Bachelorarbeit

Serendipity-Metriken für Musik Recommender

Sven Gauß

Matrikelnummer: 2921015

Albert-Ludwigs-Universität Freiburg im Breisgau
Technische Fakultät
Institut für Informatik

Bearbeitungszeitraum

20. 09. 2012 – 20. 12. 2012

Gutachter

Prof. Dr. Georg Lausen

Betreuer

Dr.-Ing. Thomas Hornung

Kurzfassung

Ziel dieser Arbeit ist es Methoden zur Generierung von Empfehlungen zu entwickeln, die den Serendipity Aspekt fokussieren. Dieser Aspekt beschreibt den Überraschungseffekt einer Empfehlung. Es werden fünf Metriken vorgestellt, die diesen Effekt hervorrufen sollen. Zusätzlich wird das Recommender System *Song Stumbler* vorgestellt, welches die Metriken zur Empfehlungsgenerierung nutzt und als Prototyp, in Form einer Webanwendung, implementiert wurde.

Ein Teil der Metriken nutzt Charts verschiedener Länder, Städte und Jahre um Empfehlungen zu berechnen. Dadurch wird die allgemeine Beliebtheit und die Diversität der unterschiedlichen Charts für Empfehlungen ausgenutzt. Die übrigen Metriken setzen einen Last.fm Account voraus, um von diesem Informationen zu Präferenzen des Nutzers abrufen zu können. Die Methoden sollen überraschende Empfehlungen erzeugen und klassische Probleme wie Überspezialisierung umgehen.

Als Datenquellen diente die API des Musik-Streamingdienstes Last.fm und verschiedene Webseiten, die Informationen zu den gesuchten Charts anbieten. Die Daten des Recommenders werden automatisch im Wochenrhythmus neu abgerufen, um deren größtmögliche Aktualität zu gewährleisten. Insgesamt wurden 13.000 Musiktitel abgerufen, die für Empfehlungen genutzt werden können.

Darüber hinaus wird die dynamische Anpassung des Recommenders vorgestellt. Diese verteilt die Plätze einer Empfehlungsliste, anhand der Bewertungen des Nutzers, an die unterschiedlichen Metriken. Beliebte Metriken erhalten dadurch mehr Plätze, unbeliebte weniger Plätze einer Empfehlungsliste.

Die unterschiedlichen Metriken wurden im Rahmen einer Evaluation bewertet. Diese zeigte, dass überraschende Empfehlungen von den Teilnehmern größtenteils positiv bewertet wurden und die vorgestellten Metriken zufriedenstellende Empfehlungen erzeugten. Die Methoden, die auf Informationen eines Last.fm Accounts zurückgreifen, stellten sich als die Beliebtesten heraus.

Schlagwörter: Serendipity, Recommender Systeme, Musik Recommender

Inhaltsverzeichnis

1 Einleitung

In Zeiten wachsender digitaler Musikanbieter, wie zum Beispiel iTunes, die es möglich machen fast jedes beliebige Album oder Lied mit nur wenigen Mausklicks zu kaufen, nimmt die Zahl der Radiohörer immer weiter ab. Das Radio im traditionellen Sinne und seine Funktion als "Trendsetter" verliert, speziell bei jungen Menschen, an Bedeutung. Dies wird durch Musik-Streamingdienste wie Spotify oder Last.fm noch weiter verstärkt, denn diese bieten die Möglichkeit Musiktitel sofort und kostenlos wiederzugeben. Eine Studie der Universität Leipzig [1] zeigt, dass 92% der 12 bis 19 Jährigen den Computer zum Musik hören nutzen und nur 57% ein traditionelles Radiogerät. Unter den Computernutzern wächst dabei der Trend zum Online Musik hören, waren es 2007 noch 77% der Befragten, die angaben auch online Musik zu hören, sind es 2010 schon 92%. Ein Grund dafür ist die differenziertere Auswahl an Musiktiteln, die im Internet möglich ist, das traditionelle Radio aber nicht bietet. Da die Beratungsfunktion des Radios im Internet fehlt, fällt das Auswählen der Musiktitel auf die Konsumenten zurück. Von dieser Aufgabe sind jedoch, aufgrund der Masse an Angeboten im Internet, viele überfordert.

Um die Beratungsfunktion zu ersetzen, wurden so genannte Recommender Systeme [2] eingeführt. Diese Systeme sollen Benutzern helfen in einer Masse an Angeboten genau die zu finden, die dem jeweiligen Benutzer möglichst gut gefallen. Dazu nutzen Recommender zum Beispiel allgemeine Beliebtheit, demografische Informationen oder Verhaltensmuster, um so die Präferenzen des Benutzers vorherzusehen und dadurch zufriedenstellende Empfehlungen anzubieten. Im Kontext der Musikempfehlung sucht und empfiehlt ein Recommender System, in den zur Verfügung stehenden Musiktiteln, jene, die dem jeweiligen Benutzer möglicherweise gefallen könnten.

Recommender Systeme verzeichnen enorme Erfolge und wurden vor allem im E-commerce Bereich weitreichend eingesetzt [3]. *Amazon*[1] betreibt zum Beispiel meh-

[1] http://www.amazon.de - Stand: 4.11.2012

rere Recommender innerhalb der Webseite. Darunter “Ihnen könnten diese Artikel gefallen...” oder “Nutzer die diesen Artikel gekauft haben, kauften auch...”. Diese Empfehlungen werden auf Basis von gesammelten Daten, wie gekauften oder angesehenen Artikeln, erstellt. Ein bekanntes Recommender System im Bereich Musikempfehlung stellt der Musik-Streamingdienst Last.fm dar. Dieser bietet seinen Nutzern die Möglichkeit eine Historie über gehörte Lieder anzulegen und auf Basis dieser Historie ein personalisiertes Radio zu erstellen, das Lieder nach deren Präferenz spielt. Dieses Radio ist im Vergleich zum traditionellen Radio sehr stark personalisiert und lässt sich durch das Bannen, Lieben oder Abbrechen von Liedern weiter beeinflussen. Darüber hinaus bietet Last.fm eine API für Entwickler an, um den Zugriff auf die gesammelten Daten zu ermöglichen.

Viele Recommender Systeme, die bis in kürzeste Vergangenheit entwickelt wurden, versuchten den so genannten “Accuracy” Aspekt zu maximieren. Dieser Aspekt fokussiert die Modellierung der Benutzerpräferenzen, um Empfehlungen auszusprechen, die dem Benutzer mit immer höherer Wahrscheinlichkeit gefallen. Dabei besteht jedoch die Gefahr, dass Empfehlungen zu stark auf den Benutzer abgestimmt werden, dadurch für den Benutzer eintönig erscheinen und die Entdeckung einer neuen Präferenz verhindern. Dieses Problem wird auch “filter bubble” [4] genannt. Nach Ziegler [5] sind Benutzer gewillt auf einen Anteil dieser Genauigkeit zu verzichten, wenn dafür Aspekte wie Serendipity und “Novelty” gesteigert werden.

In dieser Ausarbeitung werden deshalb fünf Metriken vorgestellt, die den so genannten Serendipity Effekt bei der Empfehlung von Musiktiteln hervorrufen sollen. Serendipity (Deutsch: Serendipität) beschreibt die Ungewöhnlichkeit und den positiven Überraschungseffekt einer Empfehlung [6]. Dadurch sollen Probleme wie die “filter bubble” umgangen und stattdessen der Musikgeschmack des Benutzers erweitert werden. Der wachsende Trend in diese Richtung [7] bestätigt die Annahme, dass alternative Aspekte wie Serendipität großen Einfluss auf das Maß an Zufriedenstellung einer Empfehlung haben. Im Rahmen dieser Arbeit wurde ein Prototyp implementiert, das Recommender System *Song Stumbler*. Um das Recommender System mit Daten zu versorgen, wird zum einen die oben erwähnte Last.fm API genutzt und zum anderen werden verschiedene Charts aus unterschiedlichen Ländern, Jahren und auch Städten verwendet.

1.1 Aufbau

Im folgenden Kapitel 2 wird die Datenerhebung erläutert. Dabei werden der Musik-Streamingdienst Last.fm und dessen API Methoden genauer beschrieben. Zusätzlich werden die Quellen der unterschiedlichen Charts vorgestellt und die wesentlichen Implementierungsdetails erklärt.

Nachdem mit den erhobenen Daten die Grundlage des Recommender Systems geschaffen ist, werden in Kapitel 3 die Metriken zur Empfehlungsgenerierung erläutert. Diese lassen sich in Methoden ohne Last.fm Account und mit Last.fm Account einteilen. Im darauf folgenden Kapitel 4 wird der Prototyp des Recommender Systems vorgestellt. Zuerst wird der Aufbau des Prototyps beschrieben, wobei auf die einzelnen Unterseiten von Authentifizierung bis Admin-Tools eingegangen wird. Danach werden Implementierungsdetails zu den wesentlichen Mechanismen des Prototyps erläutert.

In Kapitel 5 werden verwandte Arbeiten vorgestellt, um diese Ausarbeitung in den Kontext anderer Arbeiten und Forschungen im Bereich Recommender Systeme, speziell Musik Recommender, einzugliedern. Anschließend wird in Kapitel 6 auf die Evaluation des Prototyps eingegangen. Zuerst werden die Teilnehmer der Evaluation genauer analysiert und danach die Ergebnisse und mögliche Schlussfolgerungen präsentiert. Schließlich folgt im letzten Kapitel 7 eine Zusammenfassung und ein Ausblick für zukünftige Arbeiten.

2 Datenerhebung

Die Datenerhebung lässt sich in zwei verschiedene Gruppen einteilen. Die erste Gruppe beinhaltet nutzerunabhängige Daten, bestehend aus Charts verschiedener Länder, Jahre und Städte. Um das Recommender System mit einer möglichst großen Datenmenge zu versorgen, wurden mehrere Quellen genutzt. Als Quelle der Hörercharts unterschiedlicher Städte diente die Last.fm API. Dabei wurden Hörercharts aus über 200 verschiedenen Städten, verteilt auf 31 Länder, gesammelt. Das führte zu einer Menge von über 6.700 gesammelten Musiktiteln. Die Charts der verschiedenen Länder und Jahre wurden von zwei unterschiedlichen Musik Webseiten abgerufen, die im Laufe des Kapitels kurz beschrieben werden. Dabei wurden jeweils die Top 20 Charts der Jahre 1962 bis 2012, zusammengestellt aus US und UK Charts, abgerufen und aktuelle Charts aus 27 verschiedenen Ländern gesammelt. Das führte zu einer Menge von 1.500 abgerufenen Titeln.

Die zweite Gruppe der Datenerhebung beinhaltet nutzerspezifische Daten. Diese beziehen sich direkt auf die Hörereignisse eines Last.fm Accounts. Da der Last.fm Account nicht im Voraus bekannt ist, müssen die nutzerpspezifischen Daten "on the go" gesammelt werden, nachdem der Account zur Verfügung gestellt wurde. Die zugehörigen Implementierungsdetails werden in Kap. 4.2 beschrieben. Bei einem aktuellen Stand[1] von knapp 50 angemeldeten Nutzern, davon 21 mit Last.fm Account, belaufen sich die spezifischen Daten auf ungefähr 4.000 Titel.

Die gesammelten Daten belaufen sich insgesamt auf ca. 13.000 Musiktitel, aus denen Empfehlungen ausgewählt werden können. Die komplette Datenmenge wird periodisch, im Wochenrhythmus, aktualisiert. Ein Crawlen über mehrere Wochen wäre daher im laufenden Betrieb nicht sinnvoll und es wird stattdessen gewährleistet, dass die Daten immer die größtmögliche Aktualität besitzen.

Im Folgenden werden die unterschiedlichen Quellen und Methoden, die zur Sammlung der Daten genutzt wurden, vorgestellt und erläutert.

[1] Abgerufen am 11.12.2012

2.1 Last.fm

Der Musik-Streamingdienst Last.fm[1] bietet Nutzern die Möglichkeit ihre Hörereignisse aufzuzeichnen, auch *Scrobbeln* genannt und daraus eine persönliche Historie erstellen zu lassen. Aus dieser Historie generiert Last.fm unterschiedliche Toplisten an gehörten Liedern, Künstlern oder Genres.

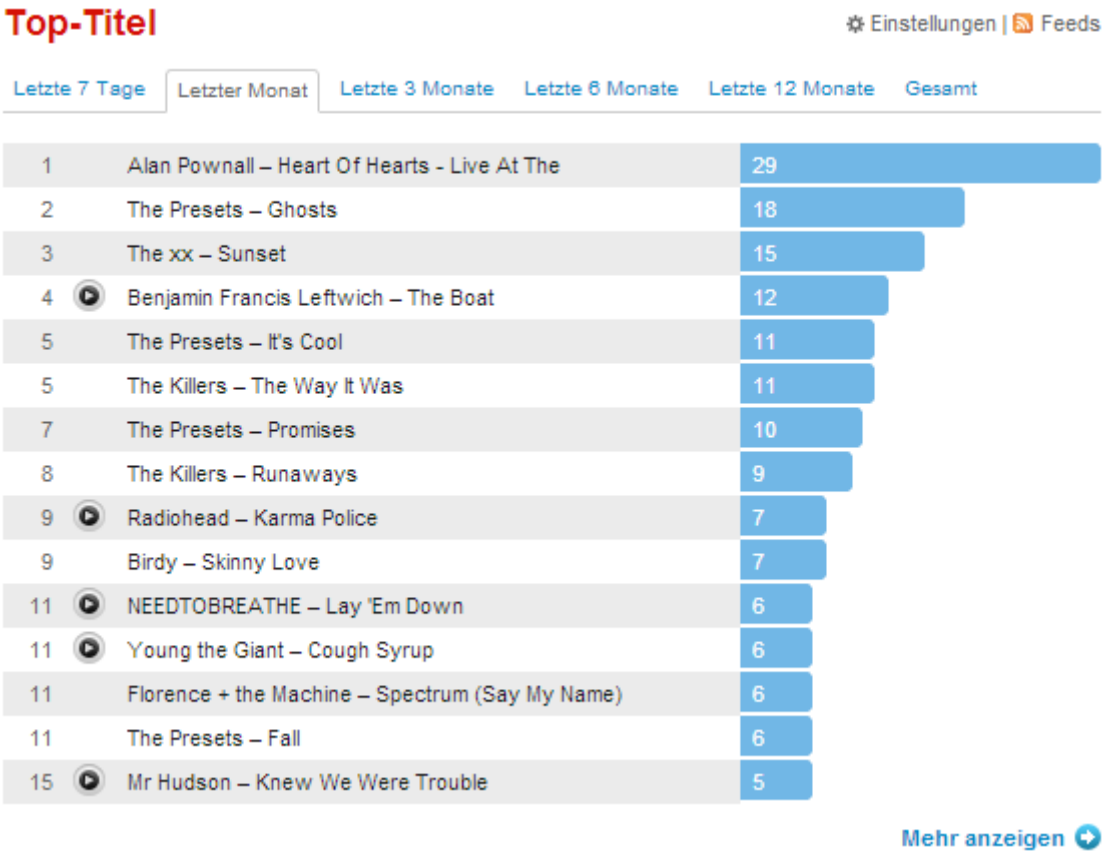


Abbildung 2.1: Last.fm Top-Titel eines Nutzers

Zu jedem Lied oder Künstler lassen sich ähnliche Lieder oder Künstler anzeigen, um auf diese Weise neue Musik zu entdecken. Des Weiteren gibt Last.fm so genannte Nachbarn an. Diese Nutzer sollen laut Last.fm einen ähnlichen Musik Geschmack haben und eignen sich daher ebenfalls zum Entdecken neuer Musik. Nutzer die nicht selbst nach Musiktiteln suchen wollen, können das so genannte “Last.fm Radio” benutzen. Dabei gibt der Nutzer zuerst ein Lied oder einen Künstler an. Anschließend werden kontinuierlich Lieder abgespielt, die diesem ähneln.

[1]http://www.lastfm.de - Stand: 11.12.2012

Darüber hinaus erstellt Last.fm auf Grundlage der Hörereignisse aller Nutzer eine Vielzahl an Hörercharts. Zum Beispiel werden, anhand geographischer Informationen, Hörercharts unterschiedlicher Regionen und Städte erstellt.

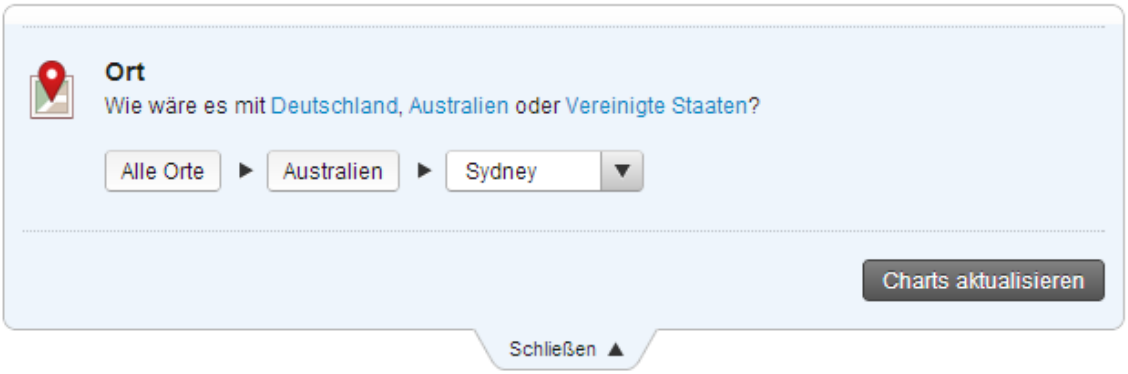


Abbildung 2.2: Auswahl der Last.fm Hörercharts

Durch die Menge an Angeboten hat Last.fm eine sehr große Nutzerbasis aufgebaut und daher große Mengen an Daten gesammelt. Aktuelle Angaben[1] geben ungefähr 178.000.000 gehörte Titel pro Woche an und seit der Gründung im Jahr 2003 eine Menge von ca. 70.000.000.000 Titeln. Aufgrund dieser Menge ist Last.fm eine beliebte Datenquelle für Forschungen im Bereich der Musik Recommender. Last.fm stellt eine API für Entwickler zur Verfügung, die eine Fülle an Möglichkeiten bietet auf die gesammelten Daten zuzugreifen. Die Methoden sind gruppiert in User, Track, Artist, Album, Geo und einige mehr. Methoden sind beispielsweise, *User.getTopTracks*, *Track.getInfo* oder *Artist.getTopAlbums*. Die API und deren Methoden werden im Folgenden erläutert.

Die Methoden der Last.fm API werden durch eine URL-Anfrage angesprochen. Das heißt es wird eine URL angefordert, die Methoden und Parameter enthält. Zum Beispiel ruft *http://ws.audioscrobbler.com/2.0/?method=geo.getmetros&country=spain* die Methode *Geo.getMetros* für das Land Spanien auf, um alle Städte in Spanien zu erhalten, von denen Hörercharts zur Verfügung stehen. Als Antwort erhält man eine XML Datei, welche die gewünschten Informationen enthält. Für den Prototyp wurde ein Crawler in Java implementiert, der die XML Antworten der Last.fm API, mithilfe von XPath[2], verarbeitet. Die folgenden Kapitel beschränken sich auf die im Prototyp genutzten Methoden.

[1] http://www.lastfm.de/community/ - Stand: 11.12.2012
[2] http://www.w3.org/TR/xpath/ - Stand: 11.12.2012

2.1.1 User.getTopTracks

Die Methode *User.getTopTracks* liefert die Top-Lieder eines bestimmten Last.fm Accounts. Dabei lässt sich die Methode mit folgenden Parametern aufrufen:

1. **User**: Last.fm Account dessen Top-Lieder abgerufen werden sollen.
2. **Period** (Optional): Zeitspanne deren Top-Lieder abgerufen werden sollen.

Als Antwort erhält man eine XML Datei, wie beispielweise für einen User namens *RJ* in Listing 2.1 dargestellt.

```
<toptracks user="RJ" type="overall">
  <track rank="1">
    <name>Learning to Live</name>
    <playcount>42</playcount>
    <url>http://www.last.fm/music/Dream+Theater/Learning+to+Live
    </url>
    <artist>
      <name>Dream Theater</name>
      <mbid>28503ab7-8bf2-4666-a7bd-2644bfc7cb1d</mbid>
      <url>http://www.last.fm/music/Dream+Theater</url>
    </artist>
  </track>
  ...
</toptracks>
```

Listing 2.1: Beispielantwort der Methode User.getTopTracks

Das Attribut *type* des Elements *toptracks* stellt die, als Parameter angegebene, Zeitspanne dar. Jedes Element *track* besitzt ein Attribut *rank*, das seinen Platz in der Rangliste wiederspiegelt. Zusätzlich besitzt ein *track* die Kinderelemente *name*, *playcount*, *url* und *artist*. Das Element *playcount* stellt dabei die Anzahl der Hörereignisse des Last.fm Nutzers dar. Das Kinderelement *artist* besitzt wiederum selbst die Element *name*, *url* und eine *MusicBrainz*[1] ID. *MusicBrainz* sammelt Metadaten über Künstler und ihre Veröffentlichungen. Die ID dient zur eindeutigen Identifizierung von Objekten in der *MusicBrainz* Datenbank.

[1] http://www.musicbrainz.org/ - Eine offene Datenbank an Informationen über Musik.

2.1.2 User.getNeighbours

Die Methode *User.getNeighbours* liefert die Nachbarn eines bestimmten Last.fm Accounts. Ein Nachbar ist laut Last.fm ein Nutzer mit ähnlichem Musikgeschmack und wird durch einen unbekannten Algorithmus berechnet. Je höher der berechnete Nachbarschaftsgrad, desto ähnlicher der Musikgeschmack.

Die Methode lässt sich mit einem Last.fm Account als Parameter aufrufen:

1. **User**: Last.fm Account dessen Nachbarn abgerufen werden sollen.

Als Antwort erhält man eine XML Datei, wie als Beispiel für den User *RJ* in Listing 2.2 dargestellt.

```
<neighbours for="RJ">
  <user>
    <name>Lauramynch</name>
    <url>http://www.last.fm/user/Lauramynch</url>
    <image>http://userserve-ak.last.fm/serve//1451306.gif</image>
    <match>0.9</match>
  </user>
  ...
</neighbours>
```

Listing 2.2: Beispielantwort der Methode User.getNeighbours

Die Antwort enthält eine Menge an *user* Elementen, welche die Nachbarn des Last.fm Accounts repräsentieren. Jeder *user* hat die Kinderlemente *name*, *url*, *image* und *match*. Das Element *url* verweist auf die Profilseite des jeweiligen Last.fm Accounts und *image* auf ein optionales Profilbild. Den Nachbarschaftsgrad der Nutzer gibt das *match* Element an. Die Menge der Nachbarn ist nach diesem Element absteigend sortiert. Der Nachbar mit dem ähnlichsten Musikgeschmack steht daher an erster Stelle und der Nachbar mit dem unterschiedlichsten Musikgeschmack an letzter Stelle.

2.1.3 Geo.getMetroTrackChart

Die Methode *Geo.getMetroTrackChart* liefert die Hörercharts einer Stadt, welche Nutzer als Wohnort eingetragen haben. Es lassen sich folgende Parameter angeben:

1. **Metro**: Name der gesuchten Stadt.

2. **Country**: Name des Landes in dem sich die Stadt befindet.
3. **Period** (Optional): Mit den Parametern **start** und **end** lässt sich eine bestimmte Zeitspanne angeben, aus denen die Hörercharts zusammengestellt sein sollen. Die Werte der Parameter müssen als Unix-Timestamp angegeben werden.

Als Antwort erhält man wiederum eine XML Datei, diese ist analog zur Antwort der Methode *User.getTopTracks* aufgebaut, die in Listing 2.1 dargestellt ist. Der einzige Unterschied besteht darin, dass das Element *playcount* eines Liedes durch das Element *listeners* ersetzt wird. Dieses gibt die Anzahl der unterschiedlichen Nutzer an, die in der angegebenen Stadt wohnen und das jeweilige Lied in der spezifizierten Zeitspanne gehört haben.

Die Hörercharts sind dabei nach Anzahl der *listeners* erstellt. Das Lied mit den meisten Hörern belegt daher den ersten Platz der Hörercharts und das Lied mit den wenigsten Hörern belegt den letzten Platz. Auf diese Weise werden Hörercharts für über 220 verschiedene Städte aus mehr als 30 Ländern erstellt.

2.1.4 Track.getTopTags

Die Methode *Track.getTopTags* liefert eine Menge an Tags, die einem Lied zugewiesen sind. Ein Tag stellt eine kurze Beschreibung des Genres oder des Inhaltes dar und wird einem Lied von unterschiedlichen Nutzern zugewiesen, wobei ein Lied eine beliebige Anzahl Tags besitzen kann. Die Methode wird mit einem Musiktitel als Kombination aus folgenden Parametern aufgerufen:

1. **Artist**: Name des Künstlers.
2. **Track**: Titel des Liedes.

Als Beispiel wurde die Methode mit dem Lied “Jailhouse Rock” von “Elvis Presley” aufgerufen. Die Antwort dieses Methodenaufrufs ist in Listing 2.3 dargestellt. Sie enthält eine Menge an Tags, die mit dem angegebenen Lied assoziiert werden. Jedes *tag* Element hat die Kinderelemente *name*, *count*, und *url*. Das Element *name* gibt den Namen des Tags an und *url* die entsprechende Last.fm Webseite. Des Element *count* beschreibt dabei die Anzahl der User, die dem angegebenen Lied den jeweiligen Tag zugeordnet haben. Die Tags sind absteigend nach dem Wert des *count* Elements geordnet. Der Tag, der dem betreffenden Lied am öftesten zugewiesen wurde, steht somit an erster Stelle.

```
<toptags artist="Elvis Presley" track="Jailhouse Rock">
  <tag>
    <name>rock n roll</name>
    <count>100</count>
    <url>http://www.last.fm/tag/rock%20n%20roll</url>
  </tag>
  <tag>
    <name>50s</name>
    <count>30</count>
    <url>http://www.last.fm/tag/50s</url>
  </tag>
  <tag>
    <name>rockabilly</name>
    <count>19</count>
    <url>http://www.last.fm/tag/rockabilly</url>
  </tag>
  ...
</toptags>
```

Listing 2.3: Beispielantwort der Methode Track.getTopTags

2.1.5 Tag.getTopTracks

Die Methode *Tag.getTopTracks* liefert die Top-Lieder eines angegebenen Tags. Es lässt sich nur der Tag selber als Parameter angeben.

Die Antwort fällt wieder analog zu Listing 2.1 aus und besteht aus einer Menge von Liedern, die den angegeben Tag besitzen. In der Antwort dieser Methode besitzt ein *track* kein *playcount* oder *listener* Element, sondern ein *rank* Attribut. Dieses repräsentiert die Platzierung des Titels in den Hörercharts.

Die meisten der, im Laufe des Kapitels genannten, API Methoden bieten die Möglichkeit, die Anzahl der Ergebnisse und Seiten, durch Angabe der Parameter **limit** und **page** festzulegen. Zusätzlich benötigen alle Methoden einen “API key”, welcher die Anfrage mit einem zugehörigen API Account des Entwicklers identifiziert. Dieser wird durch den Parameter **key** übergeben.

2.2 Charts

Neben der Last.fm API als Datenquelle wurden offizielle Charts unterschiedlicher Jahre und Länder genutzt. Dafür wurden nach längerer Recherche zwei verschiedene Quellen gefunden.

Für das Abrufen der Charts von den Jahren 1960 bis zum aktuellen Jahr 2012 wurde die Webseite *MusicVF*[1] genutzt. *MusicVF* existiert seit 2009 und bietet Jahrescharts, Jahrzent-Charts und so genannte "All-Time" Toplisten an, in denen die erfolgreichsten Künstler aller Jahre verzeichnet sind. Dabei werden Charts der Jahre 1900 bis zum aktuellen Jahr angeboten. Als Quelle nutzt *MusicVF* selbst die bekannten *Billboard*[2] Daten. Für den Prototyp wurde *MusicVF* gewählt, da die Charts früherer Jahre in der aktuellen Billboard Version nicht mehr zu finden sind.

Um Informationen über Charts möglichst vieler unterschiedlicher Länder zu erhalten, wurde die Webseite *Top40-Charts*[3] genutzt. *Top40-Charts* bietet eine große Auswahl an Charts der ganzen Welt. Dabei sind über 25 verschiedene Länder, darunter Irland, Spanien oder Dänemark, mit Top 20 oder Top 40 Charts vertreten. Des Weiteren werden globale Genre-Toplisten angeboten und die Top 100 der digitalen Verkäufe, zusammengestellt von iTunes, Amazon und EBay, zur Verfügung gestellt.

Da die genannten Quellen keine offizielle API für ihre Daten zur Verfügung stellen, sondern nur das HTML Konstrukt einsehbar ist, musste ein Weg zur Filterung der gewünschten Informationen gefunden werden. Für den Prototyp wurden die Quellen mithilfe von regulären Ausdrücken gefiltert und anschließend die gewollten Informationen ausgelesen. Im Folgenden wird der Vorgang beispielhaft an den Charts des Jahres 1975 erläutert.

Im ersten Schritt wird das gesamte HTML Dokument eingelesen, gekürzt dargestellt in Listing 2.4. Der Auschnitt zeigt das HTML Konstrukt das benötigt wird, um den Titel "December, 1963 (Oh, What a Night)" von "The Four Seasons" auf der Webseite darzustellen.

[1] http://www.musicvf.com/ - Stand: 11.12.2012
[2] http://www.billboard.com/ - Stand: 11.12.2012
[3] htttp://www.top40-charts.com/ - Stand: 11.12.2012

```
<html>
  <head>
    <meta name="robots" content="All"/>
    ...
    <script type="text/javascript" src="./cookies.js"></script>
  </head>
  <body>
    <table cellspacing="1" cellpadding="3" ...>
      ...
      <td align="right">
        <small>1.</small>
      </td>
      <td align="right">
        12/<a href="./1975.year" title="1975">1975</a>
      </td>
      <td>
        <a href="./song.php?id=42964" title="December, 1963 (Oh,
            What a Night)">December, 1963 (Oh, What a Night)</a> –
        <a href="The+Four+Seasons.art">The Four Seasons</a>
         
        ...
      </td>
      ...
    </table>
  ...
  </body>
</html>
```

Listing 2.4: Auschnitt des HTML Dokumentes

Um Titel und Künstler herauszufiltern, muss irrelevante HTML Syntax entfernt werden. Dazu werden folgende reguläre Ausdrücke verwendet:

- **<style.*?>.*?</style>**: Filtert alle Style Tags und deren Inhalt.
- **<!–.*?–>**: Filtert alle Kommentare und deren Inhalt.
- **<.*?>**: Filtert alle übrigen HTML Tags, lässt jedoch den Inhalt bestehen.

Nachdem die HTML Syntax entfernt wurde, erhält man für jedes Lied, das in den Charts vorkommt, eine Ergebniszeile der Form:

*1.12/1975December, 1963 (Oh, What a Night) - The Four Seasons *

Jede dieser Zeilen beinhaltet die Informationen eines Titels aus den abgerufenen Charts. Zuerst steht die Platzierung dann folgt ein Datum in Form “Monat/Jahr” und anschließend der Titel in der Form “Lied - Künstler”. Jeder Abschnitt eines Titels endet mit einem HTML kodierten Leerzeichen “ ”.

Um Titel und Künstler zu extrahieren, wird zuerst die Platzierung und das Jahr des aktuell betrachteten Abschnittes entfernt. Es entsteht folgendes Resultat:

*December, 1963 (Oh, What a Night) - The Four Seasons *

Der Titel des Liedes erstreckt sich bis zum Zeichen “ - ” und kann extrahiert werden. Man erhält den Titel *“December, 1963 (Oh, What a Night)”*. Danach werden alle Zeichen vor dem Trennsymbol entfernt und es entsteht folgendes:

- The Four Seasons .

Es bleibt der Künstlername, der extrahiert werden muss. Um dies zu erreichen, wird das nach jedem Künstler stehende “ ”, sowie das Trennzeichen “ - ” entfernt. Man erhält den Künstlername *“The Four Seasons”*. Nun können Titel und Künstlername gespeichert werden und es wird mit den nachfolgenden Titeln bis Platz 20 analog verfahren.

Charts anderer Länder lassen sich auf ähnliche Art und Weise abrufen, da es jedoch unterschiedliche Quellen sind, unterscheidet sich das Vorgehen nachdem die HTML Syntax entfernt wurde etwas von dem gezeigten Beispiel.

Als Abschluss der Erläuterungen zur Datenerhebung, wird in Tab. 2.1 ein Überblick der unterschiedlichen Daten und ihrer Quellen dargestellt.

Daten	**Quelle**
Jahres Charts	MusicVF.com
Länder Charts	Top40-Charts.com
Städte Charts	Last.fm API: *Geo.getMetroTrackChart*
Hörhistorien	Last.fm API: *User.getTopTracks*

Tabelle 2.1: Quellenübersicht

3 Metriken

Die Aufgabe eines Recommender Systems ist es, einem Nutzer Empfehlungen auszusprechen. Es stellt sich jedoch die Frage, welche Ziele bei der Empfehlungsgenerierung angestrebt werden. Nach Zhang [6] und anderen Arbeiten im Bereich Recommender Systeme lassen sich vier unterschiedliche Zielsetzungen erkennen.

1. **Accuracy**: Der "Accuracy" Aspekt eines Recommender Systems stellt die Exaktheit der Empfehlungen in den Vordergrund. Das heißt zu welchem Maße spiegelt die ausgesprochene Empfehlung die Präferenzen des Nutzers wider.
2. **Novelty**: Der "Novelty" Aspekt eines Recommender Systems legt den Schwerpunkt auf die Neuheit einer Empfehlung. Dieser Aspekt ist erfüllt, wenn eine Empfehlung vorher noch nie gesehen, gehört oder auf andere Art und Weise wahrgenommen wurde. Es bezieht sich auf **alle** vorangegangenen Erfahrungen des Nutzers.
3. **Diversity**: Der "Diversity" Aspekt eines Recommender Systems ähnelt stark dem der Neuheit einer Empfehlung, lässt sich aber nach Vargas [7] dadurch differenzieren, dass er sich auf eine beschränkte Menge von Objekten bezieht. Das heißt, es wird beispielsweise die Vielfalt aller ausgesprochenen Empfehlungen betrachtet, jedoch nicht die der Erfahrungen darüber hinaus, wie es beim "Novelty" Aspekt der Fall ist.
4. **Serendipity**: Der bereits erwähnte Serendipity Aspekt beschreibt den Überraschungseffekt einer Empfehlung. Recommender Systeme die den Serendipity Aspekt verfolgen, sollen Empfehlungen erzeugen, nach denen der Nutzer selbst nicht gesucht hätte oder nicht erwartet hätte, welche aber trotzdem einen positiven Effekt mit sich bringen.

Die im Folgenden beschriebenen Metriken sollen den letzteren, nämlich den Serendipity Aspekt, fokussieren.

Die Methoden lassen sich, wie schon bei der Datenerhebung, in zwei Gruppen aufteilen. Die erste Gruppe benötigt keinen Last.fm Account. Diese Methoden basieren auf demografischen Hintergründen des jeweiligen Nutzers sowie Informationen zur allgemeinen Beliebtheit. Diese Gruppe wird in den Kapiteln 3.1 bis 3.3 betrachtet.

Die nachfolgenden Kapitel 3.4 und 3.5 beschreiben die Methoden der zweiten Gruppe, welche abhängig von einem angegebenen Last.fm Account sind. Diese Methoden greifen auf die Hörhistorien der Nutzer zu, um auf Basis dieser Informationen Empfehlungen zu erstellen.

Beide Gruppen arbeiten mit dem gleichen Modell eines Nutzers, welches in Abb. 3.1 schematisch dargestellt ist.

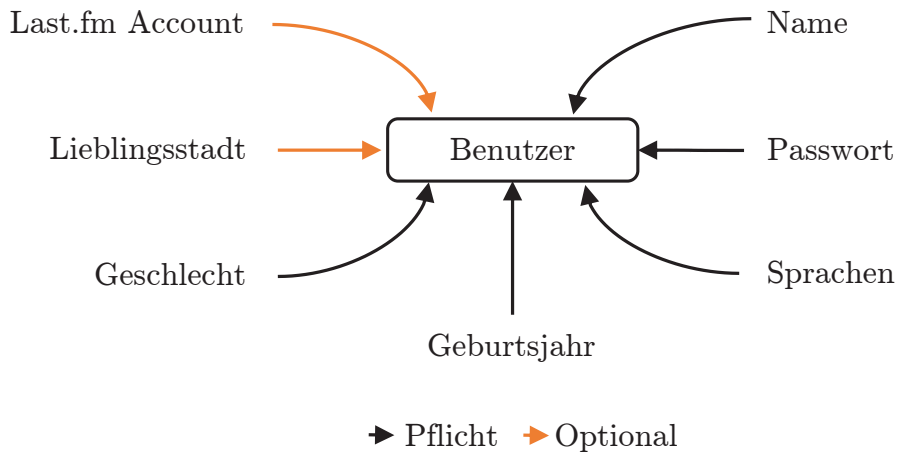


Abbildung 3.1: Benutzermodell

Von jedem Nutzer sind Name, Passwort, Geburtsjahr, Geschlecht und zusätzlich eine Menge an Sprachen bekannt. Diese besteht mindestens aus Englisch, kann aber mit anderen Sprachen erweitert werden, die der Nutzer spricht oder mag. Die Informationen werden während der Registrierung abgefragt und müssen, für eine erfolgreiche Registrierung, angegeben werden. Darüber hinaus kann ein Nutzer eine Lieblingsstadt und einen Last.fm Account angeben. Diese Angaben sind jedoch nicht verpflichtend und die Registrierung ist auch ohne sie möglich.

Die Informationen des Benutzermodells werden für die folgenden Methoden als gegeben vorausgesetzt und können für einen beliebigen Nutzer abgefragt werden.

3.1 Jahres Charts

Die Idee dieser Methoden basiert auf der Annahme, dass der Höhepunkt der Entwicklung musikalischer Präferenzen im zweiten Lebensjahrzehnt erreicht ist. Als Jugendlicher und Heranwachsender entwickelt man seinen persönlichen Musikgeschmack, wird dabei von Eltern, Freunden und immer stärker von den Medien beeinflusst. In der Regel haben Menschen daher zeit ihres Lebens eine emotionale Bindung zur Musik ihrer Jugend.

Um diese emotionale Bindung für die Empfehlungen des Recommender Systems zu nutzen, werden die in Kap. 2.2 erwähnten Charts der Jahre ab 1960 genutzt. Zusätzlich wird bei der Registrierung das Geburtsjahr erfasst, um den Nutzer dadurch in einen zeitlichen Kontext einordnen zu können. Dem Nutzer werden anschließend Lieder empfohlen, die während seines zweiten Lebensjahrzehntes aktuell und populär waren. Gibt der Nutzer beispielsweise 1975 als Geburtsjahr an, so wird er in den musikalischen Kontext von 1989 bis 1995 eingeordnet, was seinem 14. bis 20. Lebensjahr entspricht.

Um Empfehlungen auszuwählen, muss eine Art Ranking festgelegt werden, das bestimmt welche Lieder zuerst und welche erst später in eine Empfehlungsliste aufgenommen werden. Im diesem Fall wird das Ranking schon explizit mit den Daten geliefert, nämlich die Platzierung der Lieder in deren Charts. Diese werden absteigend nach ihrer Platzierung ausgewählt. Zuerst die Erstplatzierten, dann folgen die Zweitplatzierten bis hin zu den Letztplatzierten. Es werden daher nie alle Empfehlungen einer Liste von nur einem einzelnen Jahr stammen, außer die Methode bietet nur eine einzelne Empfehlung an.

Titel	**Künstler**	**Jahr**	**Platzierung**
I've Been Thinking About You	Londonbeat	1990	6
Baby Baby	Amy Grant	1991	6
Don't Walk Away	Jade	1992	6
I'm Every Woman	Whitney Houston	1993	7

Tabelle 3.1: Auszug aus Empfehlungen der Jahres Charts Methode

In Tab. 3.1 ist ein Auszug der Empfehlungen eines 1975 geborenen Nutzers zu sehen. Man erkennt bereits in diesem kleinen Ausschnitt, dass eine gewisse Vielfalt vorhanden ist, da die Lieder aus unterschiedlichen Jahren stammen. Zusammenfassend empfiehlt diese Methode dem Nutzer Lieder, die er vielleicht schon länger nicht

mehr gehört, oder sogar noch nie gehört hat und erzeugt damit möglicherweise den gewünschten positiven Überraschungseffekt.

3.2 Ausländische Charts

Diese Methode Empfehlungen zu erstellen, basiert auf der Erkenntnis, dass Lieder, die für einen Nutzer gerade aktuell sind, oft schon vor Monaten in anderen Ländern an der Spitze der Charts standen. Zusätzlich sind musikalische Geschmäcker unterschiedlicher Länder oft stark differenziert, was sich aus der Tatsache schließen lässt, dass die Charts unterschiedlicher Länder meist nicht übereinstimmen. Es wird somit die allgemeine Beliebtheit der Lieder in Charts ausgenutzt aber auch die Vielfältigkeit der unterschiedlichen Länder.

Diese Vielfältigkeit lässt sich analog zur "Listener Diversity" nach Zhang et al [6] beschreiben und für das Aussprechen von Empfehlungen verwenden. Man priorisiert dabei Künstler, die aus unterschiedlichen Hörergemeinschaften stammen, um so den Effekt einer überraschenden Entdeckung zu fördern. Die Menge an Ländern, aus deren Charts die Empfehlungen für bestimmte Nutzer geschöpft werden wird implizit über Sprachen erstellt. Für jeden Nutzer liegt, wie im Benutzermodell Abb. 3.1 festgelegt, eine Menge an Sprachen vor, die während der Registrierung angegeben wurden. Durch eine manuell erstellte Relation, siehe Tab. 3.2, kann von den angegebenen Sprachen auf die Länder geschlossen werden. Dabei werden auch multiple Amtssprachen berücksichtigt, wie beispielsweise in Kanada, das sowohl Englisch als auch Französisch als Amtssprache besitzt.

Sprache	**Land**
Portugiesisch	Brasilien
Englisch	Kanada
Französisch	Kanada
Spanisch	Chile

Tabelle 3.2: Auszug der Sprache-Land Relation in der Datenbank

Als Quelle der Charts werden die in Kap. 2.2 vorgestellten Ländercharts genutzt. Um für jeden Nutzer eine Menge an möglichen Empfehlungen zu erstellen, wird folgendermaßen vorgegangen:

Sei C_D die Menge der Lieder in den deutschen Charts und C_X die Menge der Lieder in den Charts eines anderen Landes X. Dabei sei L, die Menge der Länder, für die dem Nutzer Charts zur Verfügung stehen. Die Menge der möglichen Empfehlungen C_E berechnet sich wie folgt.

$$C_E = \bigcup_{X \in L} (C_X \setminus C_X \cap C_D)$$

Somit berechnet sich die Menge der möglichen Empfehlungen durch die Vereinigung der zur Verfügung stehenden Charts, von denen jeweils die Schnittmenge mit den Deutschen Charts subtrahiert wurde.

Das Ranking nach dem die Lieder ausgewählt werden ist, wie in der vorherigen Methode, explizit durch die Platzierung in den verschiedenen Charts gegeben. Tab. 3.3 zeigt eine Auswahl an Empfehlungen für einen Benutzer, der die Sprachen Englisch und Französisch angegeben hat.

Titel	**Künstler**	**Land**	**Platzierung**
C'est La Vie	DJ Khaled	Frankreich	6
All I Want	Kodaline	Ireland	6
Rock Star	Reece Mastin	Neuseeland	7
Hurt Me Tomorrow	K'Naan	Kanada	7

Tabelle 3.3: Auszug aus Empfehlungen der Länder Charts Methode

Auffällig ist die geographische Diversität, welche durch diese Methode zusammengestellt wird. Dadurch wird die bereits erwähnte "Listener Diversity" erzeugt, die sich nicht nur durch die Entdeckung neuer Musiktitel, sondern auch durch Entdeckung neuer Präferenzen, welche vorher nicht bewusst waren, auszeichnet. Serendipity Effekte sollten durch diese Methode vermehrt erzeugt werden und es dadurch dem Nutzer ermöglichen neue Musik zu entdecken.

3.3 Städte Charts

Diese Methode versucht auf eine Vielzahl unterschiedlicher Hörergemeinschaften zuzugreifen, um so Empfehlungen, die den Serendipity Effekt hervorrufen, zu fördern. Im Unterschied zur vorherigen Methode wird dabei die Granularität einer Hörergemeinschaft sehr viel kleiner definiert, nämlich auf unterschiedliche Städte bezogen.

Für diese Städte gelten einerseits in kleinerem Rahmen die Überlegungen der Diversität in unterschiedlichen Ländern, so können sich selbst von Stadt zu Stadt unterschiedliche Musikgeschmäcker ausprägen. Andererseits eröffnen sich durch die stark fokussierte Betrachtung neue Möglichkeiten. Beispielsweise können unbekanntere "Bar"-Bands in den Hörercharts einer Stadt durchaus eine starke Rolle spielen, was in Ländercharts hingegen fast unmöglich wäre. Dadurch entsteht für die Nutzer, die Empfehlungen durch diese Methode erhalten, die Möglichkeit diese kleineren Bands zu entdecken, nach denen man aus eigenem Antrieb heraus nicht gesucht hätte. Auf diese sehr regionalen Ergebnisse soll in dieser Methode verstärkt Wert gelegt werden.

Auch für diese Methode wird der Nutzer in einen geographischen Kontext gesetzt. Um dies zu erreichen, werden dem Nutzer zwei Möglichkeiten geboten, eine Implizite und eine Explizite. Die implizite Variante funktioniert analog zum Vorgehen in der vorherigen Methode, bei der die angegebenen Sprachen des Nutzers verwendet werden, um auf zugehörige Länder zu schließen. Sind die Länder eines Nutzers festgelegt, werden im nächsten Schritt für jedes Land die zugehörigen Städte ausgesucht. Dabei werden die Hörercharts unterschiedlicher Städte genutzt, welche schon in Kap. 2.1.3 vorgestellt wurden. Die Auswahl an Städten ist dabei auf das Angebot der Last.fm API beschränkt. Die explizite Variante gibt dem Nutzer die Möglichkeit eine Lieblingsstadt anzugeben. Diese Möglichkeit ist optional, wird sie jedoch wahrgenommen werden die Hörercharts dieser Stadt verstärkt in den Ergebnissen widergespiegelt.

Die Menge, der zur Verfügung stehenden Empfehlungen, bildet sich über die Vereinigung der Charts aller dem Nutzer zugewiesenen Städte. Dadurch entsteht eine große Auswahl an Empfehlungen für jeden Nutzer und es muss wieder ein Ranking eingeführt werden, nachdem die Empfehlungen aus dem "Pool" entnommen werden. In diesem Fall wird nicht ausschließlich die Platzierung in den Charts gewählt, weil es vermehrt vorkommen kann, dass Städte eines gleichen Landes ähnliche Lieder in ihren Hörercharts vertreten haben.

Deshalb wird eine Art Sortierung durchgeführt, um so die einzigartigen Interpreten oder Musiktitel einer Stadt zu finden. Zuerst wird für jedes Lied die Anzahl der Städte berechnet, in dessen Charts es vorkommt. Veranschaulicht für drei Städte A, B und C in Abb. 3.2, dessen Charts durch die Mengen S_A, S_B und S_C repräsentiert werden. Die Abbildung zeigt die Verteilung der verschiedenen Lieder auf

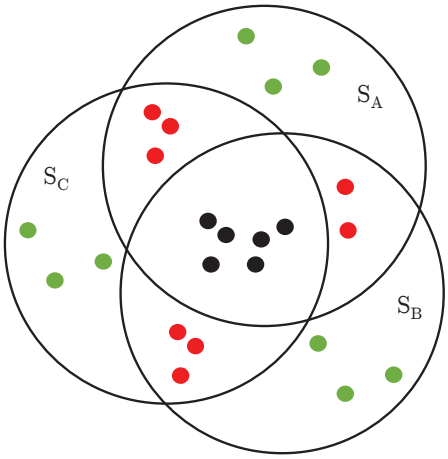


Abbildung 3.2: Schnittmengen am Beispiel von drei Städten

die unterschiedlichen Charts. Lieder können in jeder Stadt vorkommen, dargestellt durch die schwarzen Punkte. Sie können jedoch auch einzigartiger sein und wie die roten Punkte in nur zwei Städten vorkommen. Die außergewöhnlichsten Lieder stellen die grünen Punkte dar, die nur in den Charts einer einzigen Stadt vertreten sind. Da sich die Methode auf sehr regionale Ergebnisse spezialisieren soll, werden die Lieder aufsteigend nach ihrem Vorkommen sortiert. Lieder, die nur in einer Stadt vorkommen, stehen im Ranking höher und werden daher eher als Empfehlung genutzt. Innerhalb gleicher Vorkommen werden die Lieder nach ihrer Platzierung in den Charts ausgewählt.

Titel	**Künstler**	**Stadt**	**Platzierung**	**#**
Cocoon	Comets	Lausanne	10	1
Mordez Moi	Feed Me	El Paso	15	1
Flying Lotus	Sultan's Request	Melbourne	19	1
Fat Freddy's Drop	Dark Days	Auckland	27	1
Casper	Auf und davon	Magdeburg	14	2

Tabelle 3.4: Auszug aus Empfehlungen der Städte Charts Methode

In Tab. 3.4 wird ein Auszug der Empfehlungen eines Nutzers, der Englisch und Deutsch als Sprachen angegeben hat, gezeigt. Man erkennt die absteigende Sortierung nach deren Vorkommen # und die aufsteigende Sortierung nach deren Platzierung innerhalb der Lieder mit gleichem Vorkommen. An Interpreten mit dem Namen "Fat Freddy's Drop" lässt sich bereits ein gewisser Überraschungseffekt erahnen und sollte durch die Empfehlungen dieser Methode verstärkt auftreten.

Wie anfangs erläutert, basieren die in den folgenden zwei Kapiteln beschriebenen Methoden auf der Hörhistorie eines Last.fm Accounts, welcher daher für die Methoden benötigt wird. Da diese Informationen im laufenden Betrieb abgerufen werden, muss die abzurufende Informationsmenge beschränkt werden, um keine zu hohe Latenz zu erzeugen. Wie genau das "Live-Abrufen" abläuft wird während der Vorstellung des Prototyps in Kapitel 4 erklärt. In den folgenden zwei Kapiteln werden lediglich die darauf aufbauenden Methoden zur Empfehlungsgenerierung erläutert.

3.4 Nachbarschaft

Ein Nachbar soll laut Last.fm einen ähnlichen Musikgeschmack wie der zugehörige Nutzer haben. Um Nachbarn zu bestimmen, berechnet Last.fm einen so genannten Nachbarschaftsgrad der die Ähnlichkeit zweier Nutzer angibt. Um diese Angabe auszunutzen, greift die Methode auf die Nachbarn eines Nutzers zu. Dadurch können die Top-Lieder dieser Nachbarn abgerufen werden, um darin dem Nutzer unbekannte Lieder zu suchen und zu empfehlen.

Zuerst werden mithilfe der Last.fm API Methode *User.getNeighbours*, beschrieben in Kap. 2.1.2, die Nachbarn eines Nutzers abgerufen. Dabei werden die ersten fünf Nachbarn der Ergebnisliste gewählt, das heißt, die Nachbarn mit dem größten Nachbarchaftsgrad. Von jedem dieser Nachbarn werden mithilfe der API Methode *User.getTopTracks*, beschrieben in Kap. 2.1.1, 20 Top-Titel abgerufen. Da die Wahrscheinlichkeit dem Nutzer schon bekannte Lieder abzurufen mit zunehmender Zeitspanne steigt, denn darüber wird der Nachbarschaftsgrad berechnet, werden möglichst aktuelle Top-Titel abgerufen.

Um die aktuellsten Top-Titel des Nutzers zu erhalten, lässt sich die Methode mit dem Parameter *period* aufrufen. Man könnte als *period* immer "7day" angeben, um die Ergebnisse auf die letzte Woche zu beschränken. Dabei können jedoch leere Ergebnisse entstehen, wenn der Nutzer eine Woche lang wenig oder gar nicht in Last.fm aktiv war. Um das zu vermeiden, wird jedes Abrufen von Top-Liedern über eine Schleife durchgeführt. Diese beginnt mit der kleinsten Zeitspanne und erweitert diese schrittweise, bis die gewünschte Anzahl an Liedern akkumuliert wurde.

Nachdem von den fünf Nachbarn jeweils die 20 aktuellsten Top-Titel abgerufen wurden, ist eine Menge von 100 Liedern entstanden. Die relativ geringe Menge an Daten

folgt aus der Beschränkung der Latenzzeit. Ruft man sich die Definition eines Nachbars in Erinnerung, erkennt man, dass mit hoher Wahrscheinlichkeit Lieder oder Künstler enthalten sind, welche der Nutzer bereits kennt. Um das zu verhindern, werden die 25 aktuellsten Top-Titel des Nutzers abgerufen und von der Empfehlungsliste ausgeschlossen. Dadurch erhält man zum größten Teil Lieder, die nicht in der aktuellen Hörhistorie des Nutzers enthalten sind.

Titel	**Künstler**	**Playcount**
1904	Benjamin Francis Leftwich	29
Promise	Ben Howard	25
Everybody's On The Run	Noel Gallagher's High Flying Birds	8
Something Good	Alt-J	4

Tabelle 3.5: Auszug aus Empfehlungen der Nachbarschafts-Methode

Für die Menge der Empfehlungen muss wieder ein Ranking eingeführt werden. Dafür wird der so genannte *playcount* genutzt, der die Anzahl der Hörereignisse eines Titels angibt. Die Empfehlungen werden absteigend nach diesem Wert geordnet, wie in Tab. 3.5 zu sehen[1].

Alles in allem sollte diese Methode für jeden Nutzer sehr spezifische Empfehlungen erzeugen, die diesem mit großer Wahrscheinlichkeit gefallen. Dafür sind hauptsächlich die nutzerspezifischen Informationen verantwortlich, welche die Methode durch den direkten Zugriff auf den jeweiligen Last.fm Account erhält. Nachteil dabei ist die sehr starke Fokussierung auf die vom Nutzer bereits gemochten Musikrichtungen. Dadurch werden akkurate Empfehlungen hervorgerufen, jedoch stammen diese aus schon vertrauten Genres und erweitern dadurch den Geschmack des Nutzers wahrscheinlich nicht. Die Aktualität der abgerufenen Lieder kompensiert diesen Nachteil zu einem gewissen Grad, denn dadurch können selbst kleine Schwankungen in der Historie der Nachbarn aufgenommen und an den Nutzer weitergegeben werden.

3.5 Tags

Die Idee dieser Methode besteht darin neue Genres zu entdecken, die jedoch in Verbindung mit denen stehen, die der Nutzer bereits mag. Um diesen "weichen" Übergang zu schaffen, wird die Last.fm Hörhistorie des jeweiligen Nutzers mit Tags

[1] Genutzer Last.fm Account: Wody05 - Stand: 11.12.2012

in Verbindung gesetzt. Tags sind eine Art Metadaten eines Musiktitels. Sie beschreiben Eigenschaften des Titels durch kurze “Keywords” und dienen dazu einen Titel zu kategorisieren und wiederzufinden. Es lassen sich zu einem Titel der als “Indie” kategorisiert ist beispielsweise andere Tags, wie “Australian”, “Electropop” oder “female vocalist” finden und somit neue Musikrichtungen für die Empfehlungen nutzen.Tags werden den Liedern in Last.fm durch die Nutzer selbst hinzugefügt und lassen sich mithilfe der API abrufen.

Bis zur finalen Empfehlungsliste werden einige Schritte durchgeführt. Der erste Schritt besteht darin die Titel des Nutzers mit Tags zu verknüpfen. Dazu werden die zehn aktuellsten Top-Titel des Nutzers, mit der im vorherigen Kapitel erklärten Schleife, abgerufen. Zusätzlich wird die Methode *Track.getTopTags*, beschrieben in Kap. 2.1.4, genutzt, um die zugehörigen Tags zu erhalten. Das Ergebnis ist eine Liste von Liedern und deren Tags, bildlich dargestellt in Tab. 3.6.

#	**Titel**	**Künstler**	**Tags**
1	Plans	Birds Of Tokyo	rock australian alternative
2	I Don’t Think So	Priscilla Ahn	folk female-vocalist acoustic singer-songwriter
3	Ghosts	The Presets	rock australian synthetic
...			
10	John Mayer	Clarity	pop singer-songwriter folk

Tabelle 3.6: Auszug aus einem Zwischenergebnis der Tag-Methode

Aus dem in Tab. 3.6 dargestellten Zwischenergebnis werden im nächsten Schritt alle Tags extrahiert. Da diese Methode gezielt Tags sucht, die sich vom bereits Bekannten absetzen, werden populäre Tags wie Rock, Pop, Metal, Techno, Rap, 80s, 90s, ... aussortiert. Danach wird das Vorkommen eines jeden Tags gezählt, um die drei am öftesten vorkommenden Tags zu bestimmten. Im obigen Beispiel wären die Tags “australian”, “singer-songwriter” und “folk” bestimmt worden[1]. Für

[1]Tab. 3.6 zeigt nur einen Teil des verwendeten Zwischenergebnisses.

diese Tags werden, mithilfe der API Methode *Tag.getTopTracks*, die jeweiligen Top-Musiktitel abgerufen. Die API Methode, wie bereits in Kap. 2.1.5 beschrieben, ruft die Hörercharts des jeweiligen Tags ab.

Als Ranking werden die Platzierung der Lieder in den jeweiligen Tag Charts gewählt, wobei kein Tag bevorzugt, sondern alle gleich behandelt werden. Die Menge an Titeln wird, wie auch bei der vorherigen Methode, auf 100 beschränkt. Dadurch wird die Latenzzeit während dem Betrieb bei durchschnittlich zwei Minuten gehalten. Um die Anzahl schon bekannter Lieder zu verringern und den Vorteil eines direkt zugänglichen Last.fm Accounts auszunutzen, werden die 25 aktuellsten Top-Titel des Nutzers aus der Empfehlungsmenge entfernt.

Titel	**Künstler**	**Platzierung**	**Tag**
Sweet Disposition	The Temper Trap	3	australian
Hallelujah	Jeff Buckley	4	singer-songwriter
Crave you	Flight Facilities	4	australian
Skinny Love	Bon Iver	6	folk

Tabelle 3.7: Auszug aus Empfehlungen der Tag-Methode

Tab. 3.7 zeigt einen Ausschnitt aus einer Empfehlungsliste[2] der Tag Methode. Man erkennt, dass aus anfänglich "groben" Tags, wie Rock oder Pop, außergewöhnlichere Tags, wie "australian" oder "singer-songwriter", abgeleitet wurden. Die Empfehlungen dieser Methode sollten den Musikgeschmack auf eine "sanfte" Art erweitern und so den Serendipity Effekt verstärkt beim Nutzer hervorrufen.

Zusammenfassend sollten die beschrieben Methoden Empfehlungen generieren, die möglichst positive Reaktionen hervorrufen und dabei öfter unbekannt sind oder überraschend auf den Nutzer wirken. Die Methoden wurden im Prototyp des Recommender Systems implementiert, welcher im Folgenden Kapitel beschrieben wird. Mithilfe dieses Prototypen wurde eine Evaluation durchgeführt, um die Effekte der vorgestellten Methoden bestätigen zu können. Auf die genauen Ergebnisse der Evaluation wird in Kapitel 6 eingegangen.

[2]Genutzer Last.fm Account: Wody05 - Stand: 11.12.2012

4 Der Prototyp: Song Stumbler

Um die vorgestellten Methoden und ihre Empfehlungen in Realität zu testen, wurde der Prototyp *Song Stumbler* implementiert. *Song Stumbler* ist ein Music Recommender System in Form einer Web Applikation. Dieses wurde mithilfe von Java, speziell der Java Server Pages[1] Technologie, erstellt und wird im Folgenden vorgestellt.

4.1 Überblick

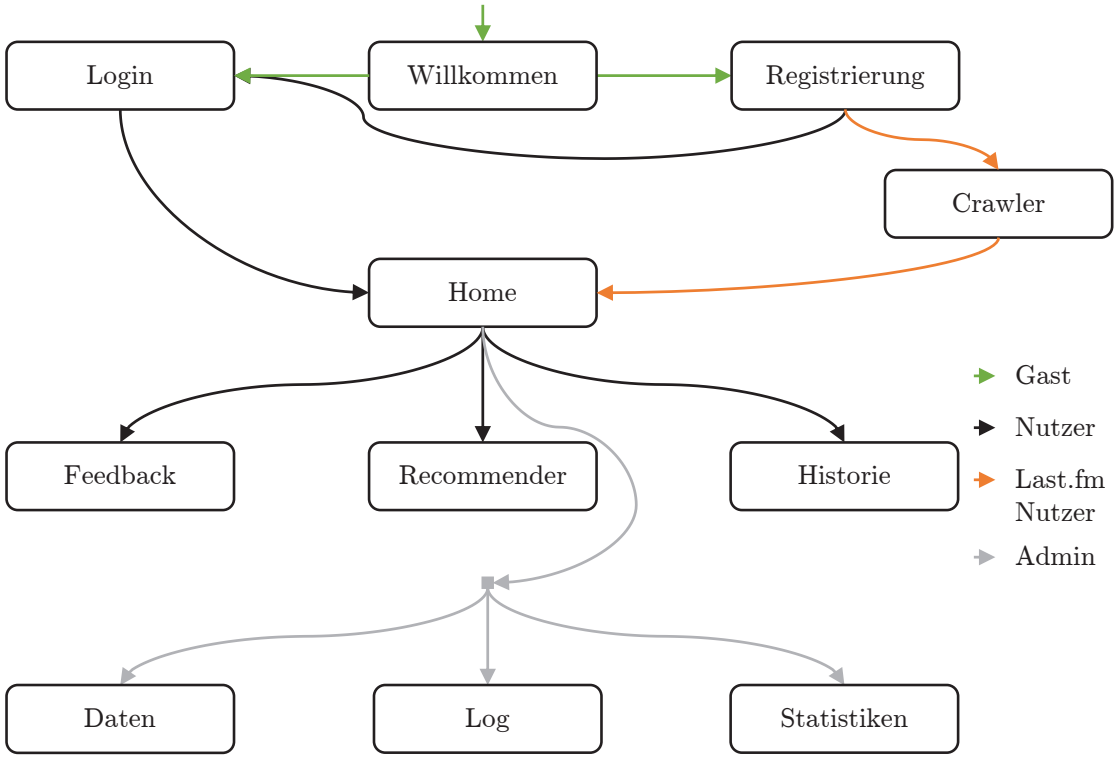


Abbildung 4.1: Aufbau des Prototyps

[1] http://www.oracle.com/technetwork/java/javaee/jsp - Stand: 11.12.2012

Abb. 4.1 zeigt den Aufbau des Prototypen nach Unterseiten gegliedert und die unterschiedlichen Zugriffsmöglichkeiten der jeweiligen Nutzergruppen. Jeder betritt zu Beginn, als Gast, die Willkommensseite. Hier wird eine kurze Erklärung zur Funktion des Prototyps angezeigt und der Zugriff auf Login oder Registrierung ermöglicht.

4.1.1 Registrierung & Login

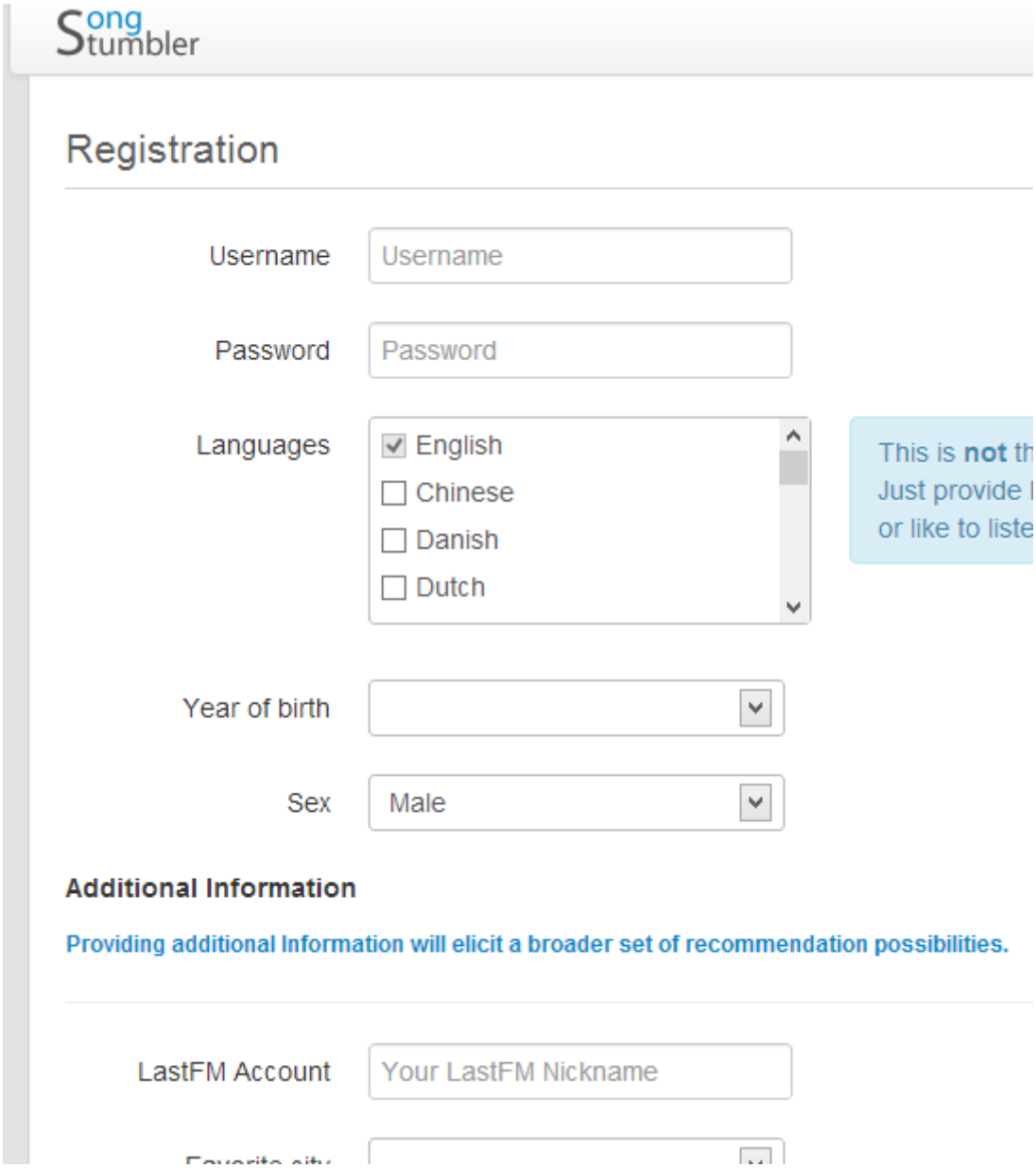


Abbildung 4.2: Ausschnitt der Registrierungseite

In Abb. 4.2 ist die Registrierungsseite des Prototyps dargestellt. Hier können sich Gäste einen Nutzeraccount anlegen und die benötigten Informationen zur Verfügung stellen. Das Benutzermodell wurde bereits in Kapitel 3 vorgestellt. Ein Nutzer wird durch den angegebenen *Username* eindeutig identifiziert. Mehrere Nutzer mit gleichem *Usernamen* werden somit nicht zugelassen. Das Passwort ist für jeden Nutzer frei wählbar und unterliegt keinen Beschränkungen. Es wird durch Sicherheitsmaßnahmen geschützt, diese werden in Kap. 4.4 vorgestellt. Es folgen Eingabemöglichkeiten für die weiteren Informationen des Nutzers, wie zum Beispiel Sprachen, Geburtsjahr oder Last.fm Account. Diese Informationen werden, für die in Kapitel 3 vorgestellten Methoden, benötigt.

Nach jedem Registrierungsversuch werden die vom Nutzer angegebenen Informationen überprüft. Gibt ein Nutzer beispielsweise einen Last.fm Account an, wird dieser auf Gültigkeit geprüft, um nicht erreichbare oder gesperrte Accounts zu vermeiden. Bei fehlerhaften Eingaben wird eine Warnmeldung dargestellt.

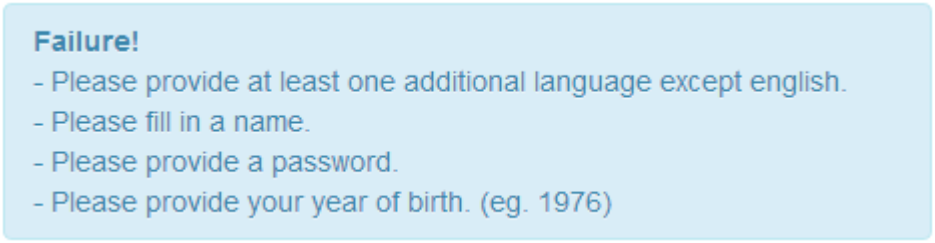


Abbildung 4.3: Beispiel einer Warnmeldung

Abb. 4.3 zeigt die Darstellung dieser Warnungen, welche im gesamten Prototyp eingesetzt werden, um den Nutzer auf Fehler aufmerksam zu machen.

Nachdem alle Informationen korrekt angegeben wurden gibt es zwei Varianten. Wurde ein Last.fm Account angegeben, müssen von diesem Informationen abgerufen werden. Dazu wird der Nutzer zum “Live-Crawler” weitergeleitet, der in Kap. 4.2 genauer erklärt wird. Ist das Abrufen abgeschlossen, wird der Nutzer auf die so genannte Home-Seite weitergeleitet.

Nutzer, die keinen Last.fm Account angeben, müssen sich einloggen und werden anschließend zur Home-Seite weitergeleitet. Darüber hinaus werden auch bereits registrierte Nutzer, die sich lediglich neu einloggen, auf diese Seite weitergeleitet.

4.1.2 Home

Die Home-Seite bildet die zentrale Seite des Prototyps, denn jeder angemeldete Nutzer findet sich zuerst auf dieser Seite wieder. Hier finden Nutzer, die den Recommender zum ersten Mal verwenden, wichtige Informationen. Diese Informationen werden in Form eines Einführungstextes dargestellt. Sie enthalten Erklärungen zu unterschiedlichen Evaluationsmöglichkeiten einer Empfehlung, Auswirkungen des Nicht-Bewertens und weiterer Aspekte, die für Nutzer vor dem ersten Start erklärungsbedürftig scheinen. Die Einführung bleibt auch nach mehrmaligem Benutzen des Recommenders auf der Home-Seite bestehen, um bei möglicherweise auftretenden Unklarheiten ein wiederholtes Lesen zu ermöglichen. Der gesamte Text findest sich in Anhang A.

Die in Abb. 4.4 dargestellte Kopfzeile findet sich auf allen Unterseiten des Prototyps, ausgenommen den Gast-Seiten[1].

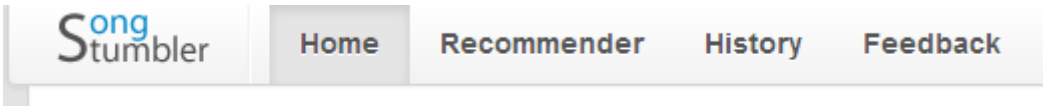


Abbildung 4.4: Kopfzeile für angemeldete Nutzer

Über die Kopfzeile lassen sich alle für den Nutzer zugänglichen Seiten aufrufen. Diese werden in den folgenden Kapiteln genauer betrachtet. Ist man als Admin eingestuft erweitert sich die Kopfzeile um einige Unterseiten, die im späteren Abschnitt "Admin-Seiten" beschrieben werden.

4.1.3 Recommender

In diesem Kapitel wird der Aufbau und der Ablauf des Recommenders erklärt. Jeder angemeldete Nutzer erhält, beim Aufruf des Recommenders, eine Liste an Empfehlungen. Abb. 4.5 zeigt einen Ausschnitt[2] einer solchen Empfehlungsliste.

Eine Liste besteht aus den Empfehlungen aller verfügbaren, in Kapitel 3 vorgestellten Methoden. Jede Methode erhält in den ersten zwei Listen vier Plätze, um Musiktitel zu empfehlen. Für Nutzer mit Last.fm Account ergeben sich 20 Empfehlungen pro Liste, da fünf Methoden zur Empfehlungsgenerierung genutzt werden.

[1] Seiten die in Abb. 4.1 durch grüne Wege erreichbar sind
[2] Teil einer Liste des Nutzers: Anna - Stand: 11.12.2012

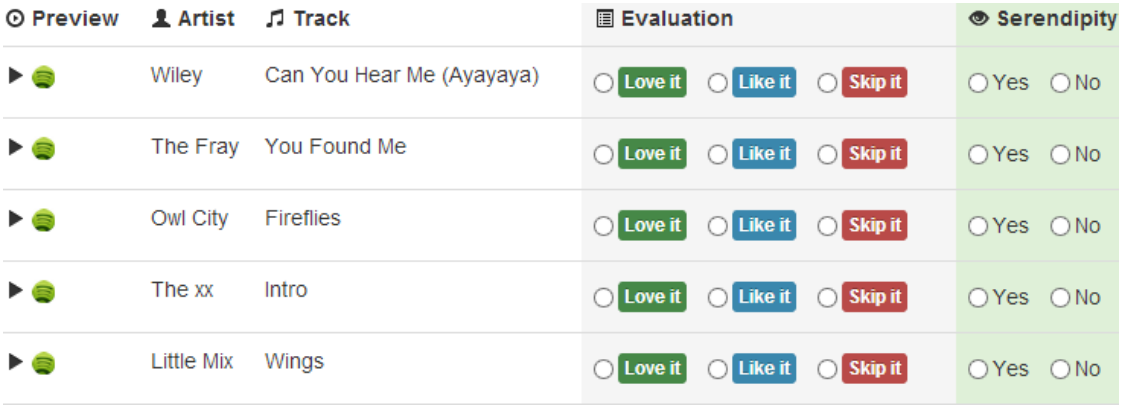


Abbildung 4.5: Ausschnitt einer Empfehlungsliste des Recommenders

Für Nutzer ohne Last.fm Account ergeben sich zwölf Empfehlungen pro Liste, da nicht alle Methoden genutzt werden können.

Für jede Empfehlung stehen dem Nutzer die folgenden Bewertungsmöglichkeiten zur Verfügung.

- **Love it**: Der Nutzer würde die Lautstärke erhöhen, wenn die Empfehlung im Radio gespielt werden würde. Stellt eine sehr positive Bewertung dar und wird durch zwei Punkte für die Empfehlung repräsentiert.
- **Like it**: Der Nutzer würde die Lautstärke unverändert lassen, wenn die Empfehlung im Radio gespielt werden würde. Stellt eine positive Bewertung dar und wird durch einen Punkt repräsentiert.
- **Skip it**: Der Nutzer würde den Sender wechseln, wenn die Empfehlung im Radio gespielt werden würde. Stellt eine negative Bewertung dar und wird durch Abzug eines Punktes repräsentiert.

Darüber hinaus kann ein Nutzer eine Empfehlung unbewertet lassen. Dies wird durch die Wertung null repräsentiert. Zusätzlich wird, wie in Abb. 4.5 zu sehen, für jede Empfehlung eine Serendipity Bewertung ermöglicht. Der Nutzer kann damit den Überraschungseffekt der Empfehlung mit *Ja* oder *Nein* bewerten.

Um die Möglichkeit des Nicht-Bewertens einzuschränken wurden zwei Regeln für die Empfehlungslisten festgelegt.

1. Es muss mindestens die Hälfte der Empfehlungen einer Liste bewertet werden, um diese Absenden zu können und damit eine neue Liste zu erhalten.

2. Für jede bewertete Empfehlung muss auch der Serendipity Effekt bewertet werden.

Wurde eine Liste nicht ausreichend bewertet, wird eine Warnmeldung, analog zu Abb. 4.3, dargestellt. Bewertungen, die vor dem Absenden angegeben wurden, bleiben in der Liste eingetragen, jedoch werden diese erst nach dem ausreichenden Bewerten der Liste permanent gespeichert. Verlässt ein Nutzer die Seite, ohne die aktuelle Liste ausreichend zu bewerten, wird diese beim nächsten Besuch wieder angezeigt.

Hat ein Nutzer eine Liste ausreichend bewertet, kann er diese absenden. Die Bewertungen werden dadurch in der Datenbank gespeichert und es kann eine neue Empfehlungsliste generiert werden. Wie anfangs erwähnt, erhalten alle Methoden in den ersten zwei Listen jeweils vier Plätze, um Empfehlungen anzubieten. Mit der dritten Liste werden, auf Basis der vom Nutzer abgegebenen Bewertungen, die Plätze der Methoden neu verteilt, um damit die Listen auf den Geschmack des Nutzers anzupassen. Methoden, die gemocht werden bekommen dabei mehr Plätze, Methoden, die nicht gemocht werden verlieren Plätze. Der genaue Ablauf der Listenanpassung wird in Kap. 4.3 erläutert.

Bewertet ein Nutzer sehr viele Listen, kann es dazu kommen, dass eine Methode alle zur Verfügung stehenden Empfehlungen bereits ausgegeben hat. Ist dies der Fall, so werden alle nicht bewerteten Empfehlungen freigegeben. Das heißt, diese Musiktitel können bei der nächsten Generierung einer Empfehlungsliste wieder genutzt werden. Da der “älteste” nicht bewertete Musiktitel durch die entsprechende Methode zuerst ausgewählt wurde, wird dies auch nach dem Freigeben wieder der Fall sein. Dadurch wird automatisch verhindert, dass erst kürzlich nicht-bewertete Lieder zu schnell in eine neue Empfehlungsliste aufgenommen werden.

Damit ein Nutzer überhaupt eine Bewertung abgeben kann, muss es eine Möglichkeit geben, die empfohlenen Lieder anhören zu können. Dafür wurden im Prototyp zwei Varianten zur Verfügung gestellt, welche in Abb. 4.5 in der Spalte “Preview” zu sehen sind. Die eine Variante wird für jede Empfehlung im Voraus berechnet und durch das “Play” Symbol repräsentiert. Klickt ein Nutzer auf das Symbol, wird dieser entweder auf Amazon[1] oder YouTube[2] weitergeleitet, um eine Vorschau des Musiktitels anhören zu können. Die andere Variante nutzt den Musik-Streamingdienst Spotify[3]

[1] http://www.amazon.de - Abgerufen am 11.12.2012
[2] http://www.youtube.de - Abgerufen am 11.12.2012
[3] http://www.spotify.de - Abgerufen am 11.12.2012

und öffnet den Desktop Client mit einer Suche nach dem gewünschten Lied. Dafür wird ein installierter Spotify Desktop Client vorausgesetzt. Genauere Informationen zu den Vorschaumöglichkeiten werden in Kap. 4.2.1 gegeben.

4.1.4 Historie

Die Historienseite bietet zum einen eine so genannte "Community's Favorites"-Liste und zum anderen einen Überblick aller bewerteten Lieder des jeweiligen Nutzers.

Abbildung 4.6: Community's Favorites vom 11.12.2012

Die "Community's Favorites" bestehen aus den fünf am besten bewerteten Empfehlungen. Dazu werden die Punkte aller Empfehlungen über alle Nutzer aufsummiert und die mit den höchsten Punktzahlen ausgewählt. Hat der Nutzer diese Titel bereits bewertet, wird eine erneute Evaluation verhindert und "Already evaluated" ausgegeben, wie in Abb. 4.6 in der Spalte "Evaluation" zu sehen. Wurde ein Titel noch nicht bewertet, werden die gleichen Bewertungsmöglichkeiten wie im Recommender zur Verfügung gestellt.

Damit ein Nutzer bereits bewertete Lieder wiederfinden kann, wird eine Auflistung aller bewerteten Empfehlungen des Nutzers angeboten. Ein Beispiel dafür ist in Abb. 4.7 zu sehen.

Der Nutzer hat die Möglichkeit nach "Loved Songs", "Liked Songs" oder "Skipped Songs" zu filtern. Dabei entsprechen die angezeigten Lieder der jeweiligen Bewertung *Love it*, *Like it* oder *Skip it*. Die Lieder sind absteigend nach ihrem Bewertungsdatum sortiert.

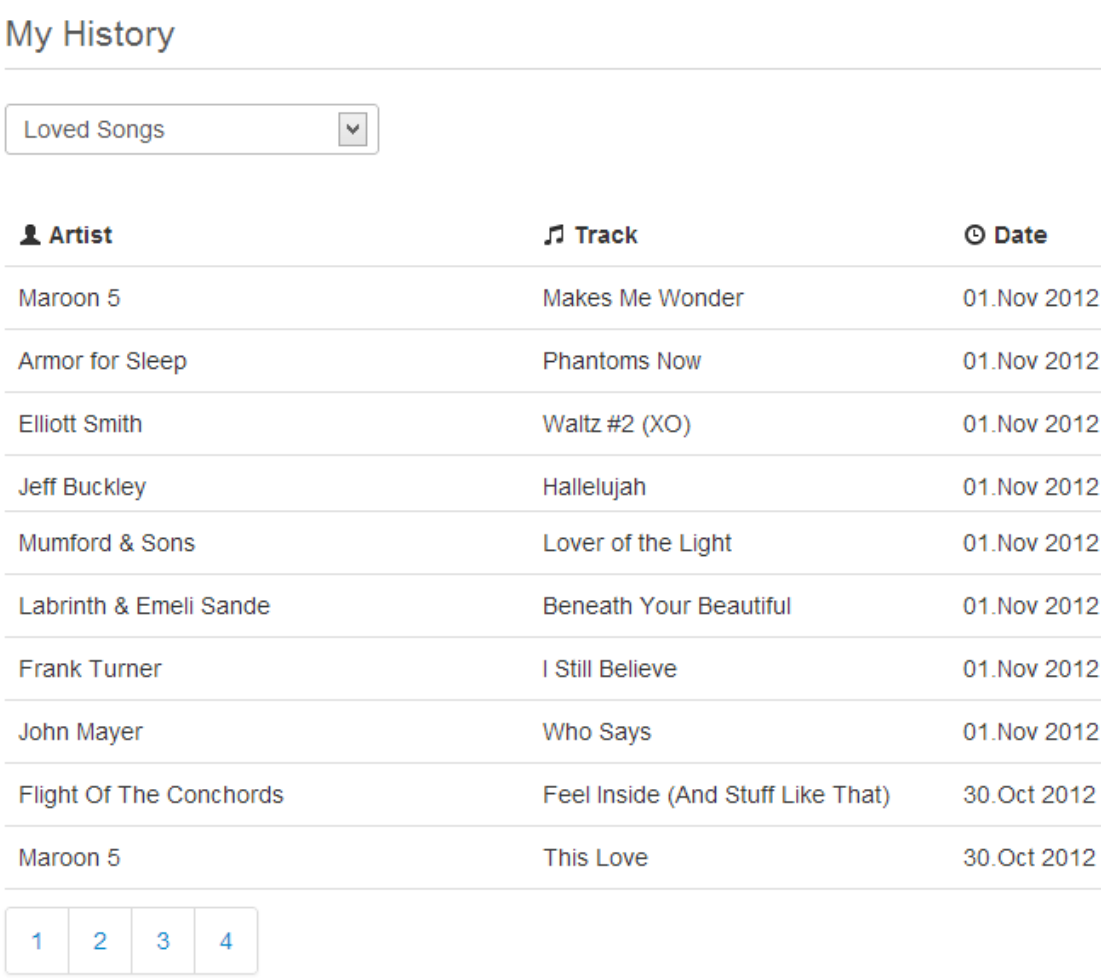


Abbildung 4.7: Auschnitt der Historie eines Nutzers

Da ein Nutzer möglicherweise sehr viele Lieder bewertet hat, wurde ein so genannter "Paginator" implementiert. Dadurch werden die Ergebnisse auf mehrere Seiten verteilt und auf jeder Seite maximal 20 Titel angezeigt.

4.1.5 Admin-Seiten

Der Prototyp stellt drei Admin Seiten zur Verfügung. Diese Seiten sind in Abb. 4.1 durch die grauen Pfade markiert und bestehen aus der Daten-Seite, auf der die gespeicherten Daten eingesehen werden können, der Statistik-Seite, auf der aktuelle Statistiken zu Bewertungen und Nutzern angezeigt werden und der Log-Seite, auf der Einträge aus der Log-Datei zu sehen sind.

Daten-Seite Die Daten-Seite bietet die Möglichkeit auf die in Kapitel 2 beschriebenen Daten zuzugreifen. Darunter die Charts der verschiedenen Länder, Jahre und Städte, sowie die Last.fm Account gebundenen Daten der Nutzer. Abb. 4.8 zeigt die Auswahlmöglichkeiten auf der Daten-Seite.

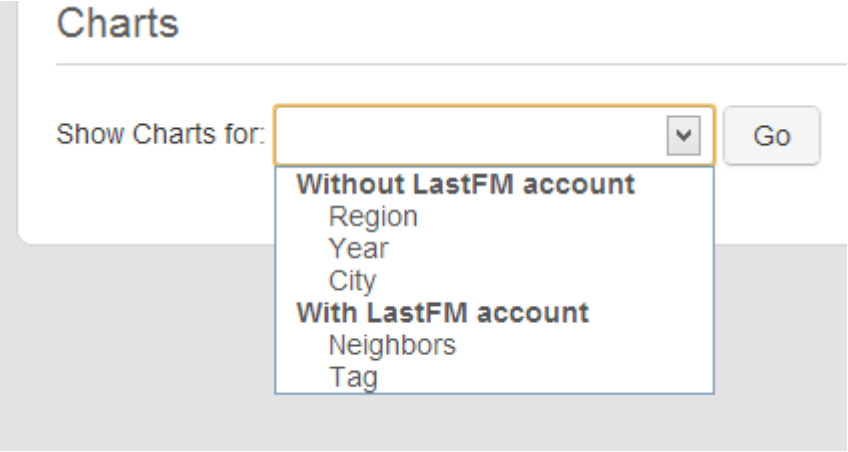


Abbildung 4.8: Auswahl an "Charts"

Alle Daten lassen sich während dem Betrieb einsehen und sind nützlich, um die Datenintegrität zu überprüfen. Es lässt sich beispielsweise überprüfen, ob die Datenerhebung einzelner Nutzer korrekt funktioniert hat oder die Daten nach der wöchentlichen Aktualisierung korrekt vorliegen.

Log-Seite Auf der Log-Seite lässt sich der Inhalt der Logdatei anzeigen, um während dem Betrieb auftretende Fehler sehen zu können. Für die Logging Funktionen des Prototypen wurde Log4j[1] genutzt. Alle Log Aktivitäten der Stufe *warn* werden in eine Datei geschrieben, niedrigere Prioritäten werden nur in den Standard Output, wie zum Beispiel die Konsole, geschrieben. Um dies zu realisieren, wurde für jede Klasse der serverseitigen Implementierung ein Logger angelegt, der mögliche Fehler oder wichtige Informationen ausgibt.

Um möglicherweise auftretende Fehler in den *.JSP*[2] Dateien zu bemerken, wurde ein zentraler "Error Handler" in Form eines *Servlets*[3] implementiert. Tritt bei den Nutzern ein Fehler auf, werden diese automatisch an den "Error Handler" weiter-

[1] http://logging.apache.org/log4j/2.x/ - Stand: 11.12.2012
[2] HTML Konstrukt mit Java Code Abschnitten
[3] Verarbeiten Anfragen vom Client an den Server, sowie Antworten vom Server an den Client

geleitet. Dieser schreibt die Details des aufgetretenen Fehlers in die Log Datei und weist den Nutzer auf einen aufgetretenen Fehler hin.

Aus Sicherheitsgründen wird nicht auf den genauen Fehler hingewiesen, sondern nur auf die Tatsache, dass ein Fehler aufgetreten ist und registriert wurde. Darüber hinaus wird dem Nutzer die Möglichkeit geboten, auf die Home-Seite zurückzukehren.

Statistik-Seite Die Statistik-Seite bietet die Möglichkeit einige Informationen zu den Bewertungen und Nutzern abzurufen. Die angezeigten Werte werden während dem Aufruf der Seite berechnet und sind daher immer aktuell. Ein Ausschnitt der Statistik-Seite ist in Abb. 4.9 zu sehen.

Registered users (with last.fm account)	21
Registered users (without last.fm account)	27
Evaluated Songs	1567
Loved Songs	475
Liked Songs	560
Skipped Songs	532
Serendipitous Songs	306

Abbildung 4.9: Ausschnitt der Statistik-Seite

Besonders nützlich war die Statistik-Seite während der Evaluationsphase des Prototyps, um schnell einen Trend unter den Nutzern oder andere statistische Informationen ablesen zu können.

Bei jedem Aufruf der in diesem Kapitel beschriebenen Seiten wird überprüft, ob der zugreifende Nutzer ausreichende Privilegien besitzt. Ist dies nicht der Fall, wird der Nutzer auf die Home-Seite umgeleitet. Darüber hinaus werden in der Kopfzeile jeder Unterseite nur Seiten angezeigt, auf die der jeweilige Nutzer Zugriff hat.

In den folgenden Kapiteln werden Implementierungsdetails der wesentlichen Funktionen des Prototypen genauer betrachtet. Zuerst werden Details zu Vorgehensweisen bei der Datenerhebung erläutert. Anschließend wird die Anpassung der Empfehlungslisten des Recommenders erklärt und zuletzt einige Sicherheitsfunktionen beschrieben.

4.2 Live-Crawler

Der "Live-Crawler" dient dazu, die benötigten Informationen, der in Kap. 3.4 und Kap. 3.5 vorgestellten Methoden, während des Betriebs abzurufen. Dieser kommt im Prototyp direkt im Anschluss an die Registrierung zum Einsatz. Wurde ein Last.fm Account angegeben, wird der Nutzer nach der Registrierung zum "Live-Crawler" weitergeleitet. Dort werden 200 Lieder aus der Hörhistorie des Last.fm Accounts abgerufen. Wie in Abb. 4.10 zu sehen, wird dem Nutzer während dem Abrufen der Fortschritt, in Form eines Ladebalkens, angezeigt. Dieser wurde mithilfe von *AJAX*[1] implementiert, um die Seite nicht im Sekundentakt neu laden zu müssen. Ist das Abrufen beendet wird der Nutzer auf die Home-Seite weitergeleitet.

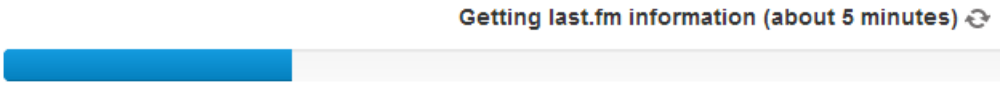


Abbildung 4.10: Ausschnitt des "Live-Crawlers"

Nach einem Vorabtest, im Rahmen weniger ausgesuchter Nutzer, ergaben sich für den Crawler folgende Anforderungen:

- **Latenzzeit** Die benötigte Zeit zum Abrufen der Daten sollte im Bereich weniger Minuten gehalten werden.
- **Nebenläufigkeit** Das Abrufen von Informationen mehrerer Nutzer muss, bis zu einem gewissen Grad, gleichzeitig ablaufen können.
- **Einmaligkeit** Das Abrufen von Informationen darf für einen Last.fm Account nur einmal durchgeführt werden.

[1] Asynchronous JavaScript and XML

Latenzzeit Die Laufzeit des Crawlers hängt in erster Linie von der Anzahl der abzurufenden Titel ab. Deshalb wurde die Zahl der Titel auf 200 beschränkt, um das Abrufen überhaupt in wenigen Minuten möglich zu machen. Der größte Zeitaufwand während dem Abrufen eines Titels entsteht durch die Berechnung eines Vorschaulinks. Um diesen Flaschenhals zu beseitigen wurden mehrerer Threads zur Berechnung der Vorschaulinks eingeführt und dadurch die Berechnungszeit mehr als halbiert.

Durch diese Maßnahmen erreicht der Crawler eine Laufzeit von ungefähr drei bis sechs Minuten für 200 Titel. Unterschiede entstehen je nach Internetverbindung und Auslastung der Last.fm Server.

Nebenläufigkeit Um mehrere Nutzer, die sich zeitnah registrieren, verarbeiten zu können, wird für jeden dieser Nutzer ein Thread angelegt. Dadurch muss niemand auf das Abrufen der Informationen des vorherigen Nutzers warten.

Um dies zu realisieren, wurde eine *Map* Datenstruktur genutzt, in der Schlüssel-Wert Paare gespeichert werden. Sobald ein Nutzer den Crawler aufruft, wird von der zugehörigen Session der Name abgefragt und als Schlüssel in die *Map* gespeichert. Anschließend wird für den Nutzer ein Thread gestartet und als zugehöriger Wert gespeichert. Durch *AJAX* wird alle zwei Sekunden der Fortschritt abgefragt. Dazu wird der Name des Nutzers in der Map gesucht und vom zugehörigen Thread der Fortschritt abgefragt. Sobald der Thread beendet ist und damit alle benötigten Daten gespeichert wurden, wird der Schlüssel aus der *Map* entfernt.

Einmaligkeit Durch diese Anforderung soll sichergestellt werden, dass der Thread eines Nutzers nicht mehrmals gestartet wird. Ohne eine Überprüfung würde der Thread zum Beispiel durch das Aktualisieren des Browsers erneut gestartet werden. Um dies zu verhindern wird bei jedem Aufruf des Crawlers zuerst überprüft, ob der Name des Nutzers bereits in der Map existiert. Ist dies der Fall wird kein neuer Thread gestartet.

Das einmalige Einsetzen des Crawlers, nach der Registrierung, ist für die Nutzer von Vorteil, denn dadurch müssen die Daten nicht während jeder Empfehlungsgenerierung einzeln abgerufen werden. Das gesamtheitliche Abrufen aller benötigten Informationen beschleunigt folglich den späteren Betrieb der Empfehlungsgenerierung und somit das gesamte Recommender System.

4.2.1 Song Previews

Der Prototyp bietet zwei verschiedene Wege zum Probehören eines Titels. Zum einen wird während der Datenerhebung ein Vorschaulink berechnet und zusammen mit dem jeweiligen Titel gespeichert. Zum anderen wird während des Abrufens einer Empfehlungsliste für jeden Titel ein Link für eine Suche im Spotify Desktop Client erstellt.

Die erste Methode berechnet die Vorschaulinks folgendermaßen. Zuerst werden Titel und Künstler zu einer Suchanfrage auf Amazon MP3 zusammengesetzt. Es entsteht beispielsweise für den Titel “State of Grace” von “Taylor Swift” die Anfrage

http://www.amazon.de/s/ref=nb_sb_noss?url=search-alias%3Ddigital-music&field-keywords=Taylor+Swift+State+Of+Grace.

Die Antwort dieser Suchanfrage wird über einen HTML Parser auf Erfolg oder Misserfolg überprüft. War die Suche erfolgreich, existiert eine Vorschau des gewünschten Titels und der Link wird gespeichert. War sie nicht erfolgreich, wird eine YouTube Suchanfrage als Link gespeichert. Da das Überprüfen der Antwort über einen HTML Parser relativ zeitaufwändig ist, werden die Vorschaulinks nach dem Abrufen der Musiktitel nebenläufig durch mehrere Threads berechnet.

Die zweite Methode nutzt den Musik-Streamingdienst Spotify, welcher so genannte “Advanced Search Queries” anbietet. Für diese Suchanfragen wird eine Web-Syntax angeboten, über die Suchanfragen direkt an den Spotify Desktop Client gestellt werden können. Die Web-Syntax wird durch einen Link repräsentiert, der mit “Spotify:” eingeleitet wird. Sucht man beispielsweise wieder nach dem Titel “State of Grace” von “Taylor Swift” ergibt sich folgender Link.

spotify:search:track%3A%22State+Of+Grace%22artist%3A%22Taylor+Swift%22.

Durch *track* und *artist* werden Titel und Künstler angegeben. Die Suche lässt sich durch das Hinzufügen von *genre* oder *year*, für ein bestimmtes Jahr, weiter spezifizieren. Die mit % beginnenden Zeichenfolgen stellen dabei URL-kodierte Symbole dar. %3A stellt beispielsweise das Symbol “:” dar.

4.3 Dynamische Anpassung

Wie bereits in Kap. 4.1.3 erwähnt, werden die Empfehlungslisten des Recommenders, auf Basis der Bewertungen des jeweiligen Nutzers, angepasst. Die Idee der Anpassung besteht darin, die Empfehlungslisten besser auf den Nutzer abzustimmen. Mag dieser beispielsweise keine Lieder einer bestimmten Methode, so würde es wenig Sinn machen dieser Methode in jeder Liste vier Plätze zur Verfügung zu stellen. Stattdessen könnten die Plätze an andere Methoden, deren Empfehlungen dem Nutzer besser gefallen, abgegeben werden. Diese Idee soll durch die dynamische Anpassung umgesetzt werden.

Damit die Anpassung auf einer ausreichenden Menge an Bewertungen arbeiten kann, startet diese erst mit Generierung der dritten Empfehlungsliste. Um die Methoden leichter identifizieren zu können, wird Folgendes festgelegt.

Metrik	Jahres Charts	Länder Charts	Städte Charts	Tag	Nachbarschaft
ID	1	2	3	4	5

Tabelle 4.1: Identifikation der Methoden

Die folgenden Definitionen betreffen jeweils einen einzelnen Nutzer und dessen Bewertungen einer einzelnen Methode $i \in \{1, 2, 3, 4, 5\}$. Sei B_k die Bewertung des *k-ten* Titels und n die Anzahl aller bewerteten Titel eines Nutzers. Die Wertung der Methode i wird in Formel 4.1 definiert.

$$Score_i = \frac{\text{Erreichte Wertungssumme}}{\text{Max. Wertungssumme}} = \frac{B_1 + B_2 + ... + B_n}{2 \cdot n} = \frac{\sum_{k=1}^{n} B_k}{2 \cdot n} \quad (4.1)$$

Da die maximale Bewertung einer Empfehlung 2 entspricht, berechnet sich die maximal erreichbare Bewertungssumme durch $2 \cdot \#\text{BewerteterTitel} = 2 \cdot n$.

Um die aktuellen Präferenzen des Nutzers verstärkt zu beachten, werden die Bewertungen der letzten zwei Listen doppelt gewichtet. Dafür muss die Formel erweitert werden. Seien

- L die Anzahl bewerteter Listen,
- n_l die Anzahl bewerteter Titel in Liste l und
- B_{k_l} die Bewertung des *k-ten* Titels der *l-ten* Liste.

Die gewichtete Wertung einer Methode i wird in Formel 4.2 angegeben.

$$\begin{aligned} Score_i &= \frac{B_{1_1} + B_{2_1} + ... + B_{1_2} + ... + B_{n_{L_L}} + B_{1_{L-2}} + ... + B_{n_{L_L}}}{2 * (\sum_{l=1}^{L} n_l + \sum_{l=L-2}^{L} n_l)} \\ &= \frac{\sum_{l=1}^{L} \sum_{k=1}^{n_l} B_{k_l} + \sum_{l=L-2}^{L} \sum_{k=1}^{n_l} B_{k_l}}{2 * (\sum_{l=1}^{L} n_l + \sum_{l=L-2}^{L} n_l)} \end{aligned} \tag{4.2}$$

Metrik	Bewertungen/Liste	Wertung	Gewichtet
1	$\{2,1,1,2\}\{2\}\{1,-1,-1,-1\}$	$\frac{6}{18} = 0,38$	$\frac{6}{28} = 0,21$
2	$\{1,1,2\}\{1,1\}\{2,1,-1,1\}$	$\frac{9}{18} = 0,5$	$\frac{14}{30} = 0,466$
3	$\{1,-1\}\{1,-1,1,-1\}\{1,2\}$	$\frac{3}{16} = 0,1875$	$\frac{6}{20} = 0,3$
4	$\{-1,-1,-1\}\{1,-1,-1\}\{-1,-1\}$	$\frac{-6}{16} = -0,375$	$\frac{-9}{26} = -0,346$
5	$\{-1,1,1,-1\}\{1,-1,1,1\}\{1,-1,-1\}$	$\frac{1}{22} = 0,045$	$\frac{6}{36} = 0,166$

Tabelle 4.2: Beispiel zur Berechnung der Wertungen

Tab. 4.2[1] zeigt die berechneten Wertungen, sowohl gewichtet als auch ungewichtet, während der Generierung einer vierten Liste. Man erkennt, dass die Bewertungen der Methode 1 einen eher negativen Trend verfolgen. Daher ist die gewichtete Wertung auch deutlich kleiner als die ungewichtete Wertung. Methode 2 wurde ausgeglichen bewertet, daher unterscheiden sich gewichtete und ungewichtete Wertung kaum. Die Bewertungen der Methode 3 verfolgen einen positiven Trend, wodurch die gewichtete Wertung im Vergleich zur ungewichteten deutlich zugenommen hat.

Nachdem alle Wertungen berechnet wurden, folgt die Allokation der Plätze. Dazu wird die Zahl der zu verteilenden Plätze auf 20 oder 12 festgelegt, je nachdem ob der Nutzer einen Last.fm Account angegeben hat oder nicht. Weiter wird festgelegt, dass eine Methode immer mindestens einen Platz behalten muss, damit der Nutzer die Chance hat die betreffende Methode wieder zu präferieren.

Es folgen einige Schritte bis zur fertigen Allokation der Plätze. Im ersten Schritt werden die berechneten Wertungen auf folgende Spezialfälle überprüft.

Negative Wertung Gilt für die Wertung einer Methode $Score_i \leq 0$, wird diese aus dem weiteren Prozess der Allokation ausgeschlossen. Dazu wird der Methode ein Platz in der nächsten Liste zugeordnet und die zu verteilenden Plätze

[1] Beispielhafte Bewertungen

entsprechend verkleinert. Dies ist zum Beispiel bei Methode 4 in Tab. 4.2 der Fall.

Gleiche Wertungen Liegt keine wirkliche Präferenz des Nutzers vor, wird jeder Methode der Standardwert von vier Plätzen zugewiesen. Das liegt vor, wenn gilt: $\forall i, k\ Score_i = Score_k \vee \forall i\, Score_i \leq 0$ wobei $i, k \in \{1, 2, 3, 4, 5\}$.

Nachdem die Spezialfälle geregelt wurden, wird die Summe W der verbleibenden Wertungen berechnet. Für das in Tab. 4.2 eingeführte Beispiel würde das bedeuten $W = \sum Wertungen = 0,21 + 0,466 + 0,3 + 0,166 = 1,142$. Da Methode 4 in diesem Beispiel bereits ein Platz zugewiesen wurde, wird die Wertung nicht in die Summe aufgenommen und die Anzahl der zu verteilenden Plätze auf 19 verringert. Die Plätze einer Methode i werden in Formel 4.3 definiert.

$$Places_i = \lceil \frac{Score_i}{W} \cdot Places - 0.5 \rceil \tag{4.3}$$

Das Subtrahieren von 0.5 dient zur Festlegung der Rundungsgrenze, wodurch die Gaußklammer in diesem Fall dem üblichen Auf- und Abrunden entspricht. Berechnet man die Plätze wieder für das Beispiel in Tab. 4.2, ergibt sich folgende Allokation der Plätze.

Methode	1	2	3	4	5
Plätze	$3,4 \approx 3$	$7,75 \approx 8$	$4,9 \approx 5$	1	$2,76 \approx 3$

Tabelle 4.3: Finale Allokation der Plätze

In diesem Beispiel ergeben sich exakt 20 verteilte Plätze. Zum Schluss müssen zwei Spezialfälle beachtet werden, bevor die Allokation abgeschlossen ist.

Rundungsfehler Es kann in seltenen Fällen zu Rundungsfehlern kommen. Das heißt, es werden zu viele oder zu wenige Plätze verteilt. Ist dies der Fall, wird der Methode mit den meisten Plätzen entweder ein Platz genommen, oder hinzugefügt.

Keine Plätze Besitzt eine Methode ein sehr kleine, positive Wertung kommt es vor, dass dieser Methode keine Plätze zugewiesen werden. Das widerspricht den Vorgaben und die Methode erhält deshalb einen Platz. Sind danach zu viele Plätze verteilt, wird der Methode mit den meisten Plätze ein Platz genommen.

4.4 Security

Um bekannte Sicherheitsrisiken vieler Webseiten, wie *Cross Site Scripting* oder *SQL Injections*, zu minimieren, wurden im Prototyp einige Sicherheitsmechanismen eingesetzt.

Zum Beispiel werden Nutzereingaben ausschließlich über den *c:out* Befehl der *JSTL*[1] auf der Seite ausgegeben. Diese Methode verhindert *Cross Site Scripting* Attacken, wie beispielsweise Javascript Aufrufe durch Nutzereingaben. Das wird durch das Umwandeln von XML-Sonderzeichen gewährleistet. Zum Beispiel wird das Symbol "<" umgewandelt in die HTML Repräsentation "<", wodurch Scripte nicht als solche interpretiert werden.

Sollen Nutzereingaben nicht ausgegeben sondern in die Datenbank gespeichert werden, können so genannte *SQL Injections* durchgeführt werden. Dabei versuchen Angreifer Datenbankeinträge durch manipulierte SQL Befehle zu verändern. Um das zu verhindern, arbeitet der Prototyp mit *Prepared Statements*[2] und erlaubt keine SQL spezifischen Zeichen in Nutzereingaben. Dadurch wird die Manipulation von SQL Befehlen verhindert.

Um Angreifern möglichst wenig Informationen zu bieten, werden nicht die automatisch generierten Fehlermeldungen verwendet. Diese Fehlermeldungen können für Angreifer oft wertvolle Informationen über Architektur oder Code Ausschnitte beinhalten und dadurch zum Sicherheitsrisiko werden. Für den Prototyp wurde deshalb ein "Error Handler" implementiert. Dieser verarbeitet möglicherweise auftretende Fehler, indem der Nutzer lediglich auf das Auftreten eines Fehlers hingewiesen wird. Detaillierte Informationen werden in eine Logdatei geschrieben, jedoch nicht an den Nutzer, beziehungsweise möglichen Angreifer, ausgegeben.

Darüber hinaus werden Passwörter nicht als Klartext in der Datenbank gespeichert oder überprüft. Dazu wird eine Hashfunktion eingesetzt, die für ein beliebiges Passwort einen Hashwert berechnet. Um möglichst wenige Kollisionen zu erzeugen, wird der Hash-Algorithmus *SHA-256*[3] verwendet. Dieser bietet eine sehr geringe Kollisionswahrscheinlichkeit, was bedeutet, dass unterschiedliche Passwörter praktisch nie gleiche Hashwerte erzeugen.

[1] Java Standard Tag Library
[2] Spezifische Werte des SQL Befehls werden als getrennte Parameter übergeben.
[3] Secure Hash Algorithm. Hashwerte bestehen aus 256 Bits.

5 Verwandte Arbeiten

Arbeiten im Bereich Recommender Systeme fokussierten in der Vergangenheit in erster Linie den "Accuracy" Aspekt. Diese Arbeit soll davon abweichen und stattdessen den Serendipity Aspekt eines Musik Recommenders verstärkt betrachten. Verwandte Arbeiten sind Forschungen, welche ebenfalls versuchen alternative Faktoren zur Empfehlungsgenerierung zu untersuchen. Ideen zur grundlegenden Zielsetzung dieser Arbeit folgen dem Trend aktueller Nachforschungen, wie Auralist [6] oder früheren Konzepten von Herlocker [8], welcher die Idee aufbrachte andere Aspekte neben "Accuracy" zu beachten.

Auralist versucht Probleme wie die "filter bubble", nach Pariser [4], und dadurch entstehende langweilige Empfehlungen zu verhindern. Dabei werden bereits erwähnte Begriffe, wie *Listener Diversity* oder *Declustering*, das Empfehlen von Titeln die außerhalb des aktuellen Musikgeschmacks eines Nutzers liegen, eingeführt. Diese Konzepte dienten bei den vorgeschlagenen Methoden als Grundlage.

Weitere verwandte Arbeiten sind Ideen zu den unterschiedlichen Aspekten von Recommender Systemen nach Vargas [7]. Es werden mit Auswahl, Entdeckung und Relevanz drei grundlegende Konzepte der Vielfältigkeit und Neuartigkeit von Empfehlungen eingeführt. Auch die Ideen nach Ziegler [5] zur Vielfältigkeit von Top-N Listen waren hilfreiche Konzepte. Diese wurden aber nur sinngemäß eingesetzt, wie zum Beispiel durch das Verhindern von gleichen Künstlern in einer Empfehlungsliste.

Die in den vorangegangenen Kapiteln vorgestellten Methoden sollen, ähnlich wie die vorher erwähnten Arbeiten, den Serendipity Effekt fokussieren. Die Methoden arbeiten jedoch nicht nach "traditionellen" Arten, wie *collaborative filtering* oder *content-based filtering*. Sie generieren Empfehlungen auf Basis von Informationen des Nutzers selbst, demografischen Hintergründen und allgemeiner Beliebtheit. Zur kontinuierlichen Spezialisierung auf den Nutzer, wurde in Kap. 4.3 die dynamische Anpassung vorgestellt. Dadurch werden Empfehlungslisten, durch Allokation der Plätze immer weiter an die Präferenzen des Nutzers angepasst.

5.1 Vergleich mit klassischen Methoden

Durch das Abweichen von traditionellen Methoden wie *collaborative filtering* oder *content-based filtering* lassen sich einige Vor- aber auch Nachteile erkennen. *Collaborative filtering* generiert Empfehlungen durch das Analysieren anderer Nutzer und schließt durch die analysierten Präferenzen auf Empfehlungen für andere Nutzer. Beim *content-based filtering* wird der Inhalt von Objekten analysiert und Empfehlungen auf Basis der dadurch erhaltenen Informationen generiert. Vorteile gegenüber diesen klassischen Methoden ergeben sich daraus, dass die bekannten Problem dieser Methoden keine große Rolle für die, in dieser Ausarbeitung, vorgeschlagenen Metriken spielen. Warum das der Fall ist, wird im Folgenden erläutert.

Coldstart Coldstart ist in Recommender Systemen weit verbreitet und beschreibt das Problem Empfehlungen zu erzeugen, wenn das Recommender System erst gestartet wurde und zu wenig Daten zur Verfügung stehen. Sowohl *collaborative filtering* als auch *content-based filtering* sind auf eine gewisse Menge an Daten angewiesen, um Empfehlungen zu generieren. Diese Daten fehlen jedoch in der Startphase eines Recommender Systems.

Die in Kapitel 3 vorgeschlagenen Methoden sind von diesem Problem nicht betroffen. Sie benötigen lediglich die Informationen, die im Benutzermodell festgelegt wurden. Die Empfehlungen werden nur auf Basis der Informationen des Nutzers selbst generiert und nicht durch Analyse anderer Objekte oder Nutzer. Da diese Informationen während der Registrierung angegeben werden, arbeiten die Methoden für einen einzigen Nutzer genau so gut, wie für eine größere Menge an Nutzern.

Überspezialisierung Durch das kontinuierliche Anpassen der Empfehlungen auf die berechneten Präferenzen des Nutzers, kann es bei traditionellen Recommender Systemen zur Überspezialisierung kommen. Das heißt, der Recommender bietet eintönige Empfehlungen an, wie zum Beispiel das zehnte Lied der Musikgruppe “Coldplay” in Folge. Diese werden dem Nutzer höchstwahrscheinlich alle gefallen, jedoch kann es auch eintönig auf den Nutzer wirken, wie das “filter bubble” Konzept beschreibt.

Da ein Teil der in Kapitel 3 vorgeschlagenen Methoden auf allgemeiner Beliebtheit basiert, werden immer auch Lieder empfohlen, die nicht direkt auf den aktuellen

Geschmack des Nutzers zugeschnitten sind. Die Anzahl dieser Empfehlungen kann durch die dynamische Anpassung zwar stark verringert werden, jedoch nie komplett verschwinden. Dadurch bleibt die Möglichkeit für den Nutzer bestehen, neue Musikrichtungen zu entdecken, die möglicherweise eine höheres Maß an Zufriedenstellung hervorrufen, als der nächste Titel eines bereits bekannten Künstlers.

Berechnungen Das Berechnen ähnlicher Nutzer und die daraus folgenden Interessengruppen für das *collaborative filtering*, sowie das Berechnen ähnlicher Objekte für das *content-based filtering* ist sehr rechenintensiv und daher auch zeitaufwändig. Um diese Berechnungen nicht während des Betriebs durchführen zu müssen, werden meist große Vorberechnungen durchgeführt.

Die Methoden dieser Ausarbeitung greifen lediglich auf Informationen zur allgemeinen Beliebtheit, zum Beispiel Charts, oder andere bereits vorliegende Daten des Nutzers zu. Die dafür benötigten Berechnungen sind vergleichsweise klein und können ohne Vorberechnungen durchgeführt werden. Die benötigten Daten müssen jedoch ebenfalls im Voraus gesammelt werden.

Natürlich bringen diese "kürzeren" Berechnungen auch Nachteile mit sich. Beispielsweise wird der konkrete Musikgeschmack eines Nutzers erst durch Angabe eines Last.fm Accounts direkt berücksichtigt. Besitzt ein Nutzer keinen Last.fm Account, können die Empfehlungen, die dann nur bedingt auf den persönlichen Geschmack abgestimmt sind, schnell ernüchternd wirken. Obwohl die meisten Daten schon im Voraus abgerufen werden können, wurde in Kap. 4.2 der "Live Crawler" vorgestellt, der die gewünschten Daten eines Last.fm Accounts abruft. Dieser benötigt eine gewisse Zeit, die je nach Internetverbindung schwanken kann und im Bereich mehrerer Minuten liegt. Diese Wartezeit tritt jedoch nur ein einzges mal auf und spielt während des Berechnens der Empfehlungen keine Rolle mehr.

6 Evaluation

Dieses Kapitel beschreibt die Ergebnisse und Schlussfolgerungen der Evaluation des Prototypen. Dabei wurden die, in Kapitel 3 vorgestellten, Metriken durch unterschiedliche Nutzer bewertet. Zuerst werden die Teilnehmer der Evaluation genauer betrachtet und anschließend die Ergebnisse analysiert.

6.1 Teilnehmeranalyse

Über die Webseite des Prototypen registrierten sich insgesamt 48 Teilnehmer für die Evaluation. Diese setzen sich aus 37 männlichen und 11 weiblichen Nutzern zusammen. Das Durchschnittsalter der Teilnehmer liegt bei 25 Jahren, wobei das kleinste Alter 21 Jahre beträgt und das größte 51 Jahre. Die genaue Altersverteilung aller Teilnehmer ist in Abb. 6.1 zu sehen.

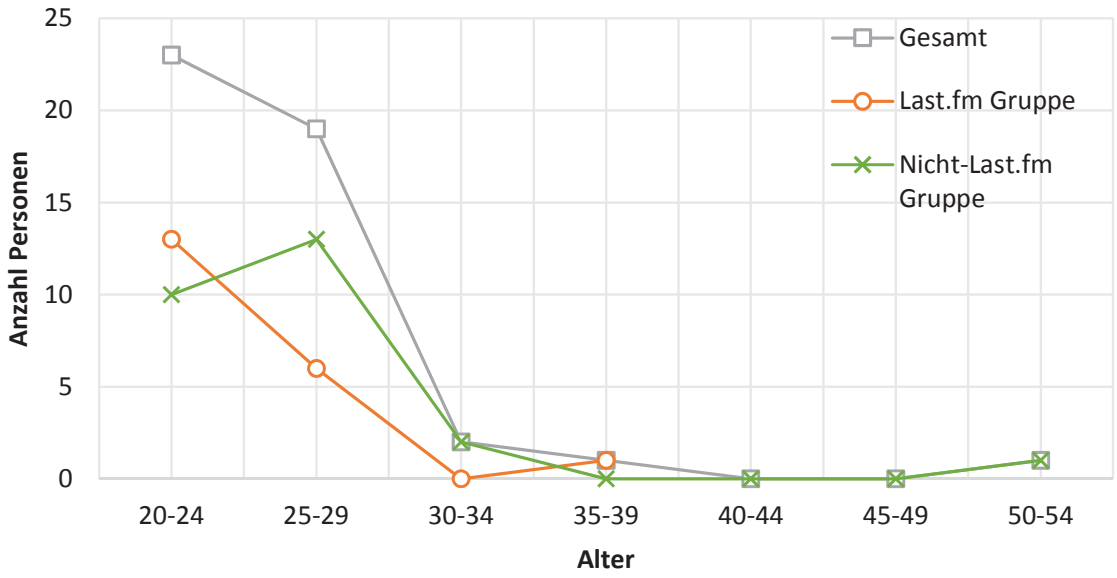


Abbildung 6.1: Altersverteilung der Teilnehmer

Jeder Teilnehmer konnte eine Menge an Sprachen angeben. Englisch wurde für jeden Nutzer vorausgesetzt, da auch die Textabschnitte des Prototyps in Englisch verfasst wurden. Weitere Sprachen, die angegeben wurden, waren Deutsch von 46 Teilnehmern, Französisch von zehn Teilnehmern, Japanisch von sieben Teilnehmern, Schwedisch oder Norwegisch von jeweils vier Teilnehmern und einige mehr.

Die im Benutzermodell als optionale Angaben festgelegten Informationen, wurden von 23 Teilnehmern zur Verfügung gestellt. Von diesen Teilnehmern gab jeder eine Lieblingsstadt an, bei denen jedoch nur geringfügige Überschneidungen festzustellen sind. Die beliebtesten Städte unter den Nutzern sind London, Sydney und Berlin, die von jeweils drei Teilnehmern angegeben wurden.

Die wichtigere optionale Information stellt der Last.fm Account eines Nutzers dar. Von den insgesamt 48 Teilnehmern stellten 21 einen Last.fm Account zur Verfügung. Durch die Angabe eines Last.fm Accounts werden zwei zusätzliche Metriken zur Empfehlungsgenerierung genutzt, die Nachbarschafts- und Tag-Metrik. Diese beziehen den aktuellen Musikgeschmack des Nutzers stärker in die Empfehlungen ein. Um die Unterschiede der Empfehlungen mit und ohne Last.fm Account und die damit verbundene Zufriedenstellung der unterschiedlichen Nutzer genauer untersuchen zu können, werden die Teilnehmer im Folgenden in zwei Gruppen eingeteilt. In Teilnehmer mit einem Last.fm Account und ohne einen Last.fm Account.

Der Großteil der Teilnehmer wurde über die Informatik-Mailingliste der Universität gewonnen, aber auch über soziale Netzwerke ließen sich einige Teilnehmer finden. Im nächsten Kapitel werden die Ergebnisse der Evaluation erläutert und mögliche Schlussfolgerungen über die Qualität der vorgeschlagenen Metriken aufgezeigt.

6.2 Ergebnisse

Wie bereits in Kap. 4.1.3 erklärt, konnten Teilnehmer eine Empfehlung mit “Love it”, “Like it” oder “Skip it” bewerten. Zusätzlich wurde für jede Empfehlung die Möglichkeit geboten, deren Serendipity Effekt mit *Ja* oder *Nein* zu bewerten.

Insgesamt wurden von den 48 Teilnehmern 1.567 Titel bewertet. Von diesen Empfehlungen wurden 34% (532) mit “Skip it” bewertet, 36% (560) der Titel wurden mit “Like it” bewertet und 30% (475) aller Titel wurden mit “Love it” bewertet. Daraus folgt, dass 66% aller Empfehlungen positiv bewertet wurden und die Nut-

zer zufriedengestellt haben. Abb. 6.2 zeigt die abgegebenen Bewertungen der unterschiedlichen Gruppen im Vergleich.

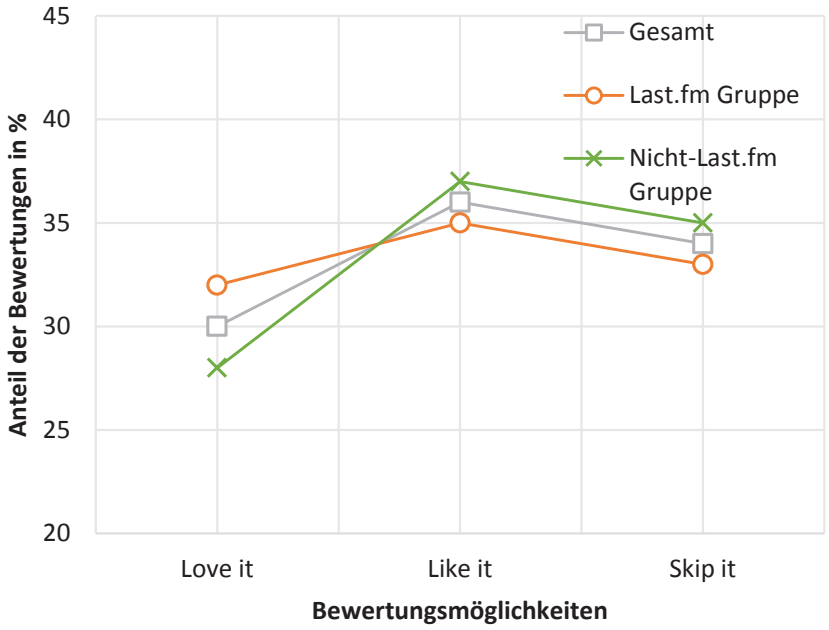


Abbildung 6.2: Bewertungen der unterschiedlichen Gruppen

Man erkennt, dass in beiden Gruppen die “Like it” Bewertungsmöglichkeit am häufigsten verwendet wurde. Durch die Last.fm Gruppe wurden 1.113 Titel bewertet, wobei jeder Nutzer durchschnittlich zwei Listen bewertete. 31% der Empfehlungen wurden mit “Love it” bewertet, 35% mit “Like it” und 34% mit “Skip it”. Obwohl die Nicht-Last.fm Gruppe mehr Teilnehmer beinhaltet wurden nur 454 Titel bewertet, wobei die meisten Teilnehmer nur eine Liste bewertet haben. Von den 454 Empfehlungen wurden ähnlich wie in der Last.fm Gruppe ca. 65% der Empfehlungen positiv bewertet.

Die Teilnehmer der Last.fm Gruppe bewerteten mehr Empfehlungen mit “Love it” und weniger mit “Skip it” als die Teilnehmer Nicht-Last.fm Gruppe. Daraus lässt sich schließen, dass in der Last.fm Gruppe ein höherer Grad an Zufriedenstellung und Motivation erreicht wurde. Das bestätigt auch die mehr als doppelte Zahl an Bewertungen, im Vergleich zur größeren Nicht-last.fm Gruppe.

Von den insgesamt 1.567 Titeln wurden 306 Empfehlungen als überraschend empfunden. Das sind ca. 20% aller Titel, deren Serendipity Bewertung mit Ja beantwortet wurden. Abb. 6.3 zeigt die Bewertungen der Empfehlungen, bei denen ein Serendipity Effekt bestätigt wurde.

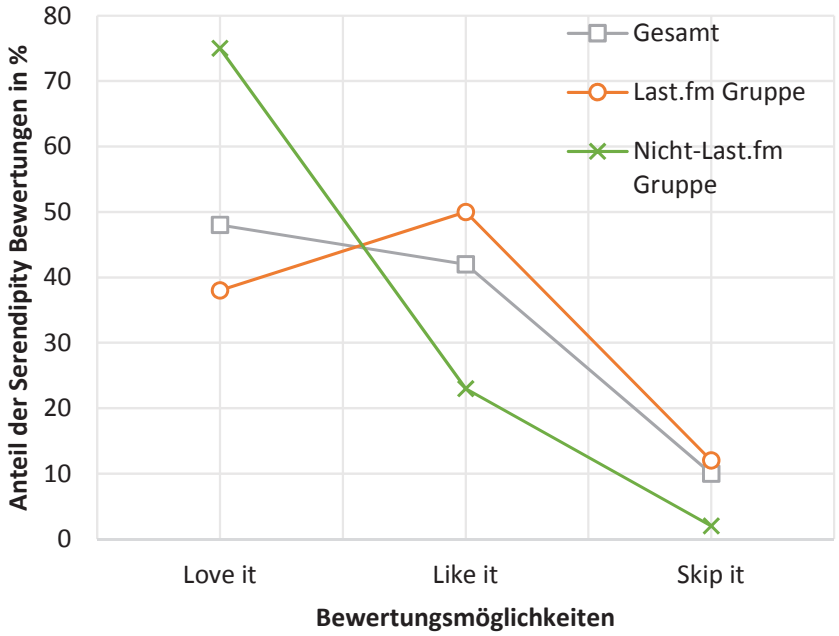


Abbildung 6.3: Bewertungen der Serendipity Empfehlungen

Man erkennt, dass der Großteil (90%) der Empfehlungen, bei denen ein Überraschungseffekt bestätigt wurde, durch die Teilnehmer positiv, mit "Love it" oder "Like it", bewertet wurde. Dies ist sowohl bei Teilnehmern der Last.fm Gruppe, als auch bei denen der Nicht-Last.fm Gruppe der Fall, wobei Last.fm Nutzer öfter "Like it" und Nicht-Last.fm Nutzer öfter "Love it" angaben. Das bestätigt die Annahme dieser Arbeit, dass ein Überraschungseffekt ein hohes Maß an Zufriedenstellung bei den Teilnehmern hervorruft.

Metriken

In diesem Abschnitt werden die Ergebnisse der unterschiedlichen Metriken detaillierter betrachtet. Abb. 6.4 zeigt die Verteilung der Bewertungen, durch die Teilnehmer der Last.fm Gruppe.

Die Abbildung zeigt, dass die Tag-Metrik mit Abstand die beliebteste Methode der Last.fm Gruppe ist. Es wurden knapp 50% aller, von der Tag Metrik generierten, Empfehlungen mit "Love it" bewertet und insgesamt 85% positiv bewertet. Danach folgt die Nachbarschafts-Metrik, die ungefähr gleich viele positive Bewertungen wie die Länder Charts Metrik erhalten hat, aber seltener negativ bewertet wurde. Die Bewertungen der Jahres Charts Methode wurden von den meisten Teilnehmern po-

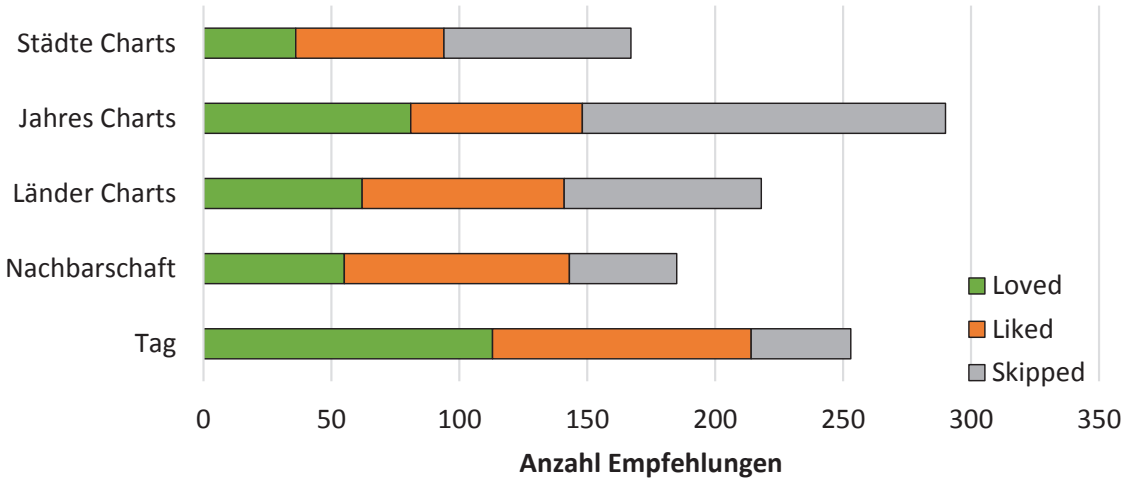


Abbildung 6.4: Bewertungsverteilung der Methoden - Last.fm Gruppe

sitiv aufgenommen, daher auch die große Zahl an Empfehlungen, die mit "Love it" bewertet wurden. Im Gegensatz dazu bewertete ein kleiner Teilnehmerkreis der Last.fm Gruppe jedoch fast alle Empfehlungen der Methode mit "Skip it". Daraus ergibt sich der große Anteil an negativen Bewertungen, welcher in Abb. 6.4 gut zu erkennen ist. Am schlechtesten schnitt die Städte Charts Methode ab.

In Abb. 6.4 erkennt man zusätzlich die Allokation der Listenplätze durch die dynamischen Anpassung. Theoretisch sollte die beliebteste Methode die meisten Plätze erhalten haben und dadurch die meisten Empfehlungen generiert haben. Die unbeliebteste wiederum die wenigsten Plätze und Empfehlungen. Diese Annahme wird zum größten Teil bestätigt, denn die Tag Metrik hat sehr viele Empfehlungen generiert, die Städte Charts Metrik am wenigsten. Das Ergebnis wird durch die Länder Charts Methode etwas verfälscht. Das lässt sich jedoch durch die Tatsache erklären, dass negative Bewertungen, sobald die Listenplätze einer Methode auf einen einzigen verringert wurden, keinen Einfluss mehr haben. Da sich die negativen Bewertungen auf einen kleinen Nutzerkreis beschränken wird dieser Effekt noch verstärkt.

Abb. 6.5 zeigt die Bewertungen der Nicht-Last.fm Gruppe und die Verteilung auf die unterschiedlichen Metriken. Für diese Gruppe konnten nur drei der fünf Metriken genutzt werden, da die Teilnehmer keinen Last.fm Account zur Verfügung gestellt haben. Die beliebtesten Methoden der Last.fm Gruppe fehlen daher.

Die Jahres Charts Metrik erhielt knapp 80% positive Bewertungen, wobei die Hälfte davon mit "Love it" bewertet wurden. Damit ist diese Methode mit deutlichem

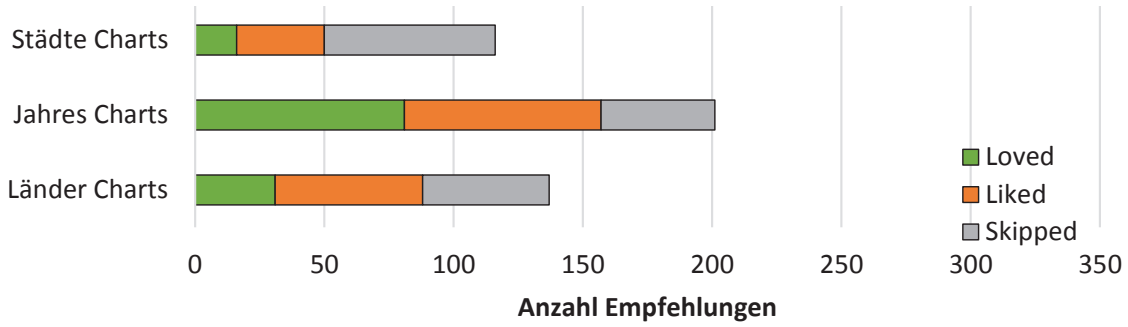


Abbildung 6.5: Bewertungsverteilung der Methoden - Nicht-Last.fm Gruppe

Abstand die beliebteste Methode der Nicht-Last.fm Gruppe. Danach folgt die Länder Charts Methode und, wie schon in der Last.fm Gruppe, die Städte Charts Metrik als letztes. Die dynamische Anpassung lässt sich in Abb. 6.5 klar erkennen, denn die Jahres Charts Methode erreichte als beliebteste Metrik auch die meisten Plätze, wobei die Städte Charts Methode am wenigsten erhielt.

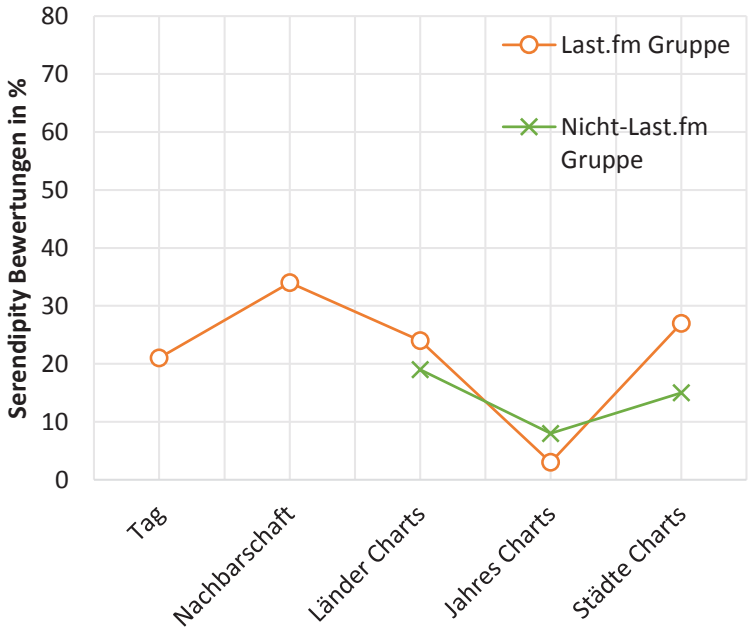


Abbildung 6.6: Serendipity Bewertungen der Metriken

Abb. 6.6 zeigt die Anteile an Serendipity Bewertungen der unterschiedlichen Metriken. In den gemeinsamen Methoden der zwei Gruppen zeigen sich ähnliche Ergebnisse. Am wenigsten Serendipity Bewertungen erhielt, in beiden Gruppen, die Jahres Charts Methode. Das war zu erwarten, da diese Metrik auf Charts bereits

vergangener Jahre zurückgreift und davon viele Lieder bereits bekannt sind. Sowohl die Länder Charts Methode als auch die Städte Charts Methode erhielten mehr Serendipity Bewertungen, wobei in der Last.fm Gruppe die Städte Charts Methode überwiegt und in der Nicht-Last.fm Gruppe die Länder Charts Methode. Die meisten Serendipity Bewertungen erhielt die Nachbarschafts Metrik.

Feedback

Nach einem Vorabtest durch einige ausgesuchte Nutzer wurde hilfreiches Feedback gesendet, welches zu folgenden Verbesserungen führte:

- Genauere Erläuterung der gewünschten Informationen während der Registrierung
- Historienseite der bereits bewerteten Empfehlungen
- Anzeige globaler Top-Titel
- Fehlerbehebungen

Während der eigentlichen Evaluationsphase wurde kaum Feedback gesendet. Die fehlerfreie Funktion und klare Struktur der Seite wurde gelobt. Zusätzlich wurde eine neutrale Bewertungsmöglichkeit der Art “In Ordnung” gewünscht.

6.3 Zusammenfassung

Insgesamt wurden die Empfehlungen des Prototyps gut aufgenommen, was knapp 70% positive Bewertungen belegen. Serendipity Effekte wurden bei ca. 20% aller Empfehlungen bestätigt, wobei der Großteil der Serendipity Empfehlungen positiv aufgenommen wurde.

Alles in allem schnitten die Last.fm Account spezifischen Methoden am besten ab. Sowohl in Bewertung, als auch Serendipity erhielten diese Methoden die meisten positiven und am wenigsten negative Wertungen. Das zeichnet sich auch in der Anzahl der Bewertungen ab, denn durch die Last.fm Gruppe wurden weitaus mehr Titel bewertet, als durch die Nicht-Last.fm Gruppe. Das deutet möglicherweise darauf hin, dass sich die Nutzer mehr Spezialisierung auf deren Geschmack wünschten als die Charts geboten haben und dadurch die Motivation verloren.

Bei den gemeinsamen Methoden verfolgten die Gruppen einen ähnlichen Trend. Die Jahres Charts Methode schnitt bei den Bewertungen am besten ab, erzeugte jedoch die wenigsten Serendipity Effekte. Länder Charts und Städte Charts erzeugten ähnlich viele Serendipity Effekte, jedoch war die Städte Charts Methode in beiden Gruppen die unbeliebteste. Möglicherweise waren die sehr regionalen Musiktitel, welche diese Methode empfehlen sollte, zu außergewöhnlich und trafen den Musikgeschmack der Teilnehmer nur in seltenen Fällen.

Über die Effekte der dynamischen Anpassung ließen sich leider keine wirklichen Aussagen treffen, da die meisten Teilnehmer nicht bis zu einer dritten Liste bewertet haben. Es hätte klarer herausgestellt werden müssen, dass mindestens fünf Listen bewertet werden sollten, um eine fundierte Aussage über die Anpassung treffen zu können. Zusätzlich sollte den Teilnehmern der Serendipity Effekt und dessen Unterschied zum “Novelty” Aspekt eindeutig erläutert werden, um so eine Vermischung während der Evaluation zu verhindern.

7 Fazit

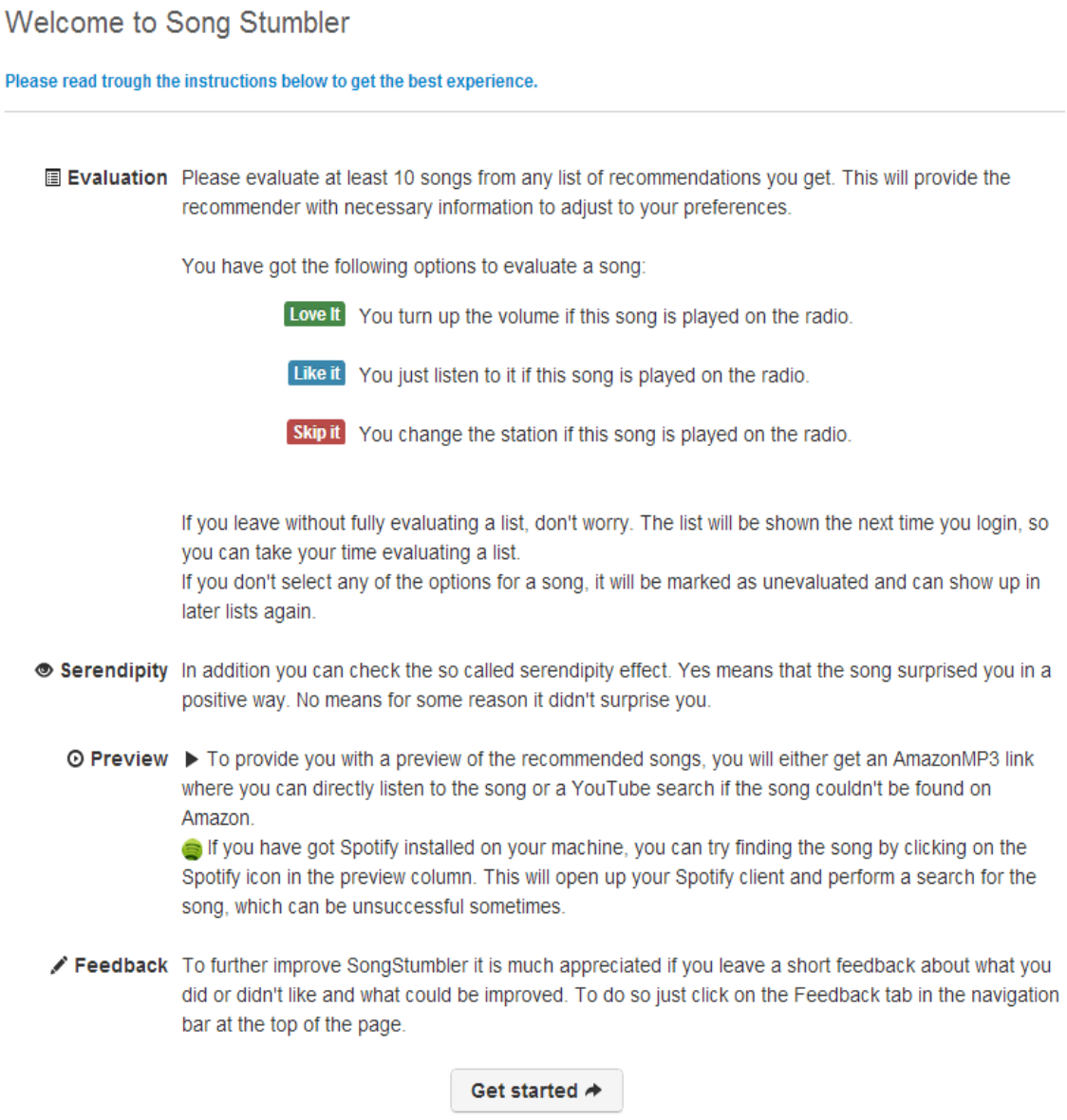
Durch die immer größer werdende Zahl an Streamingdiensten und digitalen Musikbibliotheken wird die Masse an verfügbaren Musiktiteln immer unübersichtlicher. Zusätzlich spezialisieren sich viele der eingesetzten Recommender Systeme zu sehr auf die vom Nutzer bekannten Präferenzen und verhindern dadurch überraschende und neuartige Entdeckungen.

Diese Arbeit stellte das Recommender System *Song Stumbler* vor, welches durch fokussieren des Serendipity Aspektes, überraschende Empfehlungen generieren sollte. Um diesen Zweck zu erfüllen wurden fünf Metriken vorgestellt. Drei dieser Metriken nutzen Jahres, Länder und Städte Charts, um die allgemeine Beliebtheit und die Diversität der verschiedenen Charts auszunutzen. Die übrigen zwei Methoden nutzen die Informationen eines zur Verfügung gestellten Last.fm Accounts, um die Präferenzen des Nutzers einzubeziehen aber auch zu erweitern. Es wurde gezeigt, dass einige klassische Probleme wie "Cold Start" oder "Überspezialisierung", bei Nutzung dieser Metriken nicht auftreten. Um die Bewertungen eines Nutzers einzubeziehen wurde eine dynamische Anpassung der Empfehlungslisten eingeführt. Dadurch werden die Plätze der Metriken in den Empfehlungslisten bei jeder Generierung, entsprechend der Bewertungen, neu verteilt.

Um die Annahmen und Ergebnisse der unterschiedlichen Metriken zu testen wurde eine Evaluation mithilfe des Prototyps durchgeführt. Während der Evaluation wurden 1.567 Empfehlungen bewertet. Es zeigte sich, dass die Empfehlungen mit knapp 70% positiver Resonanz von den Teilnehmer gut aufgenommen wurden. Die Metriken, welche auf die Informationen eines Last.fm Accounts zugreifen, stellten sich als die Beliebtesten unter den Evaluationsteilnehmern heraus und erzeugten zusätzlich mit das höchste Maß an Serendipität. Insgesamt wurden bei 20% aller Empfehlungen ein Überraschungseffekt bestätigt, welcher in fast allen Fällen eine positive Bewertung hervorrief. Das bestätigt den positiven Einfluss des Serendipity Aspekts auf die Zufriedenstellung einer Empfehlung.

Aus der Evaluation lässt sich schließen, dass das Auswählen der Musiktitel aus unterschiedlichen Charts die Präferenzen des Nutzers möglicherweise zu wenig einbezieht. Zukünftige Arbeiten könnten darin bestehen die Auswahl der Charts während des Betriebes auf die Präferenzen des Nutzers abzustimmen und nicht im Voraus festzulegen. Es könnte beispielsweise durch eine Art Rotationsmechanismus Charts unterschiedlicher Städte, Länder und Jahre ausprobiert und erst danach, anhand der Bewertungen, auf eine kleinere Menge eingegrenzt werden.

A Einführung



Literaturverzeichnis

[1] B. S. Michael Baumann, Mathias Berek, "Klangraum Internet," in *Report des Forschungsprojektes Medienkonvergenz Monitoring*, 2012.

[2] Paul Resnick and Hal R. Varian, "Recommender Systems," *Commun. ACM*, 1997.

[3] J. B. Schafer, J. Konstan, and J. Riedi, "Recommender Systems in E-Commerce," in *EC '99: Proceedings of the 1st ACM conference on Electronic commerce*. New York, NY, USA: ACM, 1999.

[4] E. Pariser, *The Filter Bubble: What the Internet Is Hiding from You*. Penguin Books, Limited, 2012. [Online]. Available: http://books.google.de/books?id=Qn2ZnjzCE3gC

[5] C.-N. Ziegler, S. M. McNee, J. A. Konstan, and G. Lausen, "Improving Recommendation Lists Through Topic Diversification," in *Proceedings of the 14th international conference on World Wide Web*, ser. WWW '05. New York, NY, USA: ACM, 2005.

[6] Y. C. Zhang, D. Ó Séaghdha, D. Quercia, and T. Jambor, "Auralist: Introducing Serendipity into Music Recommendation," in *Proceedings of the 5th ACM Conference on Web Search and Data Mining (WSDM-12)*, Seattle, WA, 2012.

[7] S. Vargas and P. Castells, "Rank and Relevance in Novelty and Diversity Metrics for Recommender Systems," in *Proceedings of the fifth ACM conference on Recommender systems*, ser. RecSys '11. New York, NY, USA: ACM, 2011.

[8] J. L. Herlocker, J. A. Konstan, L. G. Terveen, John, and T. Riedl, "Evaluating Collaborative Filtering Recommender Systems," *ACM Transactions on Information Systems*, 2004.

Erklärung

Hiermit erkläre ich, dass ich diese Abschlussarbeit selbständig verfasst habe, keine anderen als die angegebenen Quellen/Hilfsmittel verwendet habe und alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten Schriften entnommen wurden, als solche kenntlich gemacht habe. Darüber hinaus erkläre ich, dass diese Abschlussarbeit nicht, auch nicht auszugsweise, bereits für eine andere Prüfung angefertigt wurde.

Ort, Datum	Unterschrift