerendert und ausgegeben werden (siehe Abb. x).

4 Diskussion der erzielten Ergebnisse

Durch den im vorherigen Absatz beschriebenen Ansatz ist es gelungen, mit dem MusicMashup-Recommender einen vollwertigen Recommender-Dienst vorweisen zu können. Da es im Internet einige Music-Recommender-Dienste gibt (vgl. das Kapitel Related Work), stellt sich die Frage, inwiefern sich dieses Produkt von anderen abhebt. Zumal neben dedizierten Empfehlungsdiensten auch viele andere Dienste (Spotify, Grooveshark, Youtube, Amazon u.v.m.) eine Empfehlungslogik eingebaut haben.
Ein wichtiges, wenn nicht das wichtigste Alleinstellungsmerkmal, ist der Ansatz, eine Nachvollziehbarkeit für den Nutzer, aus welchen Gründen Empfehlungen zu einem Interpret gefunden wurden, bereitzustellen. Dies kann außer durch manuelle Suche in großen Musikdatenbanken wie beispielsweise discogs so nicht mit anderen Recommendation-Diensten erreicht werden.
Im Falle von Musik zielen Recommendation-Dienste meistens darauf ab, dem Nutzer "ähnliche" Musik vorzuschlagen und ihm dadurch zu zeigen, welche Interpreten zu dem gesuchten Interpret oft in Verbindung gebracht werden, beziehungsweise bei welchen Interpreten sich die Geschmäcker einer Vielzahl von Hörern überschneiden. Der Ansatz des MusicMashup-Recommenders ist da deutlich pragmatischer, da direkte Verknüpfungen der Mitglieder einer Band oder eines Solo-Artisten als Grund dienen, einen Interpreten vorzuschlagen. Das Ziel ist es aber auch, mit dem MusicMashup-Recommender eine Zielgruppe zu erreichen, die eher als musikaffin zu bezeichnen ist und sich für die Hintergründe interessiert und nicht einfach nur neue ähnliche Musik entdecken will. Deshalb wurde bei der Entwicklung darauf Wert gelegt, dem Nutzer die Gründe für einen Vorschlag gut sichtbar und verständlich anzuzeigen.

4.1 Bedienung

Neben der Art der Vorschläge ist ein weiterer Ansatz [g]des MusicMashup-Recommenders, die Bedienung so dynamisch[h][i] wie möglich zu gestalten. So ist es möglich, auf alle vorgeschlagenen Interpreten aber auch Bandmitglieder zu klicken, um wiederum Vorschläge für diesen Interpreten zu erhalten. Es werden also nicht nur für eine getätigte Eingabe Empfehlungen ausgegeben, sondern man erlangt durch Anklicken eben dieser Empfehlungen erneut Informationen zu diesen. Dies soll den Nutzer dazu einladen, unbekannte Zusammenhänge zu entdecken und das Wirken von Bands oder Solokünstlern, die ihn interessieren anhand von Gründen nachzuvollziehen. Durch die prominent platzierte Anzeige eines Pfades, der Breadcrumb-ähnlich dargestellt wird, soll ermöglicht werden, dass der Nutzer auch nach längerem Entdecken noch die Möglichkeit hat, nachzuvollziehen, wie er zu dem aktuellen Interpreten gelangt ist und gegebenenfalls zu einem Artist des Pfades zurückspringen kann, sollte er sich nicht weiter in die eingeschlagene Richtung vertiefen wollen.
Da dieses Produkt sich mit Musik beschäftigt, stand der Anspruch immer im Vordergrund, dem Nutzer direkt das Hören der Musik eines Interpreten zu ermöglichen. Nach längerer Evaluation viel die Wahl auf Spotify als Anbieter. Dies schränkt zwar die Zielgruppe ein, da ein Account erforderlich ist, allerdings ist Spotify einer, wenn nicht der etablierte Musikdienst zum kostenlosen Musikhören. Außerdem bietet Spotify zu fast je-

dem Künstler eine große Auswahl an Musik. Entscheidend für die Wahl von Spotify war vor allen Dingen auch, dass es trotz einfacher Einbindung in die UI einen vom Browser entkoppelten Weg ermöglicht, Musik abzuspielen. Dies bietet den entscheidenden Vorteil, dass Musik im Hintegrund weiterläuft, auch wenn im Browser neue Künstler angeklickt werden. Eine solche Verknüpfung des tatsächliche Hörens von vorgeschlagenen Künstlerns im Zusammenspiel mit der ständigen Möglichkeit parallel dazu weitere Relationen zu entdecken, ist in einer solchen Form schwer zu finden.

4.2 Zusätzliche Informationen

Zusätzlich zu den Vorschlägen, der Bedienung und der ständigen Möglichkeit, die Künstler auch anzuhören, zeichnet sich der MusicMashup-Recommender dadurch aus, dass er dem Nutzer Zusätzliche Informationen übermittelt, die interessant sind, sobald er sich zu einem Artisten genauer informieren will. Eine Liste der aktuellen und ehemaligen Bandmitglieder, welche auch einzeln anklickbar sind, sodass das System gleichermaßen für Bands, wie auch für Bandmitglieder und Solokünstler funktioniert, findet man bei anderen Music-Recommendern selten in dieser Art, was MusicMashup durchaus von anderen Empfehlungsdiensten abhebt. Neben dem in fast allen Diensten angezeigten Abstract hat der Nutzer mit dem MusicMashup-Recommender die Möglichkeit bei Bedarf durch Links zu anderen Musikrelatierten Diensten weitere Informationen über den Interpret zu erlangen. Dabei werden (jeweils wenn vorhanden) neben der Wikipedia, der offiziellen Website eines Künstlers und dem www.myspace.com-Profil sowohl www.discogs.com (sehr genau gepflegte Datenbank über Diskographien von Musikern), als auch www.last.fm als weiterer Recommender verlinkt. Für Songtexte steht ein Link zu www.musixmatch.com bereit. Außerdem wird das www.twitter.com Profil verlinkt. Um einen Überblick über aktuelle Live-Events zu gewähren, werden außerdem noch anstehende Konzerte direkt in der UI angezeigt, die jeweils einen Link auf www.songkick.com enthalten.
Insgesamt stehen dem Nutzer so viele Möglichkeiten und weitere Schritte zur Verfügung, einen Künstler zu ergründen. Im Gegensatz zu dem eher statischen Ausliefern an Vorschlägen und der Möglichkeit einzelne Lieder anzuhören, bietet MusicMashup-Recommender eher den Reiz, weiter auf dieser Plattform Musik zu erkunden, anstatt nach einmaliger Info zu einem Künstler zufrieden zu sein. Durch die Entkopplung von Musik und Browser ist auch ein Weitererkunden möglich, während die Musik weiterläuft.

4.3 Voting

Einer der Aspekte, die sicherlich noch großes Verbesserungspotential bergen, ist der Voting-Algorithmus und die teilweise fehlende Aussagekraft einer vorgeschlagenen Relation. Auf Grund der großen Variationen an Vorschlägen (wenige bzw. gar keine bis hin zu über hundert (z.B. Bob Dylan)) und einer tewilweise zufälllig wirkenden Anordnung bedarf der Voting-Algorithmus einer Überarbeitung beziehungsweise Ergänzung. Die Pageranks aus der dbpedia und der familiarity-Wert der Echonest-API dienen zwar in der Theorie als guter Indikator für die Bekanntheit eines Interpreten, allerdings kann dies auch eine verwirrende Wirkung haben, wenn ein Interpret mit einer eigentlich unwichtigen Begründung[j][k] durch seine Bekanntheit nahezu immer oben gelistet wird. Eine er-

neute Evaluation der Werte, die für einen gefundenen Grund berechnet werden, wird hier von Nöten sein. Es ist schwierig, einen ausgewogenen, generischen Voting-Algorithmus von Beginn an vorweisen zu können, da dieser viel getestet und angepasst werden muss und einige edge-cases erst nach einiger Zeit auffallen.
Neben der teilweise ungenauen Bewertung der Gründe gibt es auch noch semantisches Ergänzungspotential, welches in Zukunft noch implementiert werden wird. So sollten zum Beispiel Artisten, die zwar mehrere Begründungen enthalten, die aber alle von der gleichen Person stammen, weniger zählen, als Bands, bei denen diese alle von unterschiedlichen Personen stammen. Dies erfordert allerdings eine Neuimplementierung der Datenstruktur der Begründungen (diese werden derzeit nur als Strings in Textform abgespeichert), um dies generisch zu ermöglichen und war bis zum jetzigen Zeitpunkt noch nicht möglich.

4.4 Fallback für unbekannte Interpreten

Ein Nachteil gegenüber anderen Music-Recommender-Diensten ist außerdem, dass bei Interpreten, deren Mitglieder oder welche selbst nicht noch anderweitig tätig waren, gar keine Vorschläge gefunden werden können. Hier wäre es denkbar, stattdessen Related-Artists anzuzeigen, die zum Beispiel Spotify oder Echonest über ihre API liefern, oder die dbpedia selbst als associatedMusicalArtist listet. Allerdings würde dies dem eigentlichen Ansatz widersprechen, nur aus klar erkennbaren Gründen Bands vorzuschlagen. Sollte diese Lösung trotzdem Anwendung finden, um zu verhindern, dass man an einem toten Ende des Recommenders anlangen kann, so müsste dies allerdings erkenntlich gemacht werden.
Ein weiterer Punkt, welcher den MusicMashup-Recommender zusätzlich mit am meisten einschränkt, ist die Unvollständigkeit gerade bei unbekannteren oder neueren Musikern. Da als Musikdatenquelle ein nur eingeschränkt vollständiges Mapping der an sich umfangreichen Musikatenbank musicbrainz dient, welches durch ein Fallback mit einem weiteren Dump ergänzt wird, welches Wiederum eine Verlinkung zur dbpedia vermissen lässt, kommt es oft zu leeren Seiten. Für einige Fälle konnte hier eine Lösung gefunden werden, trotzdem Ergebnisse zu liefern, allerdings gab es leider keine generische.
Das ist eine sehr große Einschränkung der Nutzbarkeit. Die Abhängigkeit von der Vollständigkeit und Genauigkeit der benutzten Datensätze wird hier sehr gut deutlich. Neben vielen Vorteilen, die das Benutzen von Linked Open Data mit sich bringt, zeigt sich hier, dass es sehr wünschenswert wäre, wenn weitere Datenbanken zu kompletten Themengebieten - wie zum Beispiel Musik - in Linked Data vorliegen würden.

4.5 Linked Data

Generell allerdings hat das Benutzen von Linked-Data-Quellen und dem Folgen des Linked Data Ansatzes den großen Vorteil, dass die Daten bereits von Anfang an standardisiert, maschinenlesbar kodiert und verlinkt sind.
So ist das Zusammenführen der Arten simpel und die Programmierung nicht davon abhängig, für jede verwendete API die Daten selbst in das gewünschte Format zu überführen. Auch der Kontext muss nicht manuell und fehleranfällig selbst hergestellt werden,

etwa durch Suchen, sondern war von vornherein bekannt. Hierdurch konnte der Fokus bei der Programmierung von Anfang an auf die Arbeit mit den Daten gelegt werden. Bei der Verwendung anderer Quellen hätte man diese zunächst noch verarbeiten und selbst standardisieren müssen, um überhaupt mit ihnen umgehen zu können.
Auch der Fakt, dass es sich bei Linked-Data-Datenquellen um offene Datenquellen handelt, hat die Arbeit wesentlich erleichtert und beschleunigt, da keine Schlüssel oder ähnliches für die Verwendung proprietärer APIs beschafft und implementiert werden mussten.
Als hilfreich stellte sich auch heraus, dass für RDF und SPARQL bereits Bibliotheken existierten, auf die bei der Programmierung zurückgegriffen werden konnte.
Besonders hilfreich war, dass die Verlinkung zwischen einzelnen Ressourcen jeweils in einem Kontext standen. Dies machte es einfach, für die ausgegebenen Empfehlungen einen Kontext zu liefern, was ein Kernkonzept von MusicMashup ist.[l][m][n]

5 Zusammenfassung

Diese Arbeit hat einen Einblick in die Funktionsweise von MusicMashup-Recommender gegeben und ist auf Vor- und Nachteile von Linked Data im Rahmen des Projekts eingegangen. Das Ergebnis des Projekts ist ein vollwertiges, wenn auch erweiterbarer MusicRecommendationSystem, mittels welchem Benutzer sich Empfehlungen für neue Artists geben lassen können. Zusätzlich dazu werden ein Musikplayer, eine Liste anstehender Konzerte und weitere Informationen angeboten. Die so gesammelten Informationen werden mittels eines Parsers als Linked Data abgespeichert und können veröffentlich werden.

6 Ausblick

Aufgrund der begrenzten Entwicklungszeit kann das System an verschiedenen Stellen noch deutlich weiter entwickelt werden. Im Folgenden wird beschrieben, welche Entwicklungen das System in großem Maße verbessern würden.

6.1 Asynchrone Website

Momentan werden alle Queries zu Drittservern zunächst vollständig serverseitig abgewickelt, danach wird der gesamte Inhalt inklusive Empfehlungen an den Browser gesandt. Für den Nutzer entsteht so eine lange Wartezeit. Eine Alternative dazu wäre die Verwendung von AJAX, also Asynchronous Javascript and XML. Mit dieser Technologie lassen sich nach dem Laden einer Website noch weitere Inhalte nachladen. So könnte man die Grundwebsite direkt übertragen, dann die Informationen über den Künstler nachladen und schließlich die Empfehlungen einholen. Natürlich würde die Ladezeit dadurch insgesamt nicht sinken, sondern durch den Zusatz an asynchronen Abrufen eher noch leicht steigen, die Benutzererfahrung wäre jedoch wesentlich verbessert, da die Zeit, in der nichts auf der Seite passiert, stark sinkt. Der Benutzer könnte so zum Beispiel schon den Abstract der Band lesen, während die Recommendations noch geladen werden. Auch könnte man nicht alle Empfehlungen direkt laden, sondern mit einer Auswahl der relevantesten beginnen und dann die weiteren auf verschiedene Seiten verteilen („Pagination“).

6.2 Abstraktion schaffen

Website-Logik, das Abrufen der Daten von verschiedenen Endpoints, das Cachen von Daten und das Berechnen der Empfehlungen geschehen derzeit in einer einzigen Klasse. Um die Qualität und die Wartbarkeit des Codes zu erhöhen ist es sinnvoll, das System zu modularsieren und entsprechende Teile des Systems in einzelne Klassen auszulagern.[o]

6.3 Aufsetzen eines eigenen SPARQL-Endpoints

Das Aufsetzen eines eigenen SPARQL-Endpoints würde die Handhabung des Cachings vereinfachen und zusätzlich die Möglichkeit bieten, anderen Anwendungen einfachen Zugang zu den von uns gesammelten Daten zu schaffen. In der aktuellen Implementierung schreibt der Parser alle Daten in Turtle-Dateien. Das heißt das erstens bei Abruf vieler Artists auch viele Dateien enstehen und zweitens ist Turtle ein serialisierte Form von Linked Data, die im Vergleich zu einem SPARQL-Endpoint schlechter zum Abruf von Daten geeignet ist.

6.4 Personalisierung

Bisher wird als Eingabe vom Nutzer nur die eingegebene Ausgangs-Band ausgewertet. Dies könnte man erweitern, zum einen durch Auswerten von statistischen Daten, etwa welche Empfehlungen für welche Band besonders oft geklickt werden und daher möglicherweise besonders relevant sind, zum anderen durch Berücksichtigung weiterer direkter Eingaben. So könnte man beispielsweise den Standort des Nutzers bei der Anzeige von

Konzerten berücksichtigen. Eine weitere Möglichkeit wäre die Einbindung der Facebook-API. So könnte zum Beispiel für einen Nutzer von Facebook die Menge der von ihm mit einem "Like" versehen Artists geladen werden und als Grundlage für Empfehlungen herangezogen werden.
Auch die Anzeige der Ergebnisse ließe sich erweitern, indem die Liste der Empfehlungen nachträglich durch weitere Angaben gefiltert werden könnte, zum Beispiel nach einem bestimmten Genre. Auf Dauer könnte dies sogar pro Nutzer gespeichert und ausgewertet werden, sodass die Sortierung durch den Voting-Alghoritmus auch den persönlichen Geschmack des Nutzers berücksichtigt.
Ein weiterer großer Schritt wäre auch die Einbindung von Benutzeraccounts. Dies würde Benutzern zum Beispiel die Möglichkeit geben neu entdeckte Bands zu markieren und zu speichern, um einen erneuten Zugriff zu erleichtern. Ausgehend davon wäre sogar eine Anbindung an soziale Netzwerke denkbar, um neu entdeckte Artists mit sozialen Kontakten zu teilen.

Literatur