Masterarbeit

TRecs

Entwicklung eines Music Recommender auf Basis von Last.fm

Simon Franz

Matrikelnummer: 2111308

04.09.2012

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

Bearbeitungszeitraum

07. 03. 2012 – 07. 09. 2012

Gutachter

Prof. Dr. Georg Lausen

Betreuer

Dr.-Ing. Thomas Hornung

Kurzfassung

Gegenstand dieser Arbeit ist die Entwicklung eines hybriden Recommenders, der Empfehlungen für Musiktitel in Echtzeit berechnen kann. Das implementierte Track Recommender System (TRecs) basiert dabei auf drei kombinierten Recommender-Metriken. Die Ähnlichkeit von Titeln wird hierbei basierend auf gemeinsamem Hörverhalten verschiedener Benutzer der Musikplattform Last.fm, ähnlichen Tags sowie dem zeitlichen Abspielverhalten der Titel bestimmt. Die Gewichtung der einzelnen Metriken ist konfigurierbar und das System liefert eine detaillierte Aufschlüsselung der jeweiligen Beiträge für das Gesamtergebnis einer Empfehlung.

Zur Akquise der Datenbasis wurde ein Crawler implementiert, der 50 Millionen Hörereignisse zu mehr als 4,5 Millionen Musiktiteln von Last.fm sammelte. Da die für die Empfehlungen notwendigen Berechnungen quadratisch in der Anzahl der Eingabetitel sind, wurde zum einen die Berechnung der Empfehlungen in eine Vorberechnungsphase und eine Online-Phase unterteilt. Zum anderen musste die Anzahl der Titel reduziert werden, um Empfehlungen in Echtzeit auf einem Einzelplatzsystem berechnen zu können. Bei der Reduktion der Titelanzahl wurden die relevantesten Charakteristika des zugrunde liegenden Datensatzes im Wesentlichen beibehalten. Die Evaluation wurde somit auf einer Menge von 15.000 Titeln durchgeführt.

Für die Evaluation von TRecs wurde eine Web-Oberfläche entwickelt, mit der Testbenutzer sich Titel vorschlagen und anschließend die Empfehlungen bewerten konnten. Da ein Musik-Recommender personalisierte Empfehlungen bereitstellen sollte, spielen diese Bewertungen eine zentrale Rolle bei der Berechnung von weiteren Empfehlungen. Je mehr Iterationen durchlaufen werden, das heißt Musiktitel bewertet werden, umso präziser sollten auch die Empfehlungen werden. Dies wurde mit einer Evaluation, an der sich über 140 Personen beteiligten, näher untersucht.

Die Evaluation von TRecs zeigte, dass sich die Qualität der Empfehlungen mit steigender Bewertungsanzahl nur leicht verbesserte. Ein möglicher Grund hierfür könnte die beschränkte Anzahl der in der Evaluation verfügbaren Titel sein, so dass in weiteren Arbeiten Recommender-Metriken auf einem verteilten System mit der ganzen Datenbasis untersucht werden sollten.

Schlagwörter: Recommender, Last.fm, Musik, Recommender, TRecs, Real-Time

Inhaltsverzeichnis

1. Einleitung

Die Menge an verfügbaren Daten im World Wide Web steigt rasant an. So sprach Eric Schmidt, ehemaliger Chef von Google, schon 2010 von einem Datenvolumen von fünf Exabyte, welches alle zwei Tage im Internet generiert wird[1]. Diese Entwicklung macht es Benutzern immer schwieriger relevante Informationen zu finden. Um das Finden von relevanten Inhalten zu erleichtern, können Recommender Systeme eine vielversprechende Ergänzung zu Suchmaschinen darstellen. Recommender Systemen ist es möglich, aus einer Menge an Informationen einem Benutzer die Informationen anzubieten, für welche er sich interessiert.

Insbesondere bei Onlineshops oder sozialen Netzwerken gewinnen diese Systeme immer mehr an Bedeutung. Bei sozialen Netzwerken bieten sie sowohl den Betreibern als auch den Nutzern zahlreiche Vorteile. Beispielsweise können den Benutzern automatisch Eigenschaften angeboten werden, welche das Profil des Benutzers vervollständigen. Für den Betreiber bietet sich der Vorteil, dass je detaillierter ein Profil ist, desto personalisierter kann die angezeigte Werbung sein. Mit Hilfe von Recommender Systemen ist es auch möglich, Produkte anhand der angegebenen Interessen vorzuschlagen. So ist denkbar, dass ein Nutzer, der an sehr vielen unterschiedlichen Musikgruppen Interesse hat, diese auch unterwegs gerne hört und sich daher für einen MP3-Player interessieren könnte. Das Empfehlen von Produkten ist insbesondere bei Online-Shops wichtig. So können anhand der Kaufhistorie eines Benutzers Empfehlungen für weitere Produkte abgeleitet werden. Umso mehr Dienste ein Online-Shop seinen Kunden anbietet, wie zum Beispiel ein Feedback für andere Benutzer zu schreiben, umso mehr Informationen kann auch der Online-Shop zusammenfassen und diese nutzen, um dem Benutzer weitere Empfehlungen vorzuschlagen. Auf diese Weise wird versucht, mit Recommender Systemen die Rolle eines persönlichen Verkäufers in einem Online-Shop zu simulieren.

[1] http://techcrunch.com/2010/08/04/schmidt-data/ - Zugriff: 01.09.2012

Ein Beispiel für die Nutzung von Recommender Systemen bei Online-Shops findet sich bei Amazon[2]. Amazon bietet jedem Benutzer eine Auswahl an Produkten, für welche er sich interessieren könnte. Diese Produkte werden anhand der Kaufhistorie und anhand der auf der Internetseite betrachteten und gekauften Produkte vorgeschlagen [1]. Auch in einer Reihe anderer Online-Shops werden Recommender Systeme verwendet. Mit *iTunes Genius* wurde im Jahre 2008 ein Recommender System von Apple entwickelt, um die Verkäufe auf der eigenen Plattform zu steigern. Bei iTunes handelt es sich nicht um einen Online-Shop, bei dem ein Benutzer auf einer Internetseite die Produkte ansehen kann, sondern um ein Programm, das ein Käufer zuerst herunterladen muss, um auf den Musik-Katalog von iTunes zugreifen zu können. iTunes dient hierbei nicht nur zum Kauf von Musik, sondern bietet auch die Funktion, als Medienplayer genutzt zu werden. So kann iTunes alle digitalen Musikdateien verwalten und analysieren. Das Hörverhalten eines angemeldeten Benutzers kann dadurch gespeichert werden und dem Benutzer können auf der Grundlage der gesammelten Informationen passende Kaufempfehlungen angeboten werden. Netflix, eine Online-DVD-Videothek, stellte im Jahr 2006 eine Siegesprämie von einer Million US-Dollar in Aussicht für denjenigen, der, ausgehend von dem bereits existierenden Recommender System, die Empfehlungen ihrer Plattform um 10% verbessert. Erst drei Jahre später konnte die Aufgabe vom Team *BellKor's Pragmatic Chaos* gelöst werden. Dem Team gelang eine Verbesserung von 10,05%. Dies konnte jedoch nur durch die Kombination von über 80 unterschiedlichen Recommendern erreicht werden [2, 3].

In dieser Arbeit wird untersucht, wie Recommender verwendet werden können, um Benutzern Musiktitel vorzuschlagen, die ihnen gefallen könnten. Üblicherweise wird die Rolle durch das Radio übernommen, bei dem ein Team von Experten neue Musik für einen typischerweise eher breiten Musikgeschmack aussucht. Insbesondere bei Musik ist es jedoch unmöglich, über das Medium (öffentliches) Radio die Vorlieben einer einzelnen Person abzudecken. Hier versuchen sogenannte personalisierte Online-Radios für jeden Benutzer Musiktitel zu spielen, welche den Vorlieben dieses Benutzers entsprechen. Diesen Ansatz verfolgt auch die Musikplattform Last.fm[3], welche Hörereignisse von Benutzern speichert, um ein personalisiertes Radio zu spielen. Dieses Ziel, Benutzern personalisierte Empfehlungen zu geben, wird auch in dieser Arbeit verfolgt. Als Basis dient hierbei das von Last.fm verfügbare Hörver-

[2] http://www.amazon.com/ - Zugriff: 01.09.2012
[3] http://www.lastfm.de/ - Zugriff: 01.09.2012

halten von Online-Radiohörern. Mit der Implementierung des Track Recommender Systems (TRecs) werden dem Benutzer unter Anderem anhand seiner Hörhistorie neue Titelempfehlungen vorgeschlagen. Dabei kommt eine Kombination verschiedener Recommender-Algorithmen zum Einsatz, um die Qualität der Empfehlungen zu maximieren.

1.1. Struktur

In Kapitel 2 werden die relevanten Schnittstellen von Last.fm vorgestellt, mit deren Hilfe unter anderem Hörereignisse von Benutzern geladen werden können. Anschließend werden in Kapitel 3 die Grundlagen von Recommender Systemen vorgestellt. Hier wird insbesondere der Unterschied zwischen inhaltsbezogenen (Content-based) Recommendern, gemeinschafts-basierten (Collaborative) Recommendern und hybriden Recommendern beschrieben. In Kapitel 4 wird anschließend ausgeführt, mit welcher Methode die Datenbasis des hier verwendeten hybriden Recommenders erstellt wurde. Diese Datenbasis wird daraufhin in Kapitel 5 diskutiert. Der Aufbau von TRecs und die Methode wie Empfehlungen für Benutzer berechnet wurden, werden in Kapitel 6 beschrieben. Um den implementierten Prototypen zu testen, wurde eine Evaluation durchgeführt, welche in Kapitel 7 vorgestellt wird. Nachfolgend wird in Kapitel 8 ein Ausblick gegeben, an welchen Bereichen bei zukünftigen Arbeiten weitergearbeitet werden kann. Zuletzt wird die Arbeit in Kapitel 9 zusammengefasst.

Je nach Interessen des Lesers werden vier unterschiedliche Lesepfade vorgestellt, welche auch in Abb. 1.1 veranschaulicht sind:

1. Leser, welche keine Kenntnisse in den Bereichen Recommender Systeme und Datengenerierung haben, sollten dem normalen Lesepfad folgen, welcher in Abb. 1.1 schwarz gekennzeichnet ist und mit Kapitel 2 beginnt.
2. Für Leser mit Vorwissen im Bereich Recommender Systeme empfiehlt es sich, bei Kapitel 2 zu beginnen und dem blau markierten Lesepfad zu folgen. Das folgende Kapitel 3, welches sich mit den Grundlagen von Recommender Systemen beschäftigt, kann hier ausgelassen werden und es kann direkt mit der Datenakquise in Kapitel 4 fortgefahren werden. Anschließend kann dem normalen Lesepfad gefolgt werden.
3. Leser, welche hauptsächlich an der Datenakquise und dem Ergebnis eines hybriden Recommender intererssiert sind, sollten dem rot markierten Lesepfad

folgen. Dieser beginnt ebenso mit Kapitel 2, welches die genutzten Schnittstellen von Last.fm erklärt. Anschließend wird beschrieben, wie der Crawler aufgebaut war und welche Charakteristiken in dem geladenen Datensatz erkennbar waren. Anschließend wird das Ergebnis der Evaluation vorgestellt, welche mit den beschriebenem Datensatz erreicht wurden.

4. Unter der Voraussetzung, dass der Leser sich nur für Recommender Systeme im Allgemeinen interessieren sollte, sollte er dem türkis markierten Lesepfad folgen. Dieser beginnt mit Kapitel 3, welches die Grundlagen von Recommender Systemen erklärt. Anschließend wird der Lesepfad mit dem in dieser Arbeit implementierten TRecs fortgeführt. Zuletzt wird die Evaluation von TRecs vorgestellt und im Ausblick Verbesserungen von TRecs und der Evaluation diskutiert.

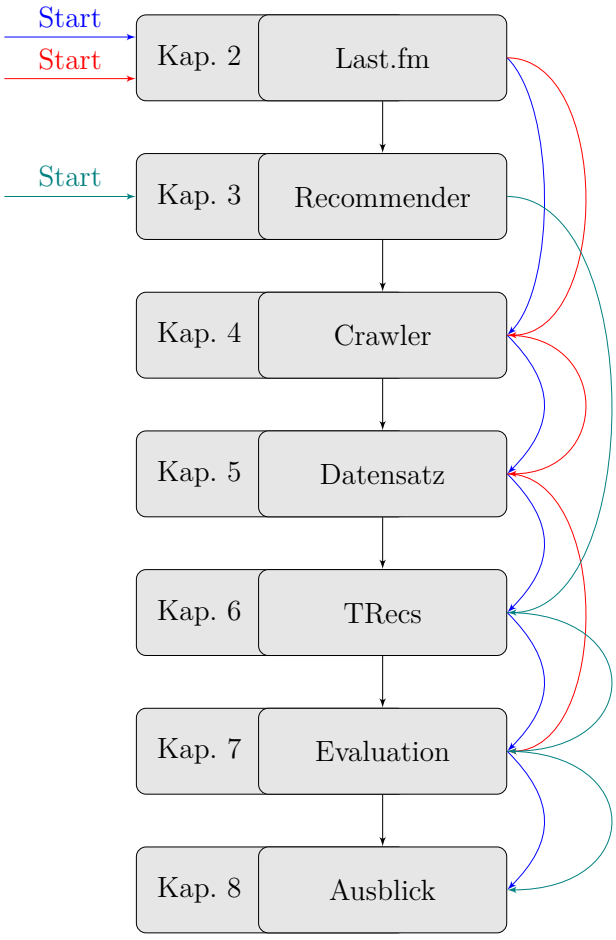


Abbildung 1.1.: Gliederung der Arbeit

2. Last.fm

Last.fm ist ein Online-Radio, welches die Hörhistorie der Benutzer speichert. Die Idee dahinter ist, ein personalisiertes Radio anbieten zu können. Im Laufe der Zeit haben sich die Leistungen von Last.fm jedoch erweitert. So bietet es heute auch die Möglichkeit, durch Musik neue Personen kennen zu lernen und Freundschaften zu schließen (Abb. 2.1). Mit Hilfe dieser Freunde kann ein Benutzer neue Musik kennenlernen, indem er zum Beispiel ein so genanntes Nachbarschaftsradio auswählt. Durch dieses werden für den Benutzer unbekannte Titel gespielt, welche seine Freunde bereits gehört haben.

Um das Hörverhalten zu sammeln, bietet Last.fm eine Vielzahl von Möglichkeiten. Diese werden als *Scrobbeln* bezeichnet. Zum einen kann sich ein Benutzer ein kostenloses Programm herunterladen, welches unterschiedliche Medienplayer auf dem Computer überwacht. Zum anderen gibt es Anwendungen, die sich ein Benutzer auf sein Smartphone herunterladen kann. Zuletzt bietet Last.fm eine Programmierschnittstelle (API) an, welche von Entwicklern anderer Plattformen genutzt werden kann, um gehörte Titel an Last.fm zu senden. Sobald ein Musiktitel gehört wird, schickt das überwachende Programm eine Nachricht an Last.fm. Auf der Plattform ist so in Echtzeit zu sehen, welchen Titel ein Benutzer gerade hört. Sobald der Titel fertig gespielt wurde, wird ebenfalls eine Nachricht an Last.fm gesendet, wodurch der Titel als Hörereignis des Benutzers gespeichert wird.

Des Weiteren ist es jedem Benutzer möglich, über verschiedene Wege, wie beispielsweise über die Webseite von Last.fm, Titel individuell zu markieren. Es besteht somit die Möglichkeit, dass Benutzer Titel in eine Liste von favorisierten Titeln oder in eine Liste gebannter Titel aufnehmen. Diese werden für das personalisierte Radio besonders beachtet, damit zum Beispiel kein Titel gespielt wird, welchen der Benutzer bereits gebannt hat. Darüber hinaus ermöglicht Last.fm, Textmarkierungen (Tags) zu einem Titel hinzuzufügen. Die Nutzung dieser Tags ist freiwillig und dient zur Kategorisierung von unterschiedlichen Titeln.

Abbildung 2.1.: Screenshot der Last.fm Plattform

Ein Problem, das sich aus dieser Herangehensweise ergibt, sind doppelte Einträge, sogenannte Dubletten. Da jeder Benutzer eigene Musikdateien hört, ist es möglich, dass derselbe Titel von einem Benutzer anders benannt wird als von einem anderen Benutzer. Unterschiedlich benannte Titel werden von Last.fm als unterschiedliche Titel gespeichert, sodass es viele unterschiedliche Schreibweisen desselben Titels geben kann.

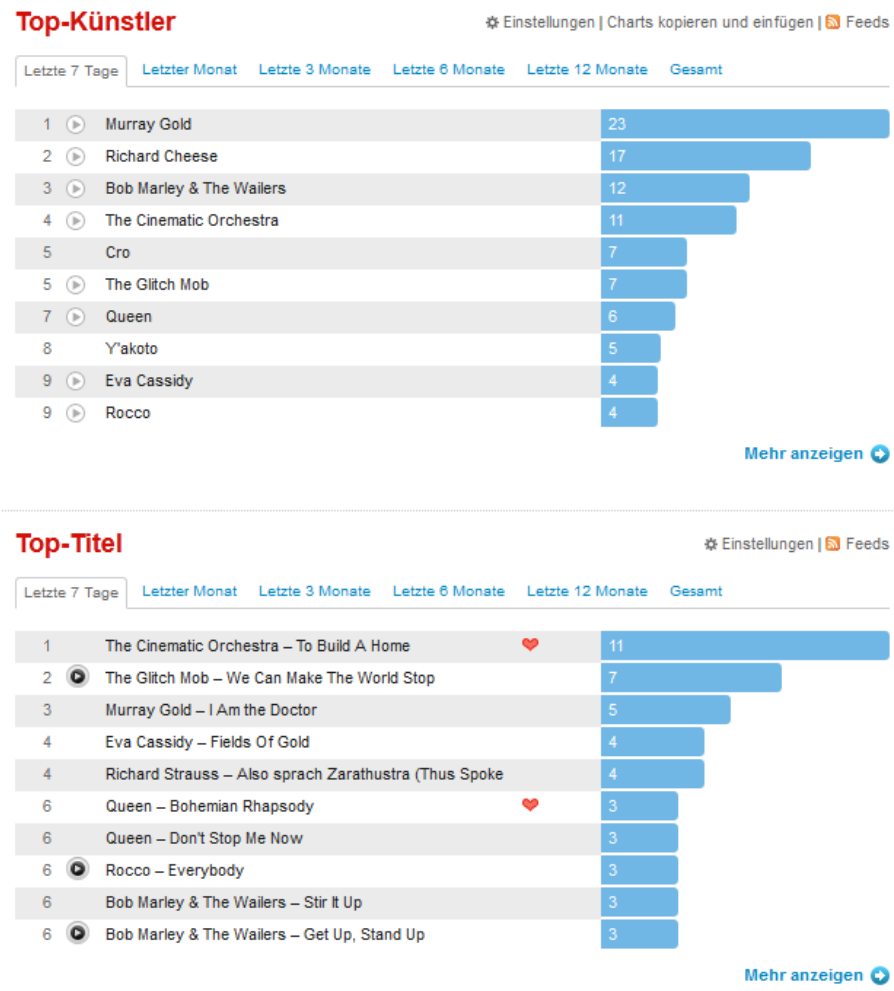


Abbildung 2.2.: Screenshot der Last.fm Charts

Aufgrund der Hörereignisse ist es Last.fm möglich, nicht nur ein personalisiertes Radio zu erstellen, sondern dem Benutzer noch viele andere Auswertungen zur Verfügung zu stellen. Jeder Benutzer kann seine eigenen Charts einsehen, die aus einer Liste mit den meist gehörten Titeln besteht (Abb. 2.2). Dabei hat der Benutzer die Möglichkeit den Zeitraum zu bestimmen, welcher zur Chart-Erstellung berücksichtigt werden soll.

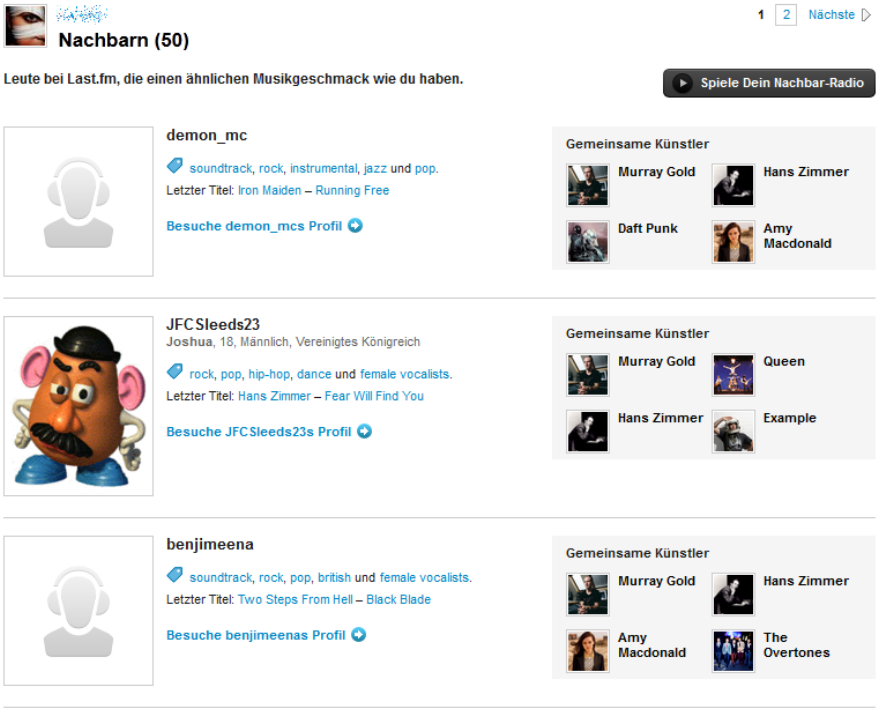


Abbildung 2.3.: Screenshot der Last.fm Nachbarschaft eines Benutzers

Das Finden neuer Freunde wird bei Last.fm durch die *musikalischen Nachbarn* unterstützt (Abb. 2.3). Es werden die Hörereignisse unterschiedlicher Benutzer verglichen und ein Prozentwert der musikalischen Übereinstimmung errechnet. Last.fm benutzt diesen Prozentwert, um bisher unbekannte Benutzer mit ähnlichem Musikgeschmack vorzuschlagen. Dabei wird nicht darauf geachtet, ob sich ein musikalischer Nachbar in derselben Region aufhält oder nicht.

Das betrachtete Schema der von Last.fm erfassten Hörereignisse ist in vereinfachter Form in Abb. 2.4 dargestellt. Ein *Titel* hat einen Titelnamen und einen definierten *Interpreten*. Zu jedem *Titel* können beliebig viele *Tags* gespeichert werden, zu welchen jeweils ein Wert gespeichert wird, der die Häufigkeit des Vorkommens angibt. Jeder *Benutzer* kann sich einen Benutzernamen auswählen, welcher noch nicht von anderen Benutzern belegt sein darf. Zu jedem Hörereignis eines Benutzers wird der genaue Zeitpunkt gespeichert. Ebenso wird bei der Liste der favorisierten Titel und der gebannten Titel der Zeitpunkt gespeichert, zu welchem die Aktion durchgeführt wurde.

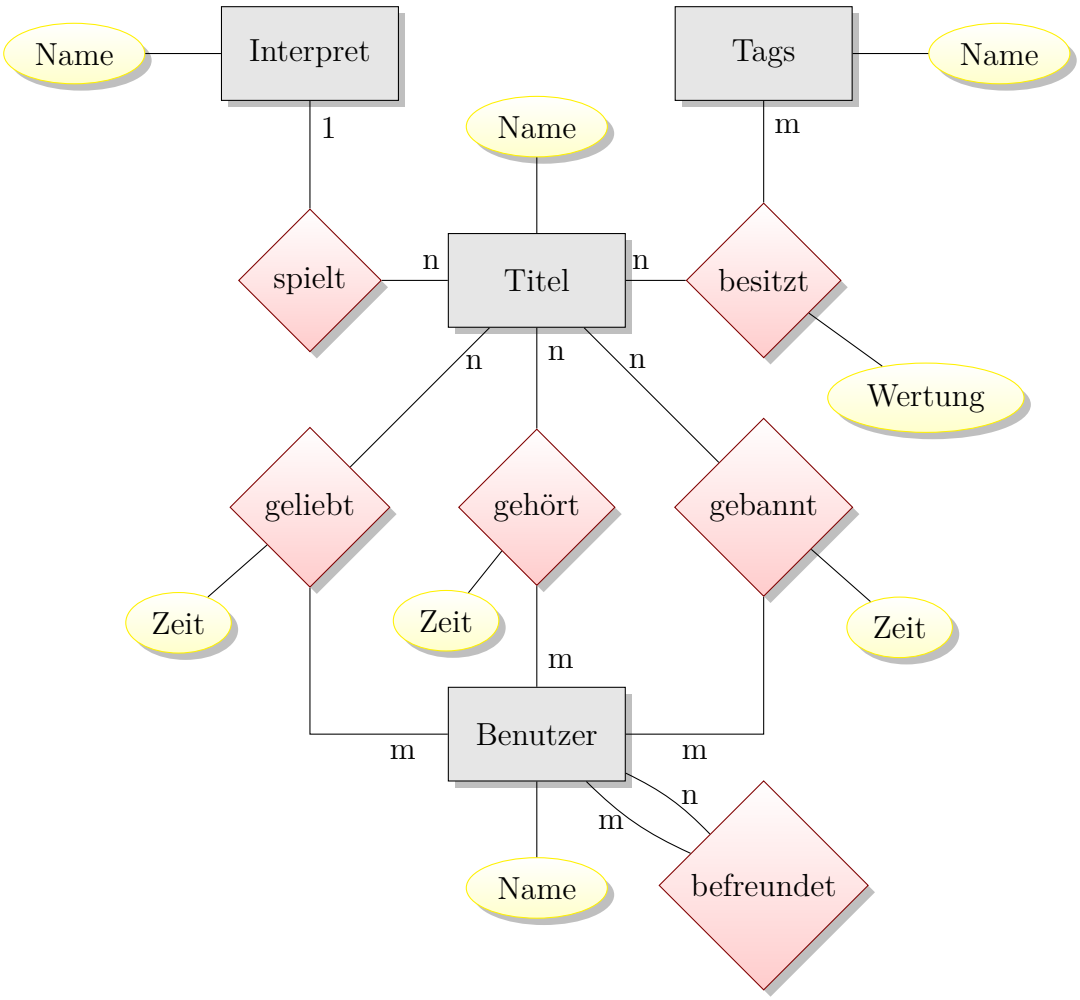


Abbildung 2.4.: Vereinfachtes Schema für Last.fm Hörereignisse

2.1. Last.fm API

Ein wesentlicher Grund dafür, Last.fm als Basis dieser Arbeit zu wählen, ist die vielseitige API, die die Plattform anbietet. Mit Hilfe dieser ist es möglich, alle Informationen über Titel, Tags, Benutzer und Interpreten abzufragen. Zur Nutzung der API muss sich ein Benutzer registrieren. Anschließend erhält er einen *api_key*, der bei Anfragen mitgesendet werden muss. Die API kann für nicht kommerzielle Zwecke kostenlos genutzt werden. Im Folgenden werden einige der in dieser Arbeit genutzten Schnittstellen vorgestellt.

Methode user.getFriends

Um die Freunde eines Benutzers zu identifizieren, wird die Funktion `user.getFriends` bereitgestellt. Durch die Angabe eines Benutzernamens liefert Last.fm eine vom Benutzer wählbare Anzahl an Freunden zurück. Die Schnittstelle liefert eine seitenweise Anzeige, sodass ebenfalls spezifiziert werden muss, wie viele Freunde auf einer Seite angezeigt werden sollen und welche Seite angezeigt werden soll. Das Maximum der von Last.fm angezeigten Freunde auf einer Seite beträgt 200 Benutzer.

Die Methode `user.getFriends` bietet folgende Möglichkeiten einen Aufruf zu erstellen:

- `user` (benötigt) : Der Last.fm Benutzername, dessen Freunde angezeigt werden sollen.
- `recenttracks` (optional) : Auswahl, ob auch Informationen über Titel angezeigt werden, die kürzlich von einem Freund gehört wurden.
- `limit` (optional) : Anzahl der Ergebnisse pro Seite; standardmäßig werden 50 Elemente angezeigt.
- `page` (optional) : Seitennummer der angeforderten Seite; standardmäßig wird die erste Seite angezeigt.
- `api_key` (benötigt) : Ein Last.fm API Schlüssel.

In Listing 2.1 ist eine Beispielantwort der `user.getFriends`-Schnittstelle zu sehen. Zu jedem Benutzer wird hier die Anzahl der gespielten Titel (`<playcount>`), der Registrierungszeitpunkt (`<registered>`) und die URL des Profilbildes in unterschiedlichen Auflösungen (`<image>`) angegeben. Auch werden Wohnort, Geschlecht und Alter angezeigt. Da diese Informationen jedoch vom Benutzer selbst eingegeben werden, kann der Wahrheitsgehalt dieser Informationen nicht überprüft werden.

```
<lfm status="ok" total="109" page="1" perPage="50" totalPages="3">
   <friends user="joanofarctan">
      <user>
         <name>eartle</name>
         <realname>Michael Coffey</realname>
         <image size="small">http://userserve-ak.last.fm/serve ↙
            /34/45718509.jpg</image>
         <image size="medium">http://userserve-ak.last.fm/serve ↙
            /64/45718509.jpg</image>
         <image size="large">http://userserve-ak.last.fm/serve ↙
            /126/45718509.jpg</image>
         <image size="extralarge">http://userserve-ak.last.fm/serve ↙
            /252/45718509.jpg</image>
         <url>http://www.last.fm/user/eartle</url>
         <id>7737850</id>
         <country>UK</country>
         <age>29</age>
         <gender>m</gender>
         <subscriber>1</subscriber>
         <playcount>45366</playcount>
         <playlists>4</playlists>
         <bootstrap>0</bootstrap>
         <registered unixtime="1189696970">2007-09-13 15:22 ↙
            </registered>
         <type>subscriber</type>
      </user>
      ...
   </friends>
</lfm>
```

Listing 2.1: Beispielantwort der `user.getFriends`-Schnittstelle

Methode user.getRecentTracks

Die kürzlich gehörten Titel eines Benutzers können durch die Funktion `user.getRecentTracks` abgerufen werden, wozu ein Benutzername benötigt wird. Um die seitenweise Anzeige der kürzlich gehörten Titel zu steuern, muss zusätzlich die Anzahl der anzuzeigenden Titel spezifiziert werden und die gewünschte Seite angegeben werden. Des Weiteren kann ein Zeitrahmen definiert werden, sodass nur Titel angezeigt werden, welche zwischen zwei Zeitpunkten gehört wurden.

Um die Methode `user.getRecentTracks` aufzurufen bietet Last.fm folgende Parameter an:

- `user` (benötigt) : Der Last.fm Benutzername, dessen letzte gehörte Titel angezeigt werden sollen.
- `limit` (optional) : Anzahl der Ergebnisse pro Seite; standardmäßig werden 50

Elemente angezeigt. Das Maximum sind 200 Elemente.

- `page` (optional) : Seitennummer der angeforderten Seite; standardmäßig wird die erste Seite angezeigt.
- `to` (optional) : Endzeitpunkt eines Zeitraums; zeigt nur Scrobbles vor diesem Zeitpunkt an. Der Zeitpunkt muss im UNIX-Timestamp Format (Anzahl Sekunden seit 1. Januar 1970) angegeben werden.
- `from` (optional) : Startzeitpunkt eines Zeitraums; zeigt nur Scrobbles nach diesem Zeitpunkt an. Der Zeitpunkt muss im UNIX-Timestamp Format (Anzahl Sekunden seit 1. Januar 1970) angegeben werden.
- `api_key` (benötigt) : Ein Last.fm API Schlüssel.

Listing 2.2 enthält eine Antwort eines Aufrufs der `user.getRecentTracks`-Schnittstelle. Hier wird zum Beispiel angezeigt, ob der Titel direkt von Last.fm angehört werden kann (`<streamable>`). Wenn ein Titel über einen Stream von Last.fm verfügbar ist, kann dieser von registrierten Benutzern angehört werden. In der angezeigten Seite wird ebenso gekennzeichnet, ob es sich um einen Titel handelt, der momentan vom Benutzer gehört wird (`<track nowplaying="true">`). Da es sich um ein bestimmtes Ereignis handelt, wird zusätzlich auch der Zeitpunkt (`<date>`) angezeigt. Des Weiteren wird der `<mbid>`-Tag angezeigt. Hierbei handelt es sich um den *MusicBrainz Identifier*[1]. Dieser ist eine eindeutige ID eines Titels und wird von der Musik-Enzyklopädie MusicBrainz[2] bereitgestellt. Er besteht aus 36 Zeichen und kennzeichnet Interpreten, Titel oder Alben eindeutig.

[1] http://musicbrainz.org/doc/MusicBrainz_Identifier - Zugriff: 01.09.2012
[2] http://musicbrainz.org/ - Zugriff: 01.09.2012

```
<lfm status="ok">
 <recenttracks user="RJ" page="1" perPage="10" totalPages="3019">
  <track nowplaying="true">
   <artist mbid="2f9ecbed-27be-40e6-abca-6de49d50299e">Aretha ↙
      Franklin</artist>
   <name>Sisters Are Doing It For Themselves</name>
   <mbid/>
   <album mbid=""/>
   <url>www.last.fm/music/Aretha+Franklin/_/Sisters+Are+Doing+It+ ↙
      For+Themselves</url>
   <date uts="1213031819">9 Jun 2008, 17:16</date>
   <streamable>1</streamable>
  </track>
  ...
 </recenttracks>
</lfm>
```

Listing 2.2: Beispielantwort der `user.getRecentTracks`-Schnittstelle

Methode user.getLovedTracks und user.getBannedTracks

Ähnlich wie die Methode `user.getRecentTracks` bieten die Methoden `user.getLovedTracks` und `user.getBannedTracks` die Möglichkeit, Titel anzuzeigen, welche vom Benutzer als geliebt oder gebannt gekennzeichnet wurden. Auch hier ermöglicht Last.fm anzugeben, wie viele Titel pro Seite angezeigt werden und welche Seite angezeigt werden soll. Die Angabe eines Benutzernamens ist auch hier zwingend notwendig.

Die Parameter der Methode ermöglichen es nicht, eine zeitliche Begrenzung wie bei `user.getRecentTracks` anzugeben. Folgende Parameter sind für `user.getLovedTracks` definiert:

- `user` (benötigt) : Der Last.fm Benutzername, dessen geliebte Titel angezeigt werden sollen.
- `limit` (optional) : Anzahl der Ergebnisse pro Seite; standardmäßig werden 50 Elemente angezeigt.
- `page` (optional) : Seitennummer der angeforderten Seite; standardmäßig wird die erste Seite angezeigt.
- `api_key` (benötigt) : Ein Last.fm API Schlüssel.

Die Parameter der Methode `user.getBannedTracks` sind analog definiert.

Listing 2.3 zeigt den Aufbau und die von Last.fm zur Verfügung gestellten Informationen der `user.getLovedTracks` Methode. Hierzu gehört neben dem Titelnamen (`<name>`) und dem Interpreten (`<artist>`) auch der genaue Zeitpunkt des Ereignisses (`<date>`). Zusätzlich werden URLs von Vorschaubilder zum jeweiligen Titel in unterschiedlichen Größen angezeigt (`<image>`). Analog liefert die Methode `user.getBannedTracks` die selbe Struktur der Antwort. Hier wird jedoch ein anderer Startknoten namens `<bannedtracks>` verwendet.

```
<lfm status="ok">
  <lovedtracks user="RJ">
    <track>
      <name>The Glass Prison</name>
      <mbid/>
      <url>www.last.fm/music/Dream+Theater/_/The+Glass+Prison</url>
      <date uts="1216371514">18 Jul 2008, 08:58</date>
      <artist>
        <name>Dream Theater</name>
        <mbid>28503ab7-8bf2-4666-a7bd-2644bfc7cb1d</mbid>
        <url>http://www.last.fm/music/Dream+Theater</url>
      </artist>
      <image size="small">...</image>
      <image size="medium">...</image>
      <image size="large">...</image>
    </track>
    ...
  </lovedtracks>
</lfm>
```

Listing 2.3: Beispielantwort der `user.getLovedTracks`-Schnittstelle

Methode track.getTopTags

Durch die Methode `track.getTopTags` ist es möglich, die meist benutzen Tags eines Titels zu laden. Um die Tags eines Titels zu finden, benötigt die API entweder den Titel und den Interpretennamen oder die `mbid`. Die Methode verfügt außerdem über einen Parameter, der es ermöglicht, die Eingaben des Titelnamens und des Interpreten automatisch zu korrigieren. Hierzu wird Last.fm durchsucht und bei falscher Schreibweise der korrigierte Eintrag zurückgeliefert. Sollte man eine falsche Schreibweise benutzen und diese ist als Titel auf Last.fm gespeichert, werden keine Korrekturen durchgeführt.

Die zur Verfügung gestellten Parameter sind:

- `track` (benötigt (wenn nicht mbid)) : Der Titelname
- `artist` (benötigt (wenn nicht mbid)) : Der Interpretenname
- `mbid` (optional) : Die MusicBrainz ID des Titels
- `autocorrect[0|1]` (optional) : Falsch geschriebene Interpreten und Titel werden korrigiert. Die Korrekturen sind in der Antwort enthalten.
- `api_key` (benötigt) : Ein Last.fm API Schlüssel.

In Listing 2.4 wird die Rückgabe dieser Methode angezeigt. Falls der Titel in der angegebenen Schreibweise nicht gefunden und `autocorrect` gesetzt wurde, wird in der Antwort automatisch die korrigierte Schreibweise (in `<toptags>`) angegeben. Wenn die Methode zum Beispiel mit *Robie Williams* an Stelle von *Robbie Williams* als Interpretname aufgerufen wird und die Autokorrektur aktiviert ist, gibt Last.fm die Antwort wie in Listing 2.4 dargestellt zurück. Zu jedem Tag (`<name>`) wird ein zusätzlicher Knoten (`<count>`) angezeigt, welcher einen Wert zwischen 0 und 100 annehmen kann. Dieser zeigt an, wie häufig der Tag mit diesem Titel verbunden wurde. Die Formel, welche diesen Wert berechnet, wurde von Last.fm nicht veröffentlicht.

```
<lfm status="ok">
  <toptags artist="Robbie Williams" track="Let Me Entertain You">
    <tag>
      <name>pop</name>
      <count>100</count>
      <url>www.last.fm/tag/pop</url>
    </tag>
    <tag>
      <name>rock</name>
      <count>33</count>
      <url>www.last.fm/tag/dance</url>
    </tag>
    ...
  </toptags>
</lfm>
```

Listing 2.4: Beispielantwort der track.getTopTags Schnittstelle

3. Recommender Systeme

Recommender Systeme bieten die Möglichkeit, dem Benutzer auf Basis von bereits bewerteten Objekten Empfehlungen zu geben. Des Weiteren stellen sie einen sinnvollen Schnittpunkt zwischen Forschung und Wirtschaft dar: Die Forschung arbeitet daran, einem Benutzer immer passendere Empfehlungen zu geben, wodurch inbesondere die Wirtschaft profitieren kann. Diese versucht mit Hilfe von Recommender Systemen eine Art persönliche Beratung zu simulieren. Durch bessere und genauere Empfehlungen können Kunden zum Beispiel animiert werden, ein anderes Produkt beim gleichen Online-Shop zu kaufen, wodurch dieser einen höheren Umsatz erzielt.

Für ein Recommender System benötigt man idealerweise eine große Menge an Benutzern, Objekten und Bewertungen. Jeder Benutzer kann einem Objekt eine persönliche Bewertung geben. Ein Objekt ist eine Entität, über welche Informationen gespeichert und verarbeitet werden können. Zur Bewertung von Objekten wird oftmals eine Likert-Skala [4, 5] angewendet (Tab. 3.1). Diese Skala bietet die Möglichkeit, persönliche Präferenzen durch Zahlen auszudrücken. So kann zum Beispiel eine Matrix wie in Tab. 3.2 entstehen. Ein Eintrag in der Tabelle bildet hier die Bewertung eines Benutzers für dieses Objekt. Für ein Recommender System besteht die Aufgabe darin, aus der Menge von Benutzern, Objekten und Bewertungen neue Empfehlungen für einen Benutzer zu berechnen.

hasse ich	**mag ich nicht**	**geht so**	**mag ich**	**liebe ich**
1	2	3	4	5

Tabelle 3.1.: Beispiel für eine Likert-Skala

Um Empfehlungen für noch nicht bewertete Objekte eines Benutzers zu berechnen, wird analysiert, welche Objekte der Benutzer bereits bewertet hat und welche Bewertung er diesem Objekt gegeben hat. Bei sehr großen Objekt- und Benutzerzahlen ist die Berechnung neuer Empfehlungen sehr aufwändig, da zu jedem noch nicht bewerteten Objekt ein Prediction-Wert berechnet werden muss. Der Prediction-Wert

	Robbie Williams	**Madonna**	**Rolling Stones**	**Rammstein**
Alice	5	4	3	1
Bob	4	-	-	1
Carol	2	-	5	-
David	1	-	5	5

Tabelle 3.2.: Bewertungsmatrix von Benutzern und Objekten

sagt aus, wie hoch die Übereinstimmung zwischen den Vorlieben des Benutzers und den Eigenschaften des Objektes ist.

Recommender Systeme können anhand der Methode, wie der Prediction-Wert berechnet wird, unterschieden werden. Es wird unterschieden zwischen [6]:

1. **Content-based Recommender Systeme**: Um den Prediction-Wert eines Objektes zu berechnen, werden typische Eigenschaften der Objekte miteinander verglichen. Dadurch können für jeden Benutzer Durchschnittswerte aus den numerisch dargestellten Werten einer Eigenschaft errechnet werden.
2. **Collaborative Recommender Systeme**: Die Idee dieser Recommender ist, Benutzer mit ähnlichen Vorlieben zu nutzen, um neue Empfehlungen berechnen zu können.
3. **Hybride Recommender Systeme**: Um die Bewertungen eines Objektes zu berechnen, wird versucht, eine Kombination aus verschiedenen Recommender-Algorithmen zu erstellen. Diese sollen die jeweiligen Vor- und Nachteile kompensieren.

In der Literatur wird das Finden der bestmöglichen Empfehlungen auch als Filterproblem bezeichnet. Daher werden Content-based Recommender auch als Content-based Filtering und Collaborative Recommender als Collaborative Filtering bezeichnet [6]. Im Weiteren werden die einzelnen Methoden erklärt und deren Stärken und Schwächen aufgezeigt.

3.1. Content-based Recommender Systeme

Bei Content-based Recommender Systemen werden Eigenschaften eines Objektes benutzt, um neue Empfehlungen zu berechnen. Die Eigenschaften werden meist automatisiert aus dem Objekt selbst extrahiert. Im Falle von Musiktiteln werden

charakteristische Eigenschaften gefiltert, darunter die Geschwindigkeit des Titels, das Geschlecht des Interpreten oder das Genre eines Titels. Es gibt jedoch auch Ansätze, die versuchen, manuell die Eigenschaften von Titeln zu erkennen. Hierzu hört eine Person einen Titel an und beantwortet zeitgleich einen Fragebogen, in dem verschiedene Eigenschaften abgefragt werden. Diese Methode wird von Pandora.com, einer Online-Radio-Station, genutzt [7, 8].

Anhand der unterschiedlichen Eigenschaften der Titel, die ein Benutzer gehört hat, werden Empfehlungen berechnet. Hierfür wird aus den extrahierten Eigenschaften für jeden Titel ein Vektor erzeugt. Jedes Element des Vektors steht hierbei für eine Eigenschaft des Objektes. Die Eigenschaften sollten voneinander unabhängig und normalisiert sein. Dies vereinfacht die Verwendung des Vektors zur Berechnung der Empfehlung.

	Male	**Pop**	**Rock**	**Metal**	**Schnell**	**Instrumental**
Robbie Williams	1	1	0	0	0	0
Madonna	0	1	0	0	1	1
Rolling Stones	1	1	1	0	1	0
Rammstein	1	0	1	1	1	0

Tabelle 3.3.: Extrahierte Eigenschaften von unterschiedlichen Titeln

In Tab. 3.3 ist ein Beispiel einer solchen Extraktion von Eigenschaften eines Titels dargestellt. Eine “1” bedeutet, dass der Titel die Eigenschaft erfüllt. Eine “0” sagt aus, dass der Titel die Eigenschaft nicht besitzt. Soll die mögliche Bewertung eines Benutzers (Prediction-Wert) vorhergesagt werden, ist der naivste Ansatz, einen Durchschnitt über die Vektoren der bereits gehörten Titel zu errechnen und diesen als Vektor des Benutzers zu speichern. Sobald der Vektor des Benutzers berechnet wurde, kann dieser mit den bereits gespeicherten Vektoren der Titel verglichen und hieraus eine Ähnlichkeit berechnet werden. Die Berechnung des Ähnlichkeitswertes zwischen Benutzer u und Objekt i ist wie folgt festgelegt (Formel 3.1).

$$r(u,i) = score(BenutzerVektor(u), ObjektVektor(i)) \tag{3.1}$$

Die eigentliche Berechnung der Übereinstimmung zwischen Benutzer und einem Titel wird in der Regel mit der Kosinus-Ähnlichkeit berechnet, da diese in sehr vielen unterschiedlichen Szenarien gute Ergebnisse liefert [9]. Hierbei gibt die Kosinus-Ähnlichkeit den Kosinus des Winkels zwischen zwei mehrdimensionalen Vektoren

an. Wenn der Vektor des Benutzers als $\vec{v}_u$ und der Vektor eines Titels als $\vec{v}_i$ definiert ist, wird die Kosinus-Ähnlichkeit wie folgt beschrieben:

$$r(u,i) = cos(\vec{v}_u, \vec{v}_i) = \frac{v_u \cdot v_i}{\|v_u\| \|v_i\|} = \frac{\sum_{k=1}^{n} v_{k,u} \cdot v_{k,i}}{\sqrt{\sum_{k=1}^{n} (v_{k,u})^2} \cdot \sqrt{\sum_{k=1}^{n} (v_{k,i})^2}} \tag{3.2}$$

Im Folgenden wird die Funktionsweise eines Content-based Recommender Systems exemplarisch erläutert: In Tab. 3.2 ist zu erkennen, dass Carol bereits *Robbie Williams* und die *Rolling Stones* bewertet hat. Die erfüllten Eigenschaften von *Robbie Williams* und *Rolling Stones* werden mit der Bewertung multipliziert, welche Carol für diese Interpreten gegeben hat. Das heißt, um den Vektor von Carol zu berechnen, wird der Durchschnitt zwischen *Robbie Williams* und den *Rolling Stones* berechnet. Jeder Wert des Vektors wird mit dem Durchschnitt der gehörten Titel berechnet. Da die Eigenschaft *Male* von den Titeln erfüllt werden (siehe Tab. 3.3), bildet sich der erste Wert durch den Durchschnitt beider Bewertungen (2 für *Robbie Williams* + 5 für *Rolling Stones* = Wert 3,5). Berechnet man den ganzen Vektor für Carol sieht er letztlich wie folgt aus:

$$\vec{v}_{Carol} = (3.5, 3.5, 2.5, 0, 2.5, 0)$$

Aus dem berechneten Vektor des Benutzers lassen sich die Vorlieben von Carol ablesen. Carol präferiert Titel von männlichen Interpreten, welche die Eigenschaft *Pop* erfüllen. Da Carol bereits zwei Titel gehört hat, müssen als nächstes die Werte für *Madonna* und *Rammstein* berechnet werden. Diese Berechnung wird mit Hilfe von Formel 3.2 im Folgenden exemplarisch durchgeführt:

$$
\begin{aligned}
\vec{v}_{Rammstein} &= (1,0,1,1,1,0) \\
\vec{v}_{Madonna} &= (0,1,0,0,1,1) \\
\vec{v}_{Carol} &= (3.5,3.5,2.5,0,2.5,0) \\
r(\vec{v}_{Carol},\vec{v}_{Rammstein}) &= \frac{(3.5\cdot 1)+(2.5\cdot 1)+(2.5\cdot 1)}{\sqrt{37}\cdot\sqrt{4}} = \frac{8.5}{12.166} \\
&= \mathbf{0.699} \\
r(\vec{v}_{Carol},\vec{v}_{Madonna}) &= \frac{(3.5\cdot 1)+(2.5\cdot 1)}{\sqrt{37}\cdot\sqrt{3}} = \frac{6}{10.536} \\
&= \mathbf{0.569}
\end{aligned}
$$

Da bei Rammstein eine höhere Übereinstimmung (0.699) mit den Vorlieben von Carol berechnet wurde, wird ihr als Nächstes *Rammstein* vorgeschlagen.

3.1.1. Grenzen

Auch wenn sich mit Content-based Recommender Systemen gute Vorhersagen generieren lassen, gibt es einige Grenzen, wodurch diese nicht überall einsetzbar sind [6].

Automatisierte Analyse

Um die Eigenschaften eines Objektes zu identifizieren, muss es in irgendeiner Form zur Analyse bereit liegen. Soll die Analyse eines Objektes automatisiert durchgeführt werden, muss eine digitale Version des Objektes vorhanden sein. Insbesondere bei Musiktiteln ist eine automatische Analyse sinnvoll, da sich mit unterschiedlichen Methoden viele Eigenschaften extrahieren lassen [10, 11]. Ein Ansatz wie bei Pandora.com, bei dem jeder Titel von nur einer Person auf Eigenschaften überprüft wird, ist hinsichtlich der Objektivität und der dafür benötigten Arbeitszeit in vielen Fällen nicht geeignet. Bei einer manuellen Analyse der Titel muss, um eine neue Eigenschaft zu extrahieren, das Objekt noch einmal von einer Person betrachtet werden, was deutlich zeitaufwändiger ist.

Qualität der vorliegenden Daten

Insbesondere bei einer automatischen Analyse von Musiktiteln spielt die Qualität der vorliegenden Daten eine wichtige Rolle. Bei alten Aufnahmen von Musiktiteln, die

durch mehrfaches Konvertieren in verschiedene Formate eine minderwertige Qualität haben, kann es vorkommen, dass eine automatisierte Analyse nur wenige Eigenschaften extrahieren kann.

Coldstart Problem

Da der Content-based Recommender auf die Hörhistorie des Benutzers zurückgreift, kann er für neue Benutzer keine Empfehlungen berechnen. Ebenso problematisch stellt sich das Hinzufügen von unvollständig analysierten Musiktiteln heraus. Wenn zu wenige Eigenschaften eines Titels bekannt sind, kann die Ähnlichkeit zwischen Benutzern und dem Musiktitel nicht berechnet werden. Dies hat zur Folge, dass ein solcher Titel nie empfohlen wird.

Überspezialisierung

Ein weiteres Problem ist die Überspezialisierung des Recommenders. Hat ein Benutzer viele Titel mit ähnlichen Eigenschaften bewertet, bietet der Recommender stets weitere Titel an, die diesen Eigenschaften entsprechen. Sollte sich der Geschmack eines Nutzers jedoch über die Zeit ändern, wird dies in der Berechnung der Empfehlungen nur geringfügig berücksichtigt, da der Großteil der Bewertungen sich am alten Geschmack des Benutzers orientiert.

3.2. Collaborative Recommender Systeme

Ein typisches Verhalten bei der Auswahl von Filmen oder Büchern ist es, Freunde nach ihren Vorlieben zu fragen. Sollten die Freunde ähnliche Vorlieben haben, so erhält man durch diese sehr gute Empfehlungen. Je besser sich Personen untereinander kennen, umso genauer können Empfehlungen sein, welche von diesen gegeben werden.

Eine vergleichbare Funktionsweise verfolgt auch ein Collaborative Recommender System. Es findet Benutzer, welche ähnliche Vorlieben haben und berechnet auf dieser Grundlage neue Empfehlungen.

Um zu wissen, was die beste Empfehlung für einen Benutzer ist, muss für jedes Objekt ein Wert ausgerechnet werden, der prognostiziert, wie gut dem Benutzer dieses Objekt gefallen könnte. Die Wissenschaft unterscheidet bei der Berechnung dieser Werte zwei algorithmische Ansätze: Zum einen den *memory-based* Ansatz, welcher auf den bereits gespeicherten Bewertungen anderer Benutzer beruht und diese für

die Berechnung eines Empfehlungswertes benutzt. Zum anderen gibt es den *model-based* Ansatz, welcher zunächst aus gespeicherten Daten von anderen Benutzern ein Modell erstellt. Dieses Modell wiederum wird genutzt, um die Empfehlungswerte vorhersagen zu können. Dieser Ansatz wird jedoch in dieser Arbeit nicht weiter betrachtet, da dies ein weiteres großes Forschungsgebiet darstellt und über den Rahmen dieser Arbeit hinaus gehen würde.

Der *memory-based* Ansatz lässt sich in zwei unterschiedliche Methoden unterscheiden: Da dieser Ansatz auf der Basis von Bewertungen von Benutzern aufgebaut ist, können Empfehlungen aufgrund von Benutzern oder Objekten berechnet werden. So gibt es zum einen den User-Item Collaborative Recommender und zum anderen den Item-Item Collaborative Recommender. Der User-Item Collaborative Recommender nutzt Ähnlichkeiten zwischen unterschiedlichen Benutzern, um neue Empfehlungen für einen Benutzer generieren zu können. Im Gegensatz dazu steht der Item-Item Collaborative Recommender, der anhand der Bewertungen Ähnlichkeiten zwischen unterschiedlichen Objekten berechnet und diese nutzt, um Empfehlungen für den Benutzer zu generieren.

Die bestmöglichen Empfehlungen kann ein Recommender System machen, wenn nahezu jedes Objekt, welches in der Datenbank gespeichert ist, von jedem Benutzer bewertet wurde. Da Benutzer jedoch oftmals nur einige wenige Objekte bewerten, entstehen sogenannte schwachbesetzte Matrizen (Tab. 3.4). Eine Bewertung ist in dieser Matrix als Punkt gekennzeichnet, wohingegen bei einem Strich keine Bewertung vorliegt. Eine Bewertung des Objektes i von Benutzer u wird mit der Funktion $r(u, i)$ abgerufen.

	i_1	i_2	i_3	$\cdots$	i_k	$\cdots$	i_n
u_1	-	•	-		•		-
u_2	•	-	-		-		•
$\vdots$							
u_j	•	-	•		$r(u_j, i_k)$		•
$\vdots$							
u_m	-	-	•		•		-

Tabelle 3.4.: Bewertungsmatrix von Benutzern auf unterschiedliche Elemente

Durch die Bewertungsmatrix in Tab. 3.4 kann die Menge der bewerteten Objekten eines Nutzers als $u_j = [r(u_j, i_1), r(u_j, i_2), r(u_j, i_3), ..., r(u_j, i_n)]$ gele-

sen werden. Alle Bewertungen eines bestimmten Objektes können als $i_k = [r(u_1, i_k), r(u_2, i_k), r(u_3, i_k), ..., r(u_m, i_k)]$ gelesen werden.

Im Folgenden werden die zwei unterschiedlichen Ansätze des User-Item Collaborative Recommenders und des Item-Item Collaborative Recommenders näher betrachtet und erklärt.

3.2.1. User-Item Collaborative Recommender

Ein User-Item Collaborative Recommender verwendet die Bewertungen anderer Benutzer, um Empfehlungen zu berechnen. Hierbei werden zu jedem Benutzer Nachbarn ermittelt, welche ähnliche Vorlieben haben. Im nächsten Schritt wird für noch nicht bewertete Objekte berechnet, wie gut dieses Objekt den Vorlieben des Benutzers entspricht. Bei der Berechnung werden die Bewertungen der Nachbarn berücksichtigt.

Um entscheiden zu können, ob sich ein Benutzer in der Nachbarschaft eines anderen Benutzers befindet, wird zu Beginn eine Funktion $sim(u_A, u_B)$ definiert. Um Benutzer zu finden, welche ähnliche Vorlieben haben, muss zuerst die Schnittmenge von unterschiedlichen Benutzern berechnet werden (Formel 3.3). Durch diese Vorlieben lassen sich Objekte finden, welche sowohl von Benutzer u_A als auch von Benutzer u_B bewertet wurden. Hierbei bezeichnet die Menge I die Menge *aller* Objekte.

$$S(u_A, u_B) = \{i \in I | r(u_A, i) \neq \varnothing \wedge r(u_B, i) \neq \varnothing\} \tag{3.3}$$

Es können zwei unterschiedliche Formeln genutzt werden, um die Ähnlichkeit zweier Benutzer zu berechnen: Zum einen die bereits vorgestellte Kosinus-Ähnlichkeit, zum anderen die Pearson-Korrelation. Die Kosinus-Ähnlichkeit nutzt wie auch in Tab. 3.5 dargestellt die Bewertungen anderer Benutzer. Mit Hilfe der Formel 3.4 kann die Ähnlichkeit zweier Benutzer u_A und u_B berechnet werden.

$$\begin{aligned} sim(u_A, u_B) &= cos(\vec{u}_A, \vec{u}_B) = \frac{u_A \cdot u_B}{\|u_A\| \|u_B\|} \\ &= \frac{\sum\limits_{i \in S(u_A, u_B)} r(u_A, i) \cdot r(u_B, i)}{\sqrt{\sum\limits_{i \in S(u_A, u_B)} (r(u_A, i))^2} \cdot \sqrt{\sum\limits_{i \in S(u_A, u_B)} (r(u_B, i))^2}} \end{aligned} \tag{3.4}$$

	i_1	i_2	i_3	⋯	i_k	⋯	i_n
u_1	•	•	-		•		-
u_2	-	•	-		-		-
⋮							
u_j	•	-	-		$r(u_j, i_k)$		-
⋮							
u_m	-	-	-		•		•

Tabelle 3.5.: Beispiel einer Berechnung einer Bewertung durch einen User-Item Collaborative Recommender

Jedoch wird durch die Kosinus-Ähnlichkeit nicht die unterschiedlichen Bewertungsspanne unterschiedlicher Benutzer berücksichtigt. Es ist möglich, dass zwei unterschiedliche Benutzer zwar die gleichen Vorlieben haben, jedoch die eingesetzte Skala anders interpretieren und hierdurch jeder Titel anders bewertet wird. Ein optimistischer Benutzer wird für seine Lieblingstitel häufiger höhere Bewertungen geben, während ein eher pessimistischer Benutzer für seine Lieblingstitel lediglich eine mittlere Bewertung gibt. Um die unterschiedlichen Skalen der Benutzern zu berücksichtigen, wird die Pearson-Korrelation (Formel 3.5) benutzt [12]. Hierbei wird die durchschnittliche Bewertung eines Benutzers u_A als $\bar{r}(u_A)$ definiert.

$$sim(u_A, u_B) = \frac{\sum\limits_{i \epsilon S(u_A, u_B)} (r(u_A, i) - \overline{r}(u_A)) \cdot (r(u_B, i) - \overline{r}(u_B))}{\sqrt{\sum\limits_{i \epsilon S(u_A, u_B)} (r(u_A, i) - \overline{r}(u_A))^2} \cdot \sqrt{\sum\limits_{i \epsilon S(u_A, u_B)} (r(u_B, i) - \overline{r}(u_B))^2}} \quad (3.5)$$

Sowohl die Kosinus-Ähnlichkeit als auch die Pearson-Korrelation haben einen Wertebereich zwischen -1 und +1. Hierbei kennzeichnet eine +1 die perfekte Ähnlichkeit zwischen zwei Benutzern, während -1 einen maximalen Unterschied kennzeichnet. Herlock et al. [13] konnten zeigen, dass die Berücksichtigung negativer Ähnlichkeiten von Benutzern zu keiner Verbesserung der Empfehlungen führte und infolgedessen für Berechnungen ignoriert werden kann.

Nachdem die Ähnlichkeit von Benutzern berechnet wurde, kann die Menge der zur Berechnung verwendeten Benutzern bestimmt werden. Diese Entscheidung wird meist hinsichtlich der Geschwindigkeit der Berechnungen getroffen. Ein oft genutzter Ansatz beruht darauf, einen Schwellenwert festzulegen, durch den die Anzahl der

Benutzer begrenzt wird. So können alle Benutzer zur Berechnung ignoriert werden, deren Ähnlichkeit unter einer bestimmten Grenze liegen. Insbesondere bei zeitkritischen Recommendern kann außerdem das Setzen einer festen Anzahl an Benutzern sinnvoll sein, da bei der Verwendung eines Schwellenwertes nicht vorhersagbar ist, wie viele Benutzer in die Berechnungen einfließen werden.

Um im nächsten Schritt die fehlenden Bewertungen des Benutzers zu berechnen, werden die Bewertungen aller Nachbarn ($nachbarn(u_A)$) für das aktuelle Objekt berücksichtigt. Als Nachbarn werden alle Benutzer definiert, die eine hohe Ähnlichkeit mit dem Benutzer u_A haben oder deren Ähnlichkeit über einem bestimmten Schwellenwert liegt. Ein naiver Ansatz wie in Formel 3.6 kann zwar zur Berechnung der fehlenden Werte genutzt werden, berücksichtigt jedoch nicht die unterschiedlichen Skalen der Benutzer. Hierfür wird $I(u_A)$ als die Menge aller vom Benutzer u_A noch nicht bewerteten Objekte definiert.

$$\begin{aligned} I(u_A) &= \{i | r(u_A, i) = 0\} \\ \forall i' \in I(u_A), r(u_A, i') &= \frac{\sum\limits_{u_n \in nachbarn(u_A)} r(u_n, i')}{\#Nachbarn} \end{aligned} \tag{3.6}$$

Die Formel 3.6 kann um die durchschnittliche Bewertung $\bar{r}(u)$ eines Benutzers erweitert werden. Somit werden die unterschiedlichen Skalen berücksichtigt (Formel 3.7).

$$\forall i' \in I(u_A), r(u_A, i') = \bar{r}(u_A) + \frac{\sum\limits_{u_n \in nachbarn(u_A)} sim(u_A, u_n) \cdot (r(u_n, i') - \bar{r}(u_n))}{\sum\limits_{u_n \in nachbarn(u_A)} sim(u_A, u_n)} \tag{3.7}$$

Im Anschluss können alle berechneten Bewertungen eines Benutzers für alle Objekte eingesehen werden, um daraus Empfehlungen für einen Benutzer generieren zu können. Hierzu muss diese Liste absteigend nach ihrer Bewertung sortiert werden. Auch hier bietet sich an, entweder einen Schwellenwert für die Empfehlungen zu setzen oder eine feste Anzahl an gewünschten Empfehlungen zu nutzen. Da bei sehr vielen Objekten und Nutzern die Anzahl von Objekten über einem Schwellenwert schlecht vorhersagbar ist, wird zumeist eine feste Anzahl an Empfehlungen gewählt.

Zur Reduzierung der Berechnungszeit ist es gerade bei Echtzeitsystemen sinnvoll, die Nachbarn eines Benutzers vorzuberechnen. Da sich die Bewertungsmatrix geringfügig ändert, reicht es aus, die Nachbarn in regelmäßigen Abständen neu zu berechnen. Hierdurch kann ein Recommender in Echtzeit-Szenarien eingesetzt werden, da die Berechnung von Empfehlungen deutlicher schneller wird.

3.2.2. Item-Item Collaborative Recommender

Der Item-Item Collaborative Recommender verwendet eine ähnliche Strategie wie der User-Item Collaborative Recommender. Anstatt jedoch anhand der Nachbarn neue Empfehlungen zu berechnen, wird die Ähnlichkeit zwischen verschiedenen Objekten genutzt, um neue Empfehlungen zu berechnen.

Die Funktionsweise besteht aus folgenden Schritten: Zu Beginn wird zu einem Objekt eine Menge von ähnlichen Objekten berechnet. Aus dieser Menge errechnet man, mit den bereits gespeicherten Bewertungen, einen Prediction-Wert für jedes Objekt. Zuletzt müssen aus den berechneten Werten die Empfehlungen ausgewählt werden.

Da bei dieser Methode keine Benutzer miteinander verglichen werden, muss eine neue Funktion $sim(i_A, i_B)$ definiert werden. Hierbei bezeichnet die Menge U die Menge aller Benutzer. Auch hier kann eine Pearson-Korrelation verwendet werden, welche auf die verwendete Bewertungsspanne eines Benutzers angepasst wird (Formel 3.8). Mit Hilfe von $U(i_A, i_B)$ werden Benutzer gefunden, welche sowohl i_A als auch i_B bewertet haben.

$$
\begin{aligned}
U(i_A, i_B) &= \{u \in U | r(u, i_A) \neq \varnothing \wedge r(u, i_B) \neq \varnothing\} \\
sim(i_A, i_B) &= \frac{\sum\limits_{u \in U(i_A, i_B)} (r(u, i_A) - \bar{r}(u)) \cdot (r(u, i_B) - \bar{r}(u))}{\sqrt{\sum\limits_{u \in U(i_A, i_B)} (r(u, i_A) - \bar{r}(u))^2} \cdot \sqrt{\sum\limits_{u \in U(i_A, i_B)} (r(u, i_B) - \bar{r}(u))^2}}
\end{aligned}
\tag{3.8}
$$

Für die Berechnung der Bewertung eines einzelnen Objektes muss - anders als beim User-Item Collaborative Recommender - nicht die Durchschnittsbewertung eines Benutzers miteinbezogen werden, da alle Bewertungen vom gleichen Benutzern kommen. Daher kann die Formel 3.9 genutzt werden, um die Bewertung für ein Objekt zu berechnen, welches der Benutzer noch nicht bewertet hat. Hierzu muss die Funktion $\bar{I}(u_A)$ definiert werden, welche alle Objekte beinhaltet, die der Benutzer u_A

bereits bewertet hat.

$$\bar{I}(u_A) = I \setminus I(u_A)$$
$$\forall i' \in I(u_A), r(u_A, i') = \frac{\sum\limits_{i \in \bar{I}(u_A)} sim(i', i) \cdot r(u_A, i)}{\sum\limits_{i \in \bar{I}(u_A)} sim(i', i)} \tag{3.9}$$

Um die Geschwindigkeit bei Echtzeitrecommendersystemen zu verbessern, bietet es sich an, die Mengen der berücksichtigten Objekte zu begrenzen. Dies kann entweder mit einem Schwellenwert oder mit einer fest ausgewählten Anzahl an Objekten erzielt werden. Hierbei sollten immer die k Objekte ausgewählt werden, welche die höchste Übereinstimmung mit dem zu vergleichenden Objekt haben. Dieser Ansatz wird als *top-k* Ansatz bezeichnet. Des Weiteren bietet es sich an, die Berechnung der Ähnlichkeiten zwischen den unterschiedlichen Objekten in eine Vorberechnung auszulagern und diese in regelmäßigen Abständen neu zu berechnen.

3.2.3. Grenzen

Collaborative Recommender Systeme haben Grenzen, welche im Folgenden dargestellt werden [6].

Coldstart-Problem

Wenn eine Plattform neu startet und ein Collaborative Recommender zum Einsatz kommen soll, wird dieser anfangs keine Empfehlungen generieren können. Ein Collaborative Recommender ist auf Bewertungen von Benutzern angewiesen und da zu Beginn einer Plattform keine Wertungen vorliegen, können keine Empfehlungen berechnet werden.

Zu wenige Bewertungen

Besitzt ein Collaborative Recommender keine Informationen über Eigenschaften der Objekte und berücksichtigt nur die Bewertungen von Benutzern, können bei einer zu schwach besetzten Matrix keine Ähnlichkeiten zwischen den Benutzern und Objekten gefunden werden.

Berechnung

In Echtzeitsystemen ist ein Collaborative Recommender ohne weitere Optimierungen nur eingeschränkt einsetzbar, da für die Berechnungen neuer Empfehlungen zu

Beginn alle ähnlichen Objekte oder Benutzer berechnet werden müssen. Anschließend muss für jedes Objekt ein Prediction-Wert berechnen werden. Insbesondere das Berechnen der ähnlichen Objekte oder Benutzer ist eine rechenintensive Aufgabe, welche bei Content-based Recommendern nicht erforderlich ist.

Eine mögliche Optimierung bei Collaborative Recommendern besteht darin, so viele Berechnungen wie möglich bereits im Vorfeld auszuführen. Auch durch die Begrenzung der zu beachtenden Objekte bei der Berechnung kann die Komplexität und Rechenzeit reduziert werden.

3.3. Hybride Recommender

Zu der Kategorie der hybriden Recommendern gehören Kombinationen von mehreren Recommendern. Ziel ist es, die Vor- und Nachteile unterschiedlicher Recommender ausnutzen, um bessere Empfehlungen generieren zu können.

Burke [14] unterscheidet sieben unterschiedliche hybride Recommender, welche auf einer Metaebene folgenden drei Kategorien zugewiesen werden können [6]:

1. Kombination von unterschiedlichen Recommendern
2. Verbesserung eines Haupt-Recommenders durch ein anderen Recommender
3. Erstellung eines Recommenders, welcher Vorzüge mehrerer Recommender berücksichtigt

Die erste Kategorie mit Kombinationen unterschiedlicher Recommender enthält vier unterschiedliche Ansätze: Zum einen der gewichtete (*weighted*) Ansatz (Abb. 3.1), welcher mehrere Recommender nutzt und die einzelnen Werte der Recommender zu einer Gesamtwertung gewichtet. Zum anderen gibt es einen wechselnden (*switching*) Ansatz (Abb. 3.2), der anhand der aktuellen Situation entscheidet, welcher implementierte Recommender genutzt wird, um Empfehlungen zu generieren. Auch der gemischte (*mixed*) Ansatz (Abb. 3.3) wird dieser Kategorie zugeordnet. Er nutzt unterschiedliche Recommender, welche abhängig von ihren errechneten Empfehlungen eine Liste aus allen benutzten Recommendern zusammenstellen. Der kaskadierte (*cascade*) Ansatz (Abb. 3.4) ist der letzte Ansatz dieser Kategorie. Mit Hilfe eines ersten Recommenders werden Bewertungen berechnet, welche durch einen zweiten Recommender verfeinert werden.

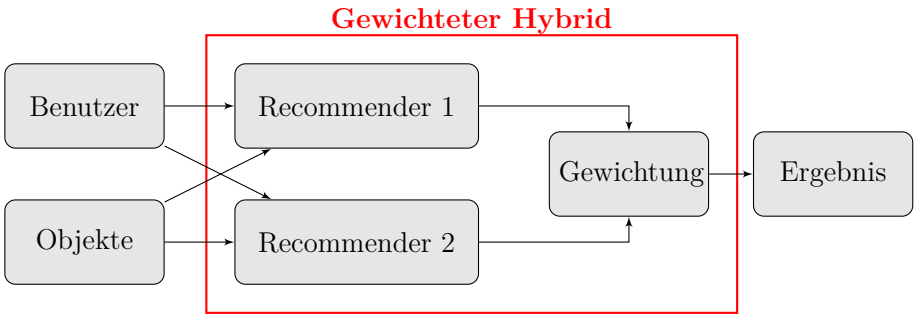


Abbildung 3.1.: Gewichteter (weighted) Ansatz (aus [15, Seite 382])

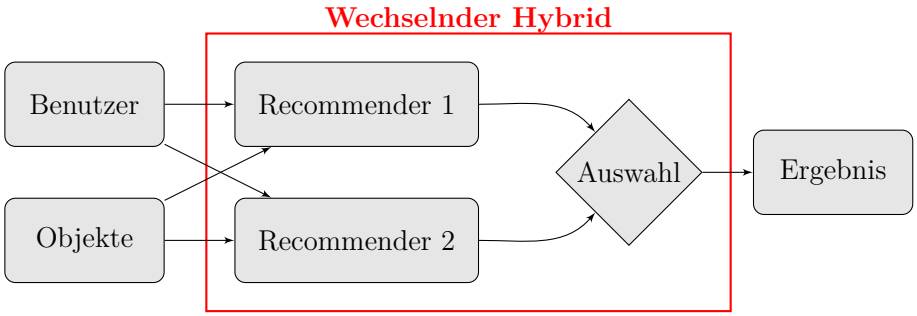


Abbildung 3.2.: Wechselnder (switching) Ansatz (aus [15, Seite 385])

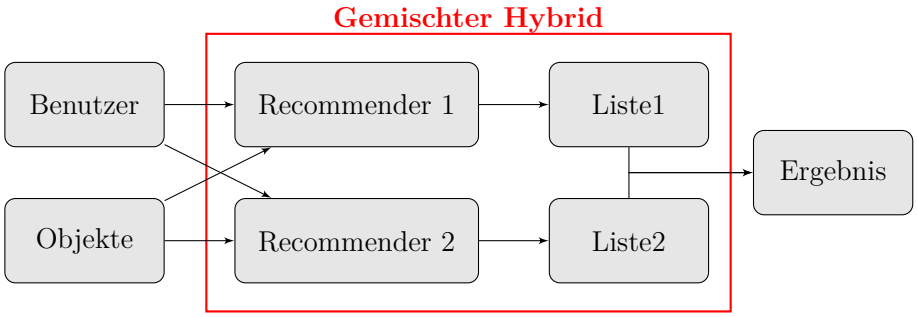


Abbildung 3.3.: Gemischter (mixed) Ansatz (aus [15, Seite 384])

In die zweite Kategorie fallen Recommender, deren Empfehlung durch einen zusätzlichen Recommender verbessert werden. Den ersten Ansatz bildet die Merkmal-Kombination (*feature combination*) (Abb. 3.5). Hierbei wird im Gegensatz zu der ersten Kategorie meist ein Hauptrecommender verwendet, welcher für seine Berechnung auf Eigenschaften eines zusätzlichen Recommenders zurückgreift. Der zweite Ansatz beruht auf der Steigerung entdeckter Eigenschaften durch eine Merkmal-Anreicherung (*feature augmentation*) (Abb. 3.6). Hierzu können Informationen ei-

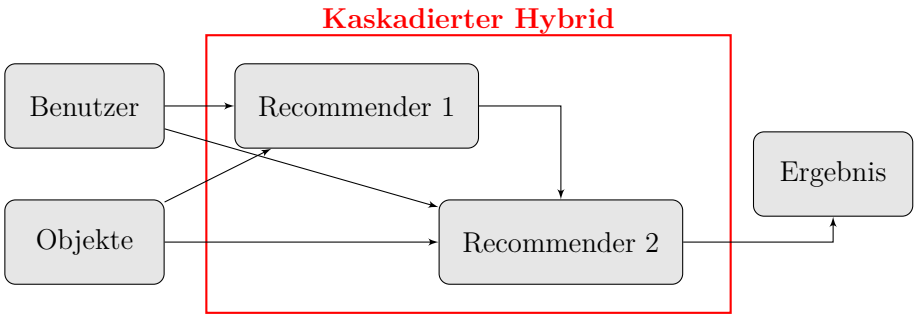


Abbildung 3.4.: Kaskadierter (cascade) Ansatz (aus [15, Seite 390])

nes Recommenders als zusätzliche Eingabe für einen zweiten Recommender benutzt werden. Dazu zählen auch kombinierte Recommender, die das gelernte Modell eines weiteren Recommenders nutzen. Der dritte Ansatz besteht aus dem *Meta-level* Hybrid (Abb. 3.7), welcher aus einem Hauptrecommender und einem unterstützenden Recommender besteht. Der unterstützende Recommender lernt die Profile der Benutzer und reicht diese als Eingabe an den Hauptrecommender weiter. Die gelernten Profile kann der Hauptrecommender nur verarbeiten, wenn ein modellbasierter Algorithmus verwendet wird.

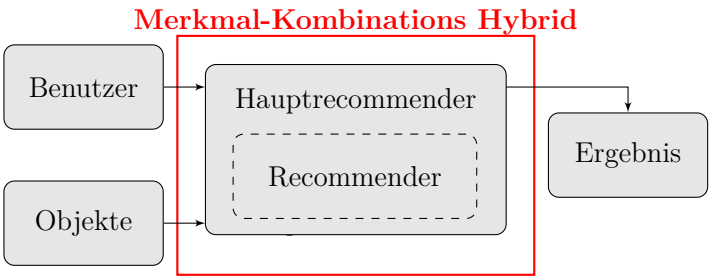


Abbildung 3.5.: Merkmal-Kombination (feature combination) (aus [15, Seite 387])

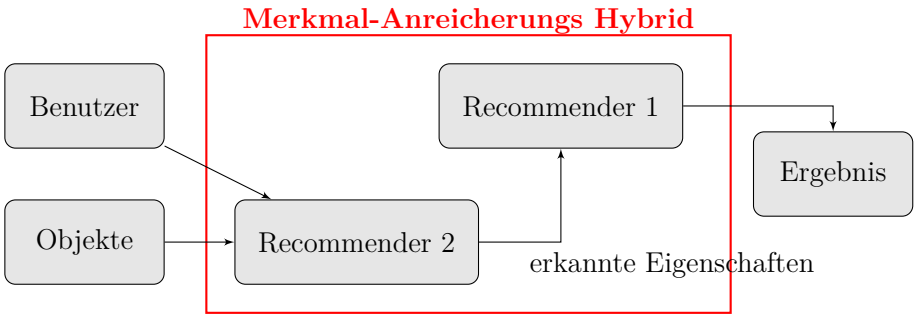


Abbildung 3.6.: Merkmal-Anreicherung (feature augmentation) Ansatz (aus [15, Seite 388])

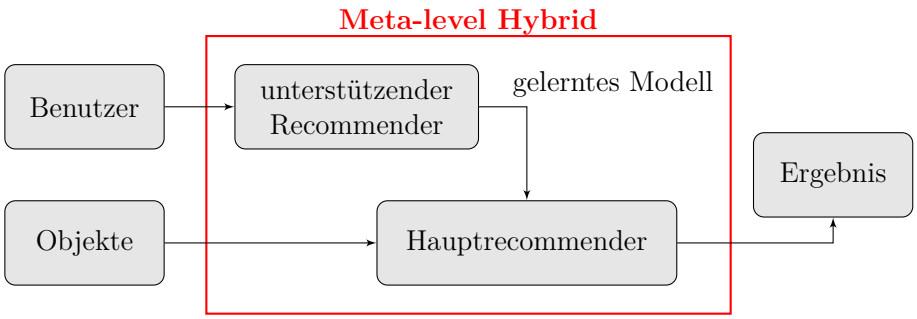


Abbildung 3.7.: Meta-level Ansatz (aus [15, Seite 391])

Die dritte Kategorie enthält vereinende Recommender (*unifying*). Diese können die Eigenschaften mehrerer Recommender nutzen, um Empfehlungen zu generieren. Da der Durchschnitt der Empfehlungen von mehreren Recommendern oft besser ist, als ein einzelner Recommender, sind diese immer häufiger ein Schwerpunkt der Forschung [3]. Diese Recommender sind meist sehr domänenspezifisch und berücksichtigen spezielle Eigenschaften, welche nur in dieser Domäne auftreten.

4. Crawler

In dieser Arbeit wurde ein hybrider Recommender auf der Basis von Titeln, Benutzern und deren Hörverhalten entwickelt. Um einen hybriden Recommender zu entwickeln, werden Informationen über Objekte, Personen und Bewertungen benötigt. Durch die von Last.fm zur Verfügung gestellten Schnittstellen (siehe Kapitel 2) konnten von Benutzern Informationen über gehörte Titel geladen werden. So konnte ein Datensatz generiert werden (Kapitel 5), welcher als Eingabe für den später implementierten TRecs (Kapitel 6) verwendet wurde.

4.1. Ziele/Anforderungen

Die benötigten Daten, um einen hybriden Recommender zu entwickeln, bestehen aus dem Hörverhalten der Benutzer und Zusatzinformationen zu den gehörten Titeln. In einer Matrix, in welcher Benutzer und Titel aufgetragen sind, müssen genügend Informationen über gehörte Titel vorhanden sein, um deren Ähnlichkeiten berechnen zu können.

Die Daten, welche für eine Empfehlung von neuen Titeln benötigt wurden, stellten sich aus Hörereignissen von Benutzern und Informationen zu Titeln zusammen. Die Hörereignisse von Benutzern konnten über die API von Last.fm abgerufen werden. Hierbei wurden nicht nur gehörte Titel gesammelt, sondern auch die favorisierten und gebannten Titel eines Benutzers. Zu jedem Ereignis wurde zusätzlich der genaue Zeitpunkt abgespeichert. Die Informationen eines Titels setzten sich aus dem Titelnamen und dem Interpretennamen zusammen. Zusätzlich wurde, wenn diese vorhanden war, zu jedem Titel eine Liste von Tags gespeichert.

Um die Ergebnisse eines Durchlaufes bereits vorab kontrollieren zu können, sollte der Crawler in der Lage sein, angehalten zu werden und zu einem späteren Punkt an der gleichen Stelle wieder fortgesetzt werden können. Die Parallelisierung des Craw-

lens durch die Unterstützung mehrere Threads sollte den Durchsatz an abgerufenen Daten steigern.

Ziel des Crawlens bestand darin, so viele Hörereignisse wie mögliche von unterschiedlichen Benutzern zu sammeln. Auf eine vollständige Liste der Hörereignisse aller Benutzer wurde verzichtet, da für den hybriden Recommender nicht die vollständige Hörhistorie eines Benutzers gespeichert sein muss. Die Anzahl der maximalen Hörereignisse pro Benutzer setzte sich aus einem ersten Test zusammen, bei dem der Durchschnitt mehrerer Benutzer berechnet wurde. Hierbei wurde festgestellt, dass ein Benutzer ca. 10.000 Hörereignisse im Jahr generiert. Da im Rahmen dieser Arbeit nur die letzten zwei Jahre eines Benutzers gesammelt wurden, sind demnach nur 20.000 Titel pro Benutzer gecrawlt worden. Auch wurde die Anzahl der geliebten und gebannten Titel pro Benutzer auf 1.000 Titel pro Benutzer begrenzt.

4.2. Architektur

Für die Speicherung der gecrawlten Daten wurde ein MySQL-Server eingesetzt. Es wurde das Datenbankmodell (Abb. 2.4, Seite 12) verwendet, welches um die jeweiligen URLs der Elemente erweitert wurde. Es bestand aus folgenden Relationen: Ein Titel (tracks) hat genau einen Interpreten (artist). Ein Titel kann mehrere Tags (tags) haben, welche mit einem bestimmten Gewicht zu einem Titel zugewiesen werden (trackTags). Ein Titel kann von einem Benutzer (user) zu einem bestimmten Zeitpunkt markiert werden. Der Benutzer kann einen Titel entweder zu einer Liste von favorisierten (lovedTracks), gebannten (bannedTracks) oder kürzlich gehörten Titeln (recentTracks) hinzufügen. Benutzer können zudem Freundschaften mit anderen Benutzern (userUsers) eingehen.

Die verwendeten Datenbanktabellen werden in Anhang A aufgeführt.

Das Crawlen der Daten setzte sich aus verschiedenen Schritten zusammen: Zuerst wird der aktuell zu bearbeitende Benutzer aus der Datenbank geladen und als “in Bearbeitung” markiert. Anschließend wurden alle Freunde eines Benutzers mit Hilfe der Last.fm-API geladen und in die Datenbank gespeichert. Dies diente zur Generierung einer Warteschlange, sodass eine Breitensuche realisiert werden konnte. Sobald alle Freunde gespeichert waren, wurden alle favorisierten, gebannten und kürzlich gehörten Titel eines Benutzers geladen. Wenn festgestellt wurde, dass es sich um einen neuen Titel handelte, wurde der Titel in die Datenbank gespeichert und gleichzeitig

alle verfügbaren Tags zu dem Titel von Last.fm geladen und gespeichert. In Abb. 4.1 ist eine Übersicht über den Ablauf des Crawlens abgebildet. Eine detailierte Ansicht mit den Aufrufen der Last.fm API ist in Abb. 4.2 zu sehen.

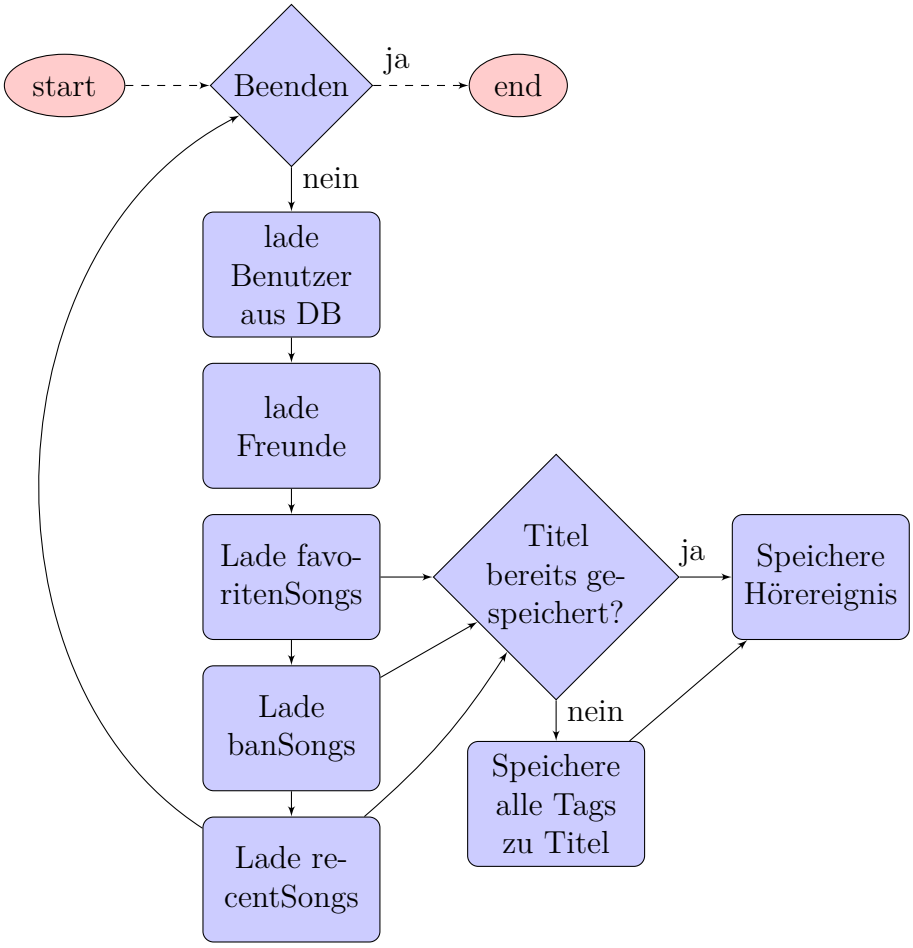


Abbildung 4.1.: Ablauf des Crawlers

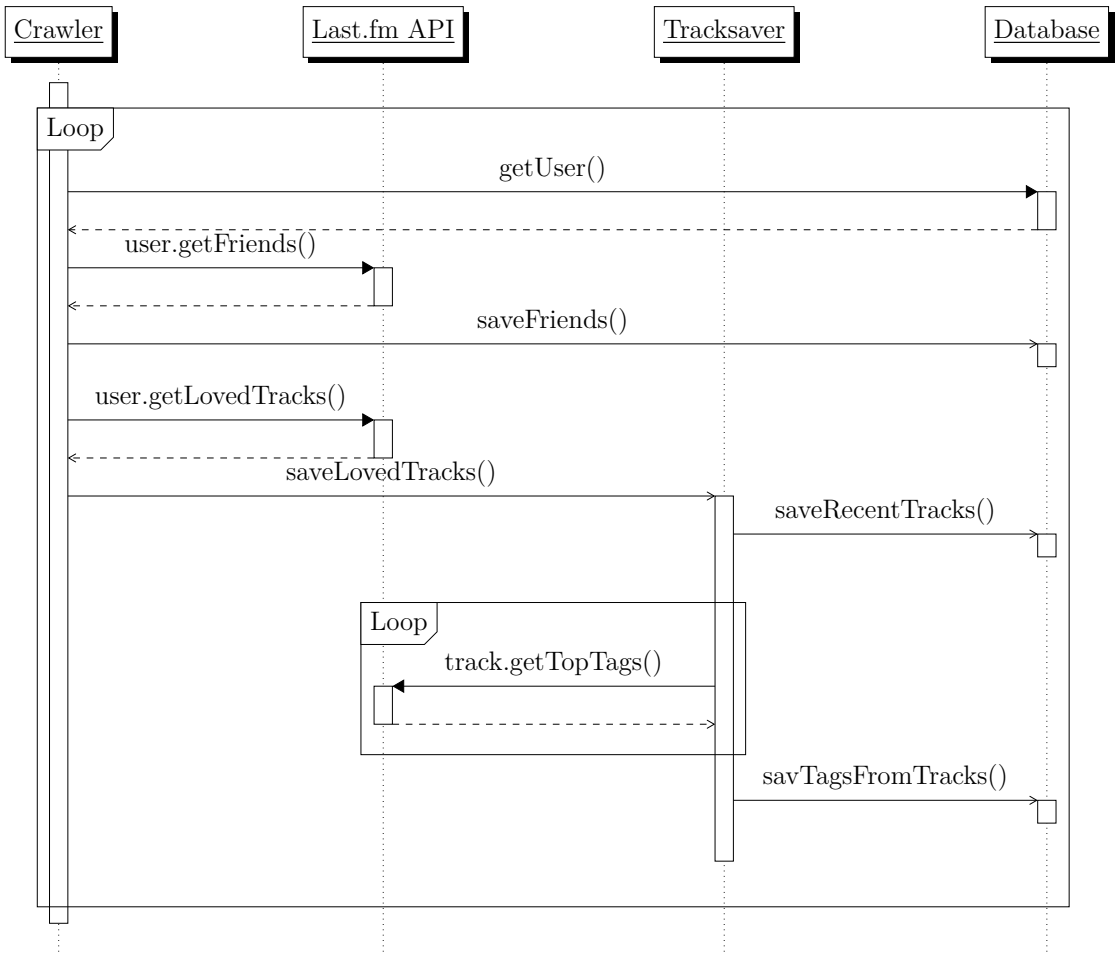


Abbildung 4.2.: Ablaufausschnitt des Crawlers bis zum Laden der favorisierten Titel eines Benutzers

4.3. Ergebnis

Mit Hilfe des Crawlers wurden zwei unterschiedliche Datensätze generiert: Der erste Datensatz wurde innerhalb von zwei Wochen erstellt und beinhaltete ungefähr 41 Millionen Hörereignisse. Um eine genaue Auswertung zu erstellen, wurde im ersten Teil ein Logging-System aktiviert, welches die Tabellengrößen der Datenbank in regelmäßigen Abständen protokollierte. Nach der Analyse der vorliegenden Daten konnte die Performanz des Crawlers verbessert und verschiedene Bugs behoben werden. Im zweiten Durchlauf wurde auf die Aktivierung eines Logging-Systems verzichtet, um zusätzlich die Performanz des Crawlers zu optimieren. So konnten im zweiten Durchlauf in ungefähr der gleichen Zeit über 51 Millionen Hörereignisse gespeichert werden.

Um zu zeigen, wie sich die unterschiedlichen Tabellen im Verlauf der Zeit füllten, wird der erste Datensatz verwendet.

In Abb. 4.3 ist zu erkennen, dass die Anzahl der gecrawlten User linear anstieg. Abschnitte, in denen sich die Steigung ändert, sind darauf zurückzuführen, dass zu diesem Zeitpunkt viele Benutzer gecrawlt wurden, welche nur wenige Hörereignisse aufwiesen.

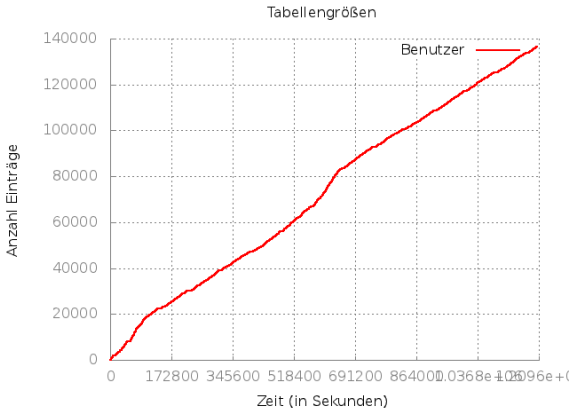


Abbildung 4.3.: Anzahl der Benutzer über die Zeit

Die Anzahl der gespeicherten Tags lässt sich in Abb. 4.4 ablesen. Zu Beginn des Crawlens waren keine Tags in der Datenbank vorhanden. Dadurch wurden beim Start des Crawlers viele unbekannte Tags gefunden und dementsprechend die meisten Tags gespeichert. Umso länger der Crawler lief, desto weniger neue Tags wurden gefunden und in die Datenbank gespeichert. Mit zunehmender Anzahl an gespeicherten Hörereignissen und Titeln wurden immer häufiger dieselben, bereits in der Datenbank gespeicherten Tags verwendet.

Der Crawler speicherte durch die angewandte Strategie immer weniger neue Titel. Zu Beginn wurden, analog zu den Tags, die meisten Titel eingefügt (siehe Abb. 4.5). Es ist zu erkennen, dass die Anzahl der Titel nicht so schnell stieg wie die Anzahl der Tags.

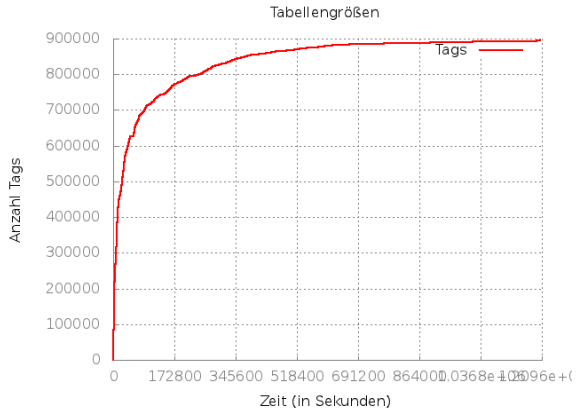


Abbildung 4.4.: Anzahl der Tags über die Zeit

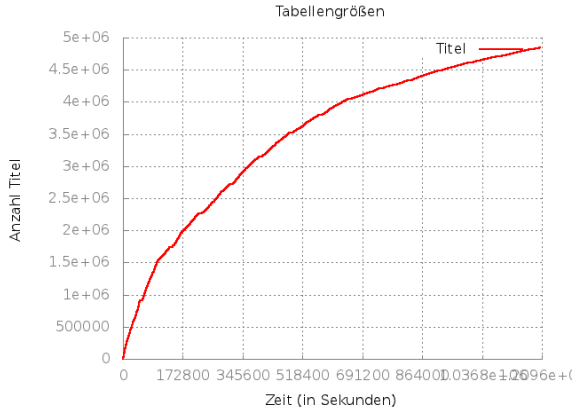


Abbildung 4.5.: Anzahl der Titel über die Zeit

4.4. Zusammenfassung

In diesem Kapitel wurde beschrieben, wie der Crawler aufgebaut war und welche Daten in der Datenbank gespeichert wurden. Es wurden zwei unterschiedliche Datensätze generiert: Der erste Datensatz eignete sich dafür, die Performanz des Crawlers abzuschätzen und den Verlauf der Tabellengrößen über die Zeit zu erfassen. Der zweite Datensatz wurde für die Entwicklung des Recommenders verwendet und wird im nachfolgenden Kapitel näher vorgestellt.

5. Datensatz

In diesem Kapitel wird der Datensatz, der mit Hilfe des Crawlers (Kapitel 4) erstellt wurde, näher beschrieben. Es werden verschiedene Eigenschaften dieses Datensatzes beleuchtet und zudem erklärt, wie der Datensatz genutzt werden kann, um die Wissensbasis des Track Recommender Systems (TRecs) aufzubauen. Des Weiteren wird aufgezeigt, welche Selektionsstrategie genutzt wurde, um aus den gecrawlten Titeln eine Untermenge zu wählen, welche bezüglich relevanter charakteristischen Eigenschaften ähnlich zu der ursprünglich gecrawlte Titelmenge ist.

Mit Hilfe des Crawlers wurden insgesamt 4.400 (4.474) unterschiedliche Benutzer geladen. Alle Benutzer hatten summiert 140.000 (142.269) Freunde, die ebenso in der Datenbank abgelegt wurden. Insgesamt wurden 51.000.000 (50.821.753) Hörereignisse generiert, welche sich auf 4.800.000 (4.796.847) unterschiedliche Titel aufteilten. Es wurden Titel von 570.000 (569.890) verschiedenen Interpreten gehört. Da viele Titel auch Tags hatten, konnten insgesamt 900.000 (904.948) Tags gecrawlt und gespeichert werden. Die Tags wurden mit 17.000.000 (17.153.367) Nennungen den Titeln zugeschrieben. Des Weiteren konnten 870.000 (873.386) geliebte Titel und 90.000 (89.589) gebannte Titel gespeichert werden.

5.1. Data Cleaning

Bereits bei den ersten Versuchen, eine Untermenge der Titel zu finden, zeigte sich ein Problem: Teilweise gab es über 100 unterschiedliche Versionen des selben Titels. Im Folgenden werden diese Titel als Dubletten bezeichnet. Eine Dublette konnte bei den Titeln erkannt werden, indem der Interpret bei unterschiedlichen Titeln gleich war. Wie in einem Auszug in Tab. 5.1 zu sehen ist, waren die Titelnamen meist gleich formatiert. Dies bedeutet, dass zum Beispiel die Nennung der Version (*Album Version*, *Single Version*, *Radio Version*) immer am Ende des Titelnamen standen. Auch die Kennzeichnung von Zusammenarbeiten mit anderen Interpreten wurden,

wenn sie im Titelnamen genannt wurden, immer nach dem eigentlichen Titelnamen genannt und durch ein *featuring*, *feat.* oder *ft* gekennzeichnet (Tab. 5.2).

Interpret	**Titel**
Lady Gaga	Bad Romance
Lady Gaga	Bad Romance (2009)
Lady Gaga	Bad Romance (Acapella)
Lady Gaga	Bad Romance (Anandose Rmx)
Lady Gaga	Bad Romance (Bimbo Jones Vocal Mix)
Lady Gaga	Bad Romance (Bimbo Jones Clean Radio Remix)
Lady Gaga	Bad Romance (Bimbo Jones Radio Edit)
Lady Gaga	Bad Romance (clip version)
Lady Gaga	Bad Romance (Dave Audio Radio Mix)
Lady Gaga	Bad Romance (Demo Version)
Lady Gaga	Bad Romance (Maxi)
Lady Gaga	Bad Romance [Official Music Video] [HQ]
Lady Gaga	Bad Romance (OST Gossip Girl)
Lady Gaga	Bad Romance (Radio)
Lady Gaga	Bad Romance (Radio Edit)
Lady Gaga	Bad Romance (Radio Edit Album Version)
Lady Gaga	Bad Romance (Remix)
Lady Gaga	Bad Romance (Short Radio Edit)
Lady Gaga	Bad Romance (Video Version)

Tabelle 5.1.: Verschiedene Schreibweisen eines Titels

Um die Reduzierung der Titelanzahl so effizient wie möglich durchzuführen, wurden die Titel nach Interpetenname und Titelname sortiert. Anschließend wurden sämtliche Klammern, Urls und Domainnamen aus dem Titel entfernt. Analog wurde mit Titelerweiterungen wie *featuring* oder *feat.* verfahren (siehe Tab. 5.2). Im darauf folgenden Schritt wurde über die bearbeitete Liste iteriert und überprüft, ob der aktuelle Titel den anderen Titeln des Interpreten ähnelte. Wenn dies der Fall war, wurde der aktuelle Titel als Dublette gekennzeichnet und eine Referenz auf den ersten Titel hinzugefügt (siehe Tab. 5.3).

ID	Interpret	Titel	Bearbeiteter Titel
512	Shakira	Loca (Featuring Dizzee R...	Loca
513	Shakira	Loca (feat. El Cata)	Loca
514	Shakira	Loca (feat. El Cata) (Span...	Loca
515	Shakira	Loca (feat. Dizzie Rascal)...	Loca
516	Shakira	Loca (Feat.Dizzee Rascal)...	Loca
517	Shakira	Waka Waka (Club Mix)	Waka Waka
518	Shakira	Waka Waka (Club Remix)...	Waka Waka
519	Shakira	Waka Waka (feat. Freshlyg...	Waka Waka
520	Shakira	Waka Waka (Time For Afr...	Waka Waka
521	Shakira	Waka Waka (This Time F...	Waka Waka

Tabelle 5.2.: Ausschnitt einer Titelliste eines Interpreten

ID	DubId	Interpret	Bearbeiteter Titel
512	512	Shakira	Loca
513	512	Shakira	Loca
514	512	Shakira	Loca
515	512	Shakira	Loca
516	512	Shakira	Loca
517	517	Shakira	Waka Waka
518	517	Shakira	Waka Waka
519	517	Shakira	Waka Waka
520	517	Shakira	Waka Waka
521	517	Shakira	Waka Waka

Tabelle 5.3.: Nach der Bearbeitung der Titelnamen

Dadurch wurden unterschiedliche Versionen eines Titels zusammengefasst. Dies hatte für TRecs den Vorteil, dass zu einem Titel nicht nur die ursprüngliche Anzahl an Hörereignissen zur Verfügung standen, sondern die Hörereignisse jeder einzelnen Version zur späteren Berechnung der Empfehlungen verwendet werden konnte. Durch diese Methode konnten auch unterschiedliche Versionen eines Titels, wie zum Beispiel die normale Version und eine Elektro-Version, zusammengeführt werden. Obgleich sich die Hörergruppe der Titel eventuell unterschieden, wurde angenommen, dass unterschiedliche Hörergruppen dennoch Interesse an dem Original hatten.

In dieser Arbeit wurde die Jaro-Winkler Distanz (Formel 5.1) angewendet, um zusätzliche Dubletten zu erkennen. Diese gibt in Abhängigkeit der Ähnlichkeit zwischen zwei Zeichenketten einen Wert zwischen 0 und 1 zurück. Hierin unterscheidet sie sich von anderen Metriken, wie zum Beispiel der Levensthein-Distanz [16], da

diese die Anzahl der Transpositionen, der Zeichenersetzungen, zwischen zwei Zeichenketten ausgibt und somit nicht die Länge der Zeichenketten berücksichtigt wird. Hierbei wird m als die Anzahl der übereinstimmenden Zeichen definiert und t als die Hälfte der Transpositionszahl, der Anzahl der Zeichenersetzungen. Des Weiteren wird die Länge des gemeinsamen Präfixes l definiert. Es wird ein maximaler Wert von 4 definiert, welcher auch in dieser Arbeit genutzt wurde [16]. Zuletzt wird der konstante Skalierungsfaktor p definiert, welcher den Standardwert 0.1 hat. Insbesondere eignet sich die Jaro-Winkler Distanz für den Vergleich von Namen, da die gemeinsame Präfix-Länge beachtet wird. So können etwaige Tippfehler oder Zeichenvertauschungen erkannt und berücksichtigt werden [17, 16, 18].

Die Jaro-Winkler Distanz besteht aus vier unterschiedlichen Teilen: Zu Beginn werden die Längen der Zeichenketten berechnet. Anschließend wird die Anzahl der übereinstimmenden Zeichen in beiden Zeichenketten bestimmt. Nachfolgend wird die Anzahl der benötigten Ersetzungen von Buchstaben berechnet. Zuletzt muss die Länge des gemeinsamen Präfixes ermittelt werden.

$$
\begin{aligned}
d_j &= \frac{1}{3}\left(\frac{m}{|s_1|} + \frac{m}{|s_2|} + \frac{m-t}{m}\right) \\
d_w &= d_j + (l * p(1 - d_j))
\end{aligned}
\tag{5.1}
$$

Um die Jaro-Winkler Distanz anhand eines Beispiels zu berechnen, wird im Folgenden Ähnlichkeit der Titel *I Was Made For Loving You Baby* und *I Was Made For Lovin' You* von *KISS* berechnet. Zu Beginn werden die Zeichenlängen der Titelnamen bestimmt, welche in diesem Beispiel $s_1 = 25$ und $s_2 = 30$ sind. Anschließend wird die Anzahl der übereinstimmenden Zeichen m berechnet. Hierzu wird die Zeichenkette s_1' berechnet, welche alle Zeichen aus s_1 beinhaltet, die auch in s_2 vorkommen. Analog wird s_2' definiert, in der alle Zeichen von s_2 enthalten sind, die auch in s_1 vorkommen. In unserem Beispiel gleichen sich s_1' und s_2', welche sich aus den 24 Zeichen ($IWasMadeForLovinYou$) zusammensetzen. Somit wird $m = 24$ gesetzt. Anschließend wird die Transpositionszahl zwischen s_1' und s_2' ermittelt. Diese wird mit Hilfe der gemeinsam genutzten Zeichen bestimmt. Da in unserem Beispiel die benötigte Anzahl an Transpositionen, um von s_1' auf s_2' zu kommen, gleich ist, wird die Transpositionszahl auf 0 gesetzt. Anschließend wird der berechnete Wert in die Formel 5.1 eingesetzt, wodurch man d_j erhält. Dieser Wert wird weiter in der Be-

rechnung von d_w verwendet. Der Berechnungsweg ist im Folgenden einzusehen:

$$
\begin{aligned}
|s_1| &= |I\ Was\ Made\ For\ Lovin'\ You| = 25 \\
|s_2| &= |I\ Was\ Made\ For\ Loving\ You\ Baby| = 30 \\
m &= 24 \\
t &= \frac{0}{2} = 0 \\
d_j &= \frac{1}{3}\left(\frac{24}{25} + \frac{24}{30} + \frac{24-0}{24}\right) = 0.92 \\
d_w &= 0.92 + (4 * 0.1(1 - 0.92)) = 0.952
\end{aligned}
\tag{5.2}
$$

Mit der Jaro-Winkler Distanz ergibt sich eine Ähnlichkeit beider Titel von 0.952. Somit kann die Aussage getroffen werden, dass es sich bei den beiden Titeln sehr wahrscheinlich um den selben Titel handelt.

Durch die verwendete Methode war es möglich, die 4.800.000 Titel auf ca. 3.700.000 Titel zu minimieren. Dies bedeutet, dass ca. 22% der gecrawlten Titel als Dubletten markiert werden konnten. Mit Hilfe dieser Markierungen gelang es, von den ursprünglichen ca. 50.000.000 Hörereignissen 12.500.000 Hörereignisse neuen Titeln zuzuweisen. Dies entspricht ca. 24% der Hörereignisse. Die durchschnittliche Anzahl der Hörereignisse konnte so von 10,4 Hörereignissen pro Titel auf 13,8 Hörereignissen pro Titel gesteigert werden.

Die Interpreten wurden in dieser Arbeit nicht weiter zusammengefasst. Grund dafür war, dass bei Interpreten oftmals minimale Änderungen der Schreibweise zu einem neuen existierenden Interpreten führten. Dadurch war eine Zusammenfassung in diesem Punkt nicht praktikabel durchführbar.

5.2. Analyse

Insgesamt wurden über 4.400 Benutzer gecrawlt. Im Durchschnitt hatte ein gecrawlter Benutzer 31 Freunde. Die Reihenfolge der zu crawlenden Benutzer wurde durch eine Breitensuche bestimmt. In Abb. 5.1 bis Abb. 5.3 ist zu erkennen, wie die Breitensuche die Reihenfolge der gecrawlten Benutzer beeinflusste.

Abbildung 5.1.: Freundschaftsgraph vom Start-Benutzer mit der Tiefe 1

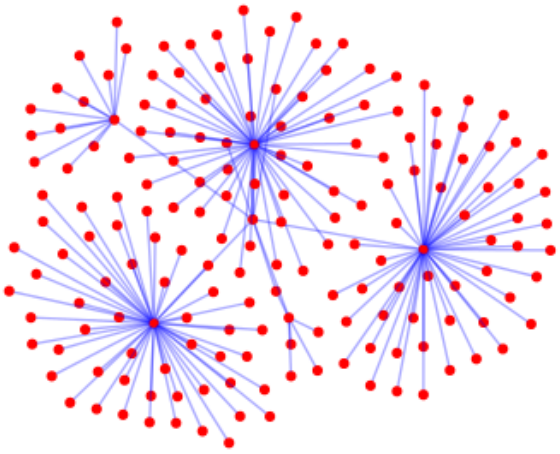


Abbildung 5.2.: Freundschaftsgraph vom Start-Benutzer mit der Tiefe 2

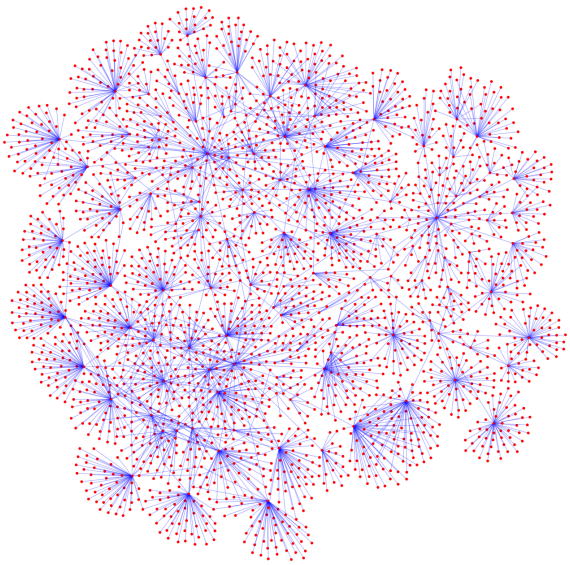


Abbildung 5.3.: Freundschaftsgraph vom Start-Benutzer mit der Tiefe 3

Abb. 5.4 zeigt, wie oft ein Titel gehört wurde. Es ist zu erkennen, wie viele Titel wie oft gehört wurden. Ein hoher Punkt auf der linken Seite zeigt, wie viele Titel nur einmal gehört wurden. In dem gecrawlten Datensatz finden sich demnach viele Titel, die nur einmal gehört wurden. Viele Punkte befinden sich auf der rechten Seite des Diagramms. In der LogLog-Skalierung ist die Anzahl der Hörereignisse pro Titel jedoch deutlich geringer. Somit werden sehr viele Titel nur einmal gehört und nur wenige Titel werden sehr häufig gehört. Es lässt sich vermuten, dass die Titel der rechten Seite auf der Last.fm Plattform sehr populär sind und dies der Grund für die hohe Hörereigniszahl ist. Die drei meist gehörten Titel in den gecrawlten Daten sind: *Rolling In The Deep* von *Adele*, *Set Fire To Rain* von *Adele* und *Bad Romance* von *Lady Gaga*.

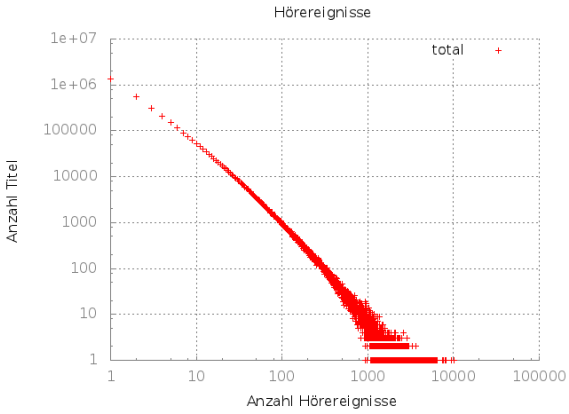


Abbildung 5.4.: Hörhäufigkeit von Titeln

Insgesamt wurden von dem Crawler ca. 900.000 Tags gespeichert. Diese wurden mittels 17 Millionen Nennungen mit 1,7 Millionen Titeln verbunden. Es gibt deutlich mehr Titel ohne Tags als Titel mit Tags. Auch dies wird durch die Funktionsweise von Last.fm ersichtlich: Jeder Benutzer hat auf Last.fm die Möglichkeit, Titel mit einem Tag zu markieren. Sobald ein Tag oft genug von unterschiedlichen Benutzern genannt wurde, wird dieser in die Liste der Tags des Titels aufgenommen. Wie bereits geschrieben, gibt es viele Titel, welche durch abweichende Schreibweisen nur von sehr wenigen Benutzern gehört wurden. Da die Tags freiwillig durch Benutzer hinzugefügt werden, ist die Wahrscheinlichkeit, Tags bei einem Titel zu finden, höher, umso mehr Benutzer diesen Titel gehört haben. Es trat ein mittlerer Effekt zwischen Hörereignissen pro Titel und Tags pro Titel ($r = .36$l) auf. Dies bedeutet, dass ein Zusammenhang zwischen der Hörhäufigkeit und der Anzahl der Tags eines Titels bestand. Die Relation zwischen Titeln mit Tags und Titeln ohne Tags wird

in Abb. 5.5 aufgezeigt.

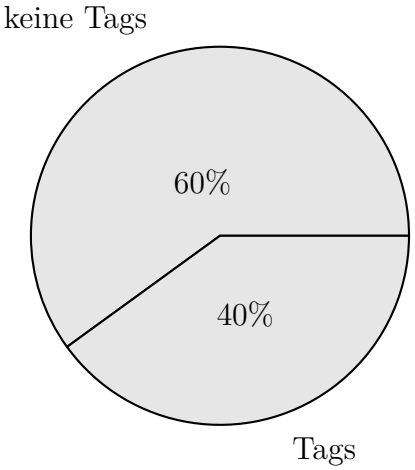


Abbildung 5.5.: Verhältnis der Titel mit und ohne Tags

In Abb. 5.6 ist zu erkennen, wie oft Tags verwendet wurden. Wie auch bei den Titeln kann man hier sehen, dass sehr viele Tags nur einmal verwendet wurden. Nur wenige Tags wurden von Benutzern sehr oft verwendet und mehr als 100.000 Mal verwendet. Dies lässt die Vermutung zu, dass viele Tags Titelinformationen wie zum Beispiel den vollständigen Titelnamen oder den Interpreten beinhalten und hierdurch eine Verwendung bei anderen Titeln ausgeschlossen wird. Unter den meist benutzten Tags finden sich Tags wie: *rock*, *pop*, *electronic*, *indie* und *alternative*.

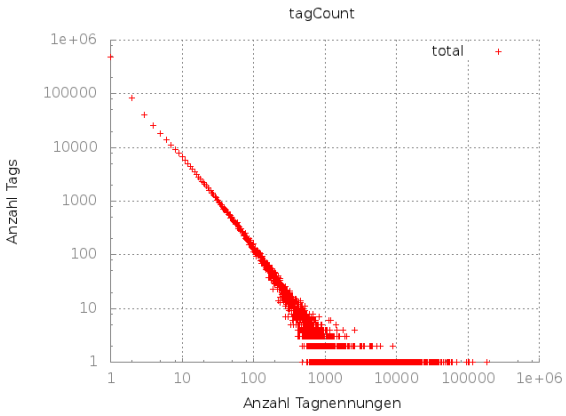


Abbildung 5.6.: Verteilung der Tags auf Titel

Das Hörverhalten der über 4.000 gecrawlten Benutzer in Bezug auf die Tageszeit wird in Abb. 5.7 dargestellt. Es zeigt, dass viele der Benutzer nachts schliefen und

hierdurch um 6 Uhr ein Minimum in den Hörereignissen entstand. Die Anzahl der Hörereignisse stieg über den Tag stetig an, bis sie um 19 Uhr Maximum erreichte. Innerhalb von 11 Stunden sank die Anzahl der Hörereignisse von ihrem Maximum auf das Minimum.

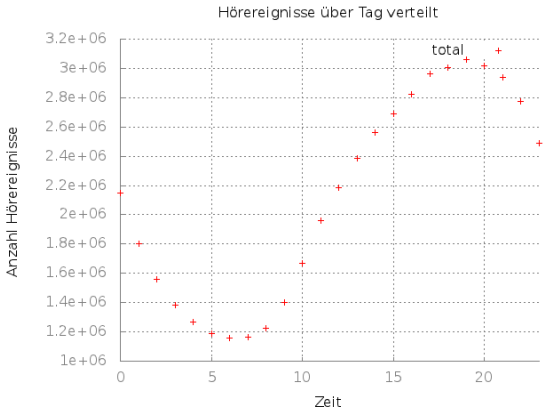


Abbildung 5.7.: Hörzeiten über 24 Stunden

Wie viele Titel durch die gecrawlten Benutzer gehört wurden, ist in Abb. 5.8 dargestellt. Im Diagramm ist die Prozentzahl der Titel im Zusammenhang mit der Benutzerzahl in Prozent abgebildet. Zu sehen ist, dass nur ca. 30% der Benutzer mehr als 2.200 Titel gehört haben. Die maximale Anzahl an Hörereignissen pro Benutzer wurde nur von 227 Benutzern erreicht, was 0,8% der Benutzer entsprach. Dass sich nur wenige Benutzer im Bereich der maximalen Hörereignisse pro Benutzer befanden, begründet sich in der durchschnittlichen Anzahl an Hörereignissen pro Jahr. Durchschnittlich hörte ein Benutzer ungefähr 10.000 Titel im Jahr. Daher wurde in dieser Arbeit ein Maximum von 20.000 Titeln eingeführt, um durchschnittlich die letzten zwei Jahre zu speichern. Nur ungefähr 10,18% der Benutzer haben über 10.000 Hörereignisse generiert. Die Steigung im Bereich der 10.000 Titel kann durch Benutzer erklärt werden, die den Dienst aktiv nutzen, jedoch noch keine zwei Jahre angemeldet sind oder die weniger als 10.000 Titel im Jahr hören. So ist es möglich, dass ein Benutzer bereits seit sieben Jahren angemeldet war, jedoch im Jahr nur 2.000 Titel hörte. Sehr viele Benutzer sind im Bereich der 1 bis 6.000 Hörereignisse, was darauf zurückzuführen sein könnte, dass viele Benutzer nach einer kurzen Testphase von Last.fm den Dienst nicht weiter nutzen, jedoch das Mitgliedskonto weiter besteht.

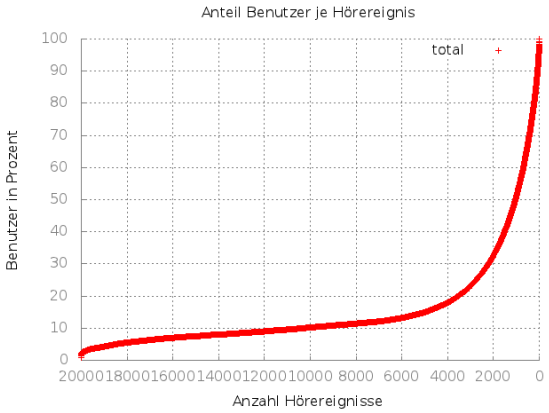


Abbildung 5.8.: Anteil der Benutzer je Anzahl Hörereignisse

5.3. Titelauswahl

Es wurde nur eine Teilmenge der Titel für die Empfehlungen genutzt, um die Anzahl der empfehlbaren Titel von 3,7 Millionen zu begrenzen und den Berechnungsaufwand für neue Empfehlungen zu reduzieren. Die Verwendung einer Teilmenge der Titel sollte es zudem ermöglichen, neue Empfehlungen innerhalb von maximal 30 Sekunden zu berechnen, sodass bei der späteren Evaluation Benutzer nicht zu lange auf neue Empfehlungen warten müssen.

Es musste ein geeignetes Verfahren entwickelt werden, um die Charakteristika der Menge der gecrawlten Titel und der Titelauswahl beizubehalten. Es bestand aus zwei unterschiedlichen Selektionsstrategien: Zum einen wurden Titel mittels der meist genutzten Tags ausgewählt; zum anderen aus den meist gehörten Titeln der Benutzer.

Durch diese Selektionsstrategie konnten mit Hilfe der Benutzer bekannte Titel geladen werden, welche auf Last.fm häufig gehört wurden. Mit der Selektionsstrategie der Tags wurden mitunter viele eher unbekannte Titel der Titelauswahl hinzugefügt, die zwar eine sehr hohe Tagwertung erhielten, jedoch von relativ wenigen Benutzern gehört wurden.

Zunächst wurden alle Titel verworfen, die nur von einem einzigen Benutzer gehört wurden, um die 3,7 Millionen Titel vorab zu reduzieren. Wenn nur ein Benutzer einen Titel gehört hatte, konnten keine Aussagen darüber getroffen werden, wie gut anderen Benutzern der Titel gefallen könnte. So wurden insgesamt ca. 2,2 Millionen Titel markiert, die nur von einem einzelnen Benutzer gehört wurden. Zur Auswahl standen somit ca. 1,5 Millionen Titel - diese entsprachen knapp 40% der ursprüng-

lichen Titelmenge.

Als nächstes wurden für die Selektionsstrategie Titel anhand der verwendeten Tags ausgewählt. Da 40% der Titel mit Tags markiert waren, sollte diese Menge auch in der späteren Titelauswahl vorhanden sein. Zur Selektion der Titel wurden die 100 meist genutzten Tags verwendet.

Zur Auswahl einer geforderten Menge wurde die Anzahl der Titel berechnet, welche durch die Tags ausgewählt wurden. Anteilig wurden anschließend je nach Anzahl der Titel pro Tag Titel aus der Menge selektiert. Es wurde darauf geachtet, dass kein Titel aufgrund unterschiedlicher Tags mehrfach aufgenommen wurde. Von jedem Tag wurden daher diejenigen Titel mit den höchsten Tagwertungen ausgewählt. Analog wurde bei der Selektion der Titel anhand des Hörverhaltens der Benutzer verfahren. Die verbleibenden 60% der geforderten Menge wurden mit Titeln aufgefüllt, welche keine Tags enthielten und meist gehört pro Benutzer waren. Auch hier wurde beachtet, dass ein Titel nicht bereits durch Tags oder andere Benutzer ausgewählt wurde.

Die Vorgehensweise wird im Folgenden an einem Beispiel von 10.000 ausgewählten Titeln beschrieben: Bei 10.000 Titeln werden 4.000 Titel mit Hilfe der Tags und 6.000 Titel mit Hilfe der Benutzer ausgewählt. Zu Beginn werden die 100 meist genutzten Tags verwendet. Angenommen, es gibt vier Tags und deren Benutzung liegt bei 5.000, 3.000, 1.000 und 1.000 Nennungen. Dementsprechend werden durch den ersten Tag 2.000 Titel aus der ursprünglichen Menge an Titeln mit den höchsten Tagwertungen ausgewählt. Durch den zweiten Tag werden durch dieses Verfahren weitere 1.200 Titel selektiert. Durch den dritten und vierten Tag werden jeweils 400 weitere Titel ausgewählt (Abb. 5.9).

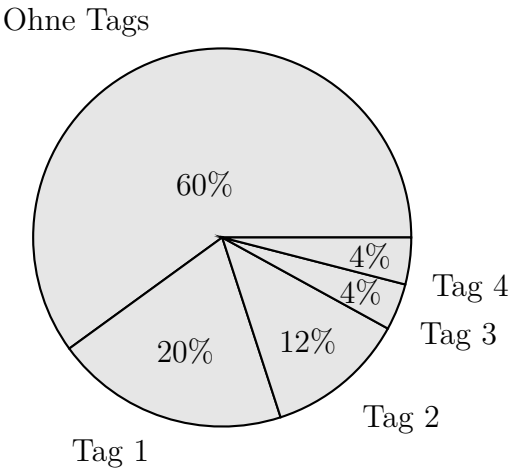


Abbildung 5.9.: Titelselektion anhand von Tagverteilung

Falls ein Titel eine sehr hohe Tagwertung bei Tag 4 hat, jedoch bereits durch einen Tag davor ausgewählt wurde, wird er bei Tag 4 ignoriert und es wird der nächste Titel ausgewählt, welcher noch nicht in der Selektion enthalten ist. Die restlichen 6.000 Titel werden durch die Benutzer ausgewählt. Die Anzahl der Hörereignisse eines Benutzers wird für die Bestimmung der Anzahl an Titeln benötigt, welche über diesen Benutzer ausgewählt werden. Hat ein Benutzer 25% der gesamten Anzahl an Hörereignissen erstellt, so werden hier im Beispiel 1.500 Titel ausgewählt, welche der Benutzer am häufigsten gehört hat. Sollte ein Benutzer zwei Titel gleich oft gehört haben, wird der Titel mit der höheren Anzahl unterschiedlicher Hörer bevorzugt.

Im Folgenden werden die Ergebnisse der Selektionsstrategie auf unterschiedlich große Titelmengen näher betrachtet: Es wurden in dieser Ausarbeiten fünf unterschiedliche Titelauswahlen gewählt. Die Größen der einzelnen Mengen waren: 2.500, 5.000, 10.000, 15.000 und 30.000.

Nachfolgend werden einige Eigenschaften näher betrachtet, um zu überprüfen, ob die ausgewählten Titelmengen die gleichen Eigenschaften wie der gecrawlte Datensatz hatten. In Abb. 5.10 ist auf den ersten Blick zu erkennen, dass die Datenpunkte der selektierten Mengen unterschiedlich aussehen und somit diese Charakteristik nicht beibehalten wurde. Dies lag zum einen direkt an der Vorsortierung: Titel, welche nur einmal gehört wurden, wurden entfernt. Zum anderen bevorzugte die Selektionsstrategie Titel, welche eine hohe Anzahl an Hörereignissen hatten. Da für die Evaluierung absichtlich auf sehr seltene Titel verzichtet wurde, ist diese Abweichung jedoch akzeptabel.

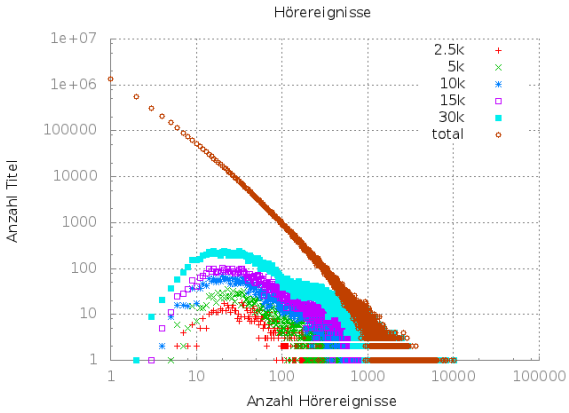


Abbildung 5.10.: Hörhäufigkeit von Titeln

Als nächstes werden, wie bereits vorgestellt, die Uhrzeiten der Hörereignisse der Benutzer für die verschiedenen Datensätze verglichen. In Abb. 5.11 ist zu erkennen, wie sich das Hörverhalten der Benutzer über den Tag verteilte. Auch hier finden sich die Minima und Maxima des gesamten Datensatzes: morgens um 6 Uhr und abends um 19 Uhr. Einen Unterschied findet sich lediglich in der Anzahl der gehörten Titel pro Stunde. So lässt sich bei dem gesamten Datensatz eine Schwankung von 1,9 Millionen Hörereignissen zwischen Minimum und Maximum feststellen. Bei der kleinsten ausgewählten Menge von 2.500 Titeln ist diese Schwankung mit 82.000 deutlich geringer. Der prozentuale Anteil zwischen Maximum und Minimum ist jedoch bei beiden Datensätzen etwa 37 Prozent.

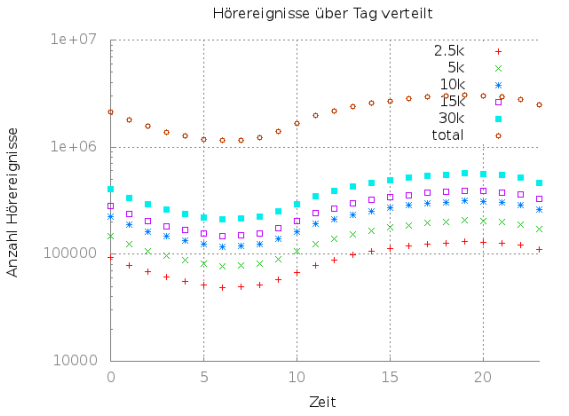


Abbildung 5.11.: Hörverhalten über 24 Stunden

Die Menge der Hörereignissen pro Titel, wie sie in Abb. 5.12 abgebildet ist, unterschied sich zwischen den ausgewählten Mengen (2.500, 5.000, 10.000, 15.000, 30.000

Titel) und der ursprünglichen Menge (3,7 Millionen Titel) deutlich. Da die Selektionsstrategie von jedem Benutzer nur so viele Titel auswählte, wie er anteilig an den gesamten Hörereignissen beteiligt war, zeigte sich ein deutlich anderes Bild: Sehr viele Benutzer haben in der Abbildung nur sehr wenige Titel gehört, was jedoch durch die kleineren selektierten Titelmengen erklärbar ist. Wenn der Musikgeschmack eines Benutzers durch Titel, die durch die Tags ausgewählt wurden, abgedeckt wurde, so hatte dieser eine hohe Anzahl an Hörereignissen, welche durch die Selektionsstrategie ausgewählt wurden. Wenn der Benutzer einen eher exotischeren Musikgeschmack hatte, wurden lediglich die meist gehörten Titel von ihm zur Titelauswahl hinzugefügt. Dadurch hatte er wenige Hörereignisse in den kleinen Titelauswahlen. Aufgrund der vorgestellten Selektionsstrategie werden sehr populäre Musiktitel bevorzugt. Folglich lässt sich hieraus eine Tendenz für den späteren Recommender erkennen, welche populäre Titel bevorzugt.

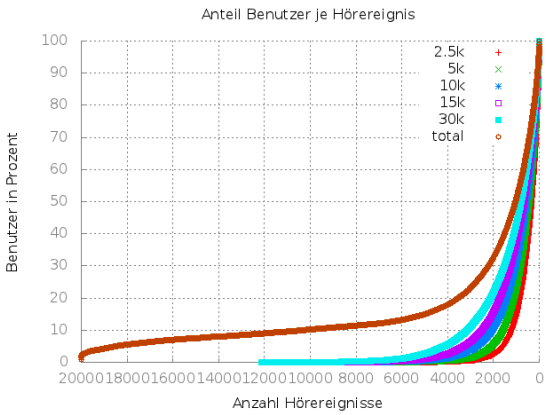


Abbildung 5.12.: Anteil der Benutzer je Anzahl Hörereignisse

5.4. Zusammenfassung

In diesem Kapitel wurde der gecrawlte Datensatz vorgestellt. Insgesamt wurden von ca. 4.400 Benutzern über 4,8 Millionen Titel gespeichert. Durch die Methode, mit welcher Last.fm Hörereignisse von Benutzer speichert, kam es dazu, dass mehrere Schreibweisen des gleichen Titels in den Datensätzen gespeichert wurden. Aufgrund der Elimination dieser Dubletten sank die Anzahl der Titel auf ca. 3,6 Millionen Titel.

Da in dieser Arbeit Empfehlungen durch ein Einzelplatzsystem berechnet wurden, konnten nicht alle 3,6 Millionen Titel für die Evaluation verwendet werden. Stattdessen wurde ein Auswahlverfahren entwickelt, welches versuchte, aus der Gesamtmenge der Titel eine Untermenge zu bilden, welche die gleichen Charakteristika erfüllte. Es wurden vier unterschiedliche Datensätze mit den Größen von 2.500, 5.000, 10.000 und 15.000 Titel generiert, um dieses Auswahlverfahren zu überprüfen. Mit Hilfe der durchgeführten Untersuchungen konnte bestätigt werden, dass die Titelauswahlen vergleichbare Charakteristika hatten. Lediglich bei der Anzahl der Hörereignisse wurde zugunsten des Recommenders entschieden, die Charakteristik nicht beizubehalten. Da der gecrawlte Datensatz viele Titel enthielt, die nur von sehr wenigen Benutzern gehört wurden, waren diese Einträge als Basis für den Recommender unbrauchbar, da zwischen diesen Titeln keine Ähnlichkeiten berechnet werden konnten.

Zur Berechnung von Empfehlungen und der später durchgeführten Evaluation wurde der Datensatz von 15.000 Titeln gewählt. Im folgenden Kapitel wird detailliert beschrieben, wie neue Empfehlungen für einen Benutzer berechnet wurden.

6. Track Recommender System (TRecs)

In dieser Arbeit wurde ein hybrider Recommender basierend auf drei unterschiedlichen Teil-Recommendern entwickelt. Dieser wird im Folgenden als Track Recommender System (TRecs) bezeichnet. Zum einen wurde ein Item-Item Collaborative Recommender benutzt, welcher die Ähnlichkeiten zwischen verschiedenen Titeln berücksichtigte. Zum anderen wurde ein Content-based Recommender implementiert, welcher anhand der Tags eine Ähnlichkeit der Titel berechnen und hierdurch Empfehlungen für einen Benutzer generieren konnte. Zuletzt wurde ein Content-based Recommender programmiert, der in Abhängigkeit des zeitlichen Hörverhaltens Empfehlungen berechnete. Der Zusammenhang zwischen den genutzten Recommendern ist in Abb. 6.1 dargestellt.

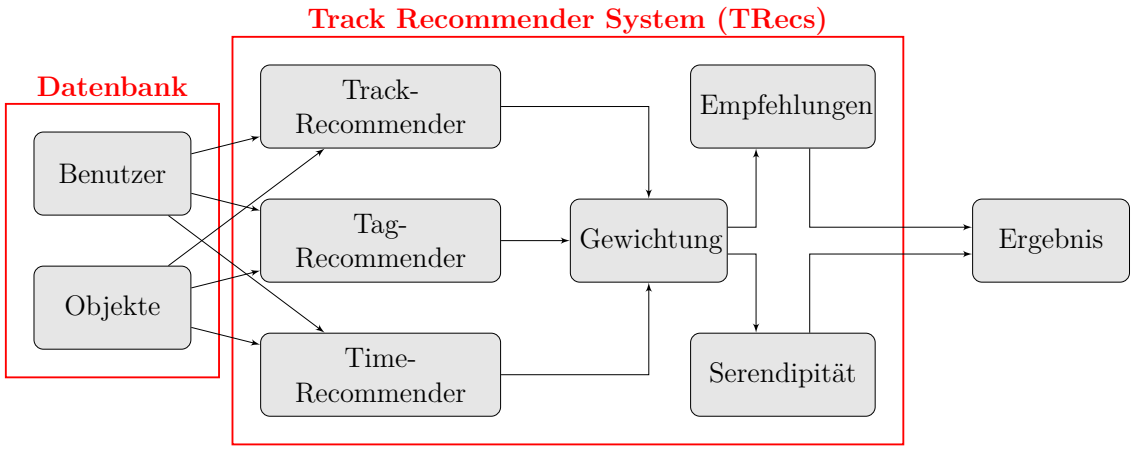


Abbildung 6.1.: Aufbau von TRecs

Die im vorangegangenen Kapitel vorgestellten Daten wurden genutzt, um Ähnlichkeiten zwischen Titeln zu berechnen. Auf diese Ähnlichkeiten wurde zurückgegriffen, um neue Empfehlungen für Benutzer zu berechnen. In diesem Kapitel werden zu-

erst die verwendeten Teil-Recommender beschrieben, welche genutzt wurden, um die Empfehlungen für Benutzer zu berechnen. Es wird dargestellt, wie mit Hilfe einer Vorberechnung die nachfolgende Berechnung der Empfehlungen beschleunigt werden konnte.

Es wurde ein zusätzliches Verfahren implementiert, um eine Überspezialisierung zu vermeiden. Ziel dieser Methode war es, einen sogenannten Serendipitätseffekt zu ermöglichen. Darunter versteht man beispielsweise die Situation, dass ein Benutzer beim Surfen im Internet unbeabsichtigt eine nützliche Internetseite entdeckt. Mit Hilfe dieser Methode wurde versucht, dem Benutzer weitere Titel vorzuschlagen, welche nicht in direkter Verbindung zu ihm stehen und dem Benutzer dennoch gefallen könnten.

6.1. Genutzte Metriken

Zur Berechnung neuer Empfehlungen nutzte TRecs drei unterschiedliche Teil-Recommender. Hierbei unterschieden sich die drei Recommender in ihrem Ansatz. Die einzelnen Recommender werden im Folgenden näher beschrieben.

6.1.1. Track-Recommender

Bei dem Track-Recommender handelte es sich um einen Item-Item Collaborative Recommender. Es wurden dabei, wie in Kap. 3.2.2 beschrieben, Hörereignisse unterschiedlicher Objekte beziehungsweise Titel verglichen.

Es wurde die Pearson-Korrelation (Formel 6.1) verwendet, um die Ähnlichkeiten zu berechnen. Des Weiteren wurde die Funktion $c(u,t)$ definiert, welche angab, wie oft Titel t von Benutzer u gehört wurde. Die durchschnittliche Anzahl der Hörereignisse eines Benutzers u wurde als $\bar{c}(u)$ definiert. Mit Hilfe der Pearson-Korrelation konnte vermieden werden, dass sich die Vorlieben eines Benutzers zu sehr im Ergebnis widerspiegelten.

$$tracksim(t_1,t_2) = \frac{\sum_u (c(u,t_1)-\bar{c}(u))\cdot(c(u,t_2)-\bar{c}(u))}{\sqrt{\sum_u (c(u,t_1)-\bar{c}(u))^2}\cdot\sqrt{\sum_u (c(u,t_2)-\bar{c}(u))^2}} \qquad (6.1)$$
$$u \in Users$$

Die Bewertungsmatrix wurde somit bei diesem Recommender durch eine Matrix ersetzt, in welcher die Anzahl der Hörereignisse pro Benutzer pro Titel gespeichert war (Tab. 6.1). So gab $c(u_2, t_2)$ die Anzahl, an wie oft Benutzer u_2 den Titel t_2 gehört hat. Durch den Einsatz des Track-Recommenders konnten so Ähnlichkeiten zwischen den gespeicherten Titeln berechnet werden.

	t_1	t_2	t_3
u_1	0	2	3
u_2	3	4	6
u_3	1	2	0
u_4	8	0	2

Tabelle 6.1.: Beispiel einer Hörmatrix von Benutzern und Titeln

6.1.2. Tag-Recommender

Da die Tags ein inhaltsbezogenes Merkmal der Titel waren, kann man bei dem Tag-Recommender von einem Content-based Recommender sprechen. Auch dieser Recommender verwendete eine Matrix. In dieser waren Titel und Tags aufgetragen. So wurde $r(t, a)$ als Funktion definiert, welche zu einem Titel t die Wertung des Tags a ausgab. Formel 6.2 zeigt die Kosinus-Ähnlichkeit, welche dazu verwendet wurde die Ähnlichkeit zweier Titel mit Hilfe der Tags zu berechnen. Da bei dem Tag-Recommender keine subjektiven Bewertungskriterien durch Benutzer berücksichtigt werden mussten, reichte es aus die Kosinus-Ähnlichkeit anstelle der Pearson-Relation zu verwenden. Die Wertung eines Tags zu einem Titel wurde wie in Kapitel 2 beschrieben durch die Last.fm-API bereitgestellt. In Tab. 6.2 ist ein Beispiel einer Tag-Titel-Matrix dargestellt, wie sie auch zur Berechnung einer Ähnlichkeit verwendet wurde.

$$tagsim(t_1, t_2) = \frac{\sum_a r(t_1, a) \cdot r(t_2, a)}{\sqrt{\sum_a r(t_1, a)^2} \cdot \sqrt{\sum_a r(t_2, a)^2}} \qquad a \in Tags \tag{6.2}$$

	t_1	t_2	t_3	t_4
a_1	100	40	0	0
a_2	25	15	60	40
a_3	53	80	0	0
a_4	0	0	70	12

Tabelle 6.2.: Beispiel einer Tag-Titel-Matrix mit Wertungen

6.1.3. Time-Recommender

Ziel des Time-Recommenders war es, eine hohe Ähnlichkeit zwischen Titeln zu finden, welche an ähnlichen Zeiten gehört wurden. Ein klassisches Weihnachtslied sollte hiermit eine hohe Ähnlichkeit mit anderen Weihnachtsliedern haben. Auch war es möglich, besondere Ereignisse wie zum Beispiel den 14. Februar (Valentinstag) zu berücksichtigen. Sollte ein Benutzer eine Vorliebe für Weihnachtslieder haben, wurden ihm durch diesen Recommender weitere Weihnachtslieder empfohlen, da diese an gleichen Tagen oder Zeiträumen eine Erhöhung der Hörereignisse hatten.

Die Ähnlichkeit zwischen zwei Titeln wurde durch die Formel 6.3 berechnet. Hierbei wurde $c(t, d)$ als die Höranzahl von Titel t an dem Tag d definiert. Der Time-Recommender wurde aus den Hörereignissen der Benutzer gebildet. Diese wurden jedoch nicht über die Benutzer gebildet. Stattdessen wurden die Hörereignisse eines Titels unabhängig des Benutzers eines Tages addiert. So entstand für jeden Titel ein Vektor mit 366 Elementen, welche die 366 Tage des Jahres darstellten (Tab. 6.3). Dieser Vektor konnte, ähnlich wie beim Tag-Recommender, dazu verwendet werden, eine Ähnlichkeit zwischen zwei unterschiedlichen Titeln zu errechnen.

$$timesim(t_1, t_2) = \frac{\sum_{d=1}^{366} c(t_1, d) \cdot c(t_2, d)}{\sqrt{\sum_{d=1}^{366} c(t_1, d)^2} \cdot \sqrt{\sum_{d=1}^{366} c(t_2, d)^2}} \tag{6.3}$$

In dem Beispiel in Tab. 6.3 würde der Titel t_1 einen Titel repräsentieren, der insbesondere an Silvester eine erhöhte Anzahl an Hörereignissen aufweist. Titel t_2, t_3 und t_4 zeigen Titel, die über das ganze Jahr relativ häufig gehört werden. Jedoch kann man an den Zahlenwerten erkennen, dass es sich bei Titel t_3 um einen Titel handelt, der deutlich öfter als Titel t_2 oder t_4 gehört wurde.

	t_1	t_2	t_3	t_4
1	80	3	50	5
2	25	15	60	6
3	2	20	64	8
4	3	10	58	12
...				
365	180	15	67	2
366	250	8	70	9

Tabelle 6.3.: Beispiel einer Zeit-Titel-Matrix mit Wertungen

Ein Beispiel von zwei unterschiedlichen Titeln wird in Abb. 6.2 dargestellt. Zu sehen sind die Hörereignisse zweier Titel über das Jahr verteilt. Bei den beiden Titeln handelt es sich um *All I Want For Christmas Is You* von *Mariah Carey* und *Let it Snow! Let It Snow! Let It Snow!* von *Dean Martin*. Obgleich der Titel von *Mariah Carey* öfter gehört wurde, ist zu sehen, dass beide Titel meist in der Weihnachtszeit gehört wurden (Diagrammbereich 330 bis 360). Hieraus entsteht eine hohe Ähnlichkeit beider Titel. Ebenso deutlich wird die Ähnlichkeit zweier Titel bei Abb. 6.3. Hier sind die Hörereignisse zweier Titel des Interpreten *Coldplay* dargestellt. Beide Titel befinden sich auf dem selben Album, welches erst im November 2011 erschienen ist. Beide Titel haben eine sehr ähnliche Hörhäufigkeit an gleichen Tagen, wodurch sich eine Ähnlichkeit beider Titel von 0,974 ergibt.

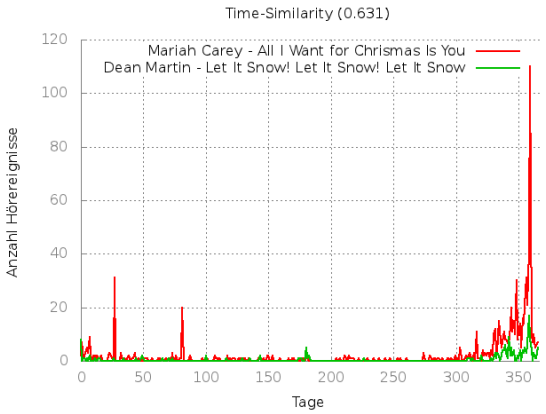


Abbildung 6.2.: Time-Similarity zweier Weihnachtstitel

Nachdem die verschiedenen Recommender vorgestellt wurden, wird im Folgenden auf die Vorberechnung eingegangen. Diese wurde genutzt, um die Berechnungszeit der Empfehlungen zu senken und dem Benutzer ein schnelleres Feedback zu geben.

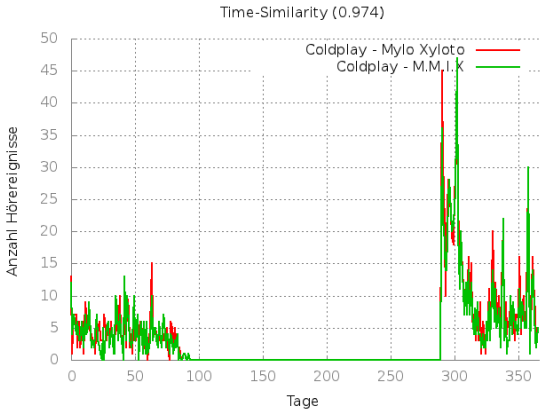


Abbildung 6.3.: Time-Similarity zweier Titel des selben Albums

6.2. Vorberechnungen

Wie bereits in Kapitel 3 erwähnt, lässt sich der Recommender durch gewisse Vorberechnungen signifikant beschleunigen. In diesem Kapitel wird die hierfür eingesetzte Strategie beschrieben.

Für jeden Recommender mussten in der Vorberechnung alle Ähnlichkeiten zwischen den Titeln berechnet werden. Eine Ähnlichkeit zweier Titel kann mit $f(t_1, t_2)$ ausgedrückt werden. Da die Matrix aus Titel und Titel symmetrisch ist, gilt $f(t_1, t_2) = f(t_2, t_1)$. Die Anzahl der benötigten Ähnlichkeitsberechnungen bei n Titeln kann somit auf $n^2 \cdot 0,5$ begrenzt werden. So ergibt sich eine Rechenaufwand für die drei unterschiedlichen Recommender von $1,5 \cdot n^2$.

Es wurden die Vorberechnungen auf den in Kapitel 5 vorgestellten Titelauswahlen durchgeführt, um die Laufzeit der Vorberechnung zu überprüfen. In Abb. 6.4 ist die benötigte Zeit der unterschiedlichen Datensatzgröße aufgetragen. Hierdurch ist zu erkennen, dass die benötigte Zeit, um die Vorberechnungen durchzuführen, polynomiell wächst. Werden bei der Datensatzgröße von 5.000 Titeln noch 42 Minuten benötigt, vervielfacht sich die benötigte Zeit bei 15.000 Titeln auf über 5 Stunden. Verdoppelt sich die Anzahl der Titel, vervierfacht sich die Anzahl der durchzuführenden Berechnungen. Im Durchschnitt benötigte demnach eine Berechnung der Ähnlichkeit zwischen zwei Titeln ca. 67 Mikrosekunden. Bei einer Titelmenge von 100.000 Titeln würde somit die Vorberechnung 11,6 Tage benötigen. Wenn für die gesamte gecrawlte Titelmenge von 3.800.000 Titeln eine Vorberechnung ausgeführt werden sollte, würde dies mit der verwendeten Hardware knapp 46 Jahre dauern.

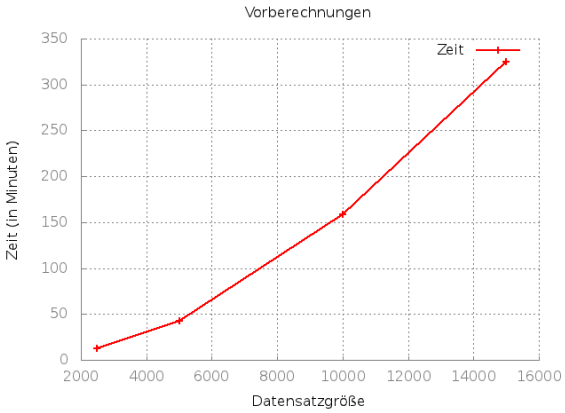


Abbildung 6.4.: Benötigte Zeit für die Vorberechnungen

Ähnlich wie bei der benötigten Zeit verhielt es sich auch mit den zwischengespeicherten Ähnlichkeiten, welche in Abb. 6.5 zu sehen sind. Sobald die Anzahl der Titel wächst, wächst auch die Anzahl der zwischengespeicherten Ähnlichkeiten der Titel. Hierbei wurde die Erkenntnis von Herlocker et al. [13] verwendet, nach welcher negative Ähnlichkeiten zu keiner Verbesserungen der Empfehlungen führten. In der Datenbank wurden hierbei nur Ähnlichkeiten gespeichert, welche höher als 0 waren. Die hohe Anzahl der berechneten Time-Similarity zwischen den Titeln ist durch die Hörereignisse an den verschiedenen Tagen zu erklären. Viele Titel wurden nur sehr selten gehört, dadurch wurde eine Ähnlichkeit berechnet, welche über der Grenze von 0 lag.

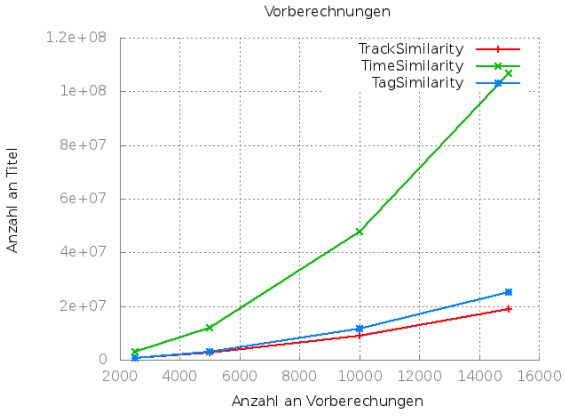


Abbildung 6.5.: Größe der zwischengespeicherten Ähnlichkeiten

Da die Vorberechnungen abhängig von der Titelmenge sehr lang dauern können, wurde die Berechnung bereits zuvor durchgeführt. Im späteren Live-Einsatz können diese

Berechnungen jedoch nicht nochmal auf die selbe Weise berechnet werden. Dennoch sollten die Ähnlichkeiten großer Titelmengen regelmäßig aktualisiert werden. Hierbei dürfen nicht die bereits vorberechneten Ähnlichkeiten verworfen werden, an Stelle dessen werden die Vorberechnungen an einem anderen Ort gespeichert. Nachdem die neuen Vorberechnungen durchgeführt wurden, kann die Tabelle mit den genutzten Ähnlichkeiten durch die neu berechneten Ähnlichkeiten ausgetauscht werden. Eine Aktualisierung der bereits berechneten Ähnlichkeiten ist prinzipiell möglich, wurde in dieser Arbeit jedoch nicht implementiert, da die Anzahl der Bewertungen neuer Benutzer zu gering eingeschätzt wurde, sodass nur minimale Änderungen in den berechneten Ähnlichkeiten erwartet wurden.

Da zu einer Berechnung eines Prediction-Wertes eines Titels alle Ähnlichkeiten des Titels mit einbezogen werden, wurde, wie in Kapitel 6 beschrieben, die Anzahl der zur Berechnung verwendenden Titel beschränkt. Hierzu wurden nach der Beendigung der Vorberechnungen je Titel die 150 ähnlichsten Titel in einer neuen Tabelle gespeichert. Nachdem dies mit allen vorberechneten Werten geschehen war, gab es je Titel maximal 450 Ähnlichkeiten, welche für die spätere Berechnung zur Verfügung standen. Wie in Tab. 6.4 veranschaulicht, wurden die IDs der zwei unterschiedlichen Titel und die Ähnlichkeiten (Similarity) der verschiedenen Recommender gespeichert.

Titel1	**Titel2**	**tracksim**	**timesim**	**tagsim**
Come Undone	Kings of Medicine	0.7399	0.8418	0.9085
Breathe Underwater	Kings of Medicine	0.7413	0.8154	0.8907
Devil In The Details	Kings of Medicine	0.7177	0.8069	0.8941

Tabelle 6.4.: Auszug der Ähnlichkeiten von Titel *Kings of Medicine* von *Placebo*

6.3. Berechnungen

Es mussten folgende Parameter angegeben werden, um Empfehlungen neuer Titel für einen Benutzer zu berechnen: Um die Hörhistorie und die Liste der gebannten und favorisierten Titel eines Benutzers und seine Vorlieben zu berücksichtigen, musste angegeben werden, um welchen Benutzer es sich handelte. Dies wurde mit Hilfe einer ID gewährleistet, welche einmalig für jeden Benutzer festgelegt wurde. Unter Berücksichtigung einer Gewichtung der verwendeten Teil-Recommender konnten die

berechneten Ähnlichkeiten der Titel unterschiedlich berücksichtigt werden. Durch die Angabe der gewünschten Menge an Empfehlungen und Serendipitätseinträgen konnte festgelegt werden, wie viele Titel in einer Liste angezeigt werden sollten.

Da der hybride Recommender in einem Bereich angewendet wurde, in dem einzelne Titel nur eine relativ kurze Spieldauer von drei bis vier Minuten haben, sollten neue Empfehlungen innerhalb dieser Zeit berechnet werden. Hierbei handelte es sich um die maximale Berechnungszeit. Sobald die Berechnungen länger als die angegebene Zeit brauchten, wurden alle noch laufenden Berechnungen gestoppt. Anschließend wurde mit den bereits berechneten Titeln fortgefahren. Dies konnte zur Folge haben, dass nicht zu allen Titeln ein Prediction-Wert berechnet wurde.

Nach Angabe dieser Parameter, wurden alle Hörereignisse des Benutzers geladen. Zusätzlich wurden auch die ignorierten, die gebannten und die favorisierten Titel geladen, sodass das komplett gespeicherte Hörprofil, mit all den Vorlieben eines Benutzers geladen wurde.

Anschließend wurde ein sogenannter Prediction-Wert berechnet. Dieser Wert gibt eine Aussage darüber, wie gut ein Titel t zu dem Benutzer u passt. Diese Funktion wurde als $p(u,t)$ definiert. Zusätzlich wurde die Funktion $r(u,t)$ definiert, welche angibt, wie der Titel t von einem Benutzer u bewertet wurde, um die Vorlieben eines Benutzers zu berücksichtigen.

Der Wertebereich der Funktion $r(u,t)$ kann selbst gewählt werden. Legt man ein Minimum min und ein Maximum max für die Bewertungsskala fest, so liegt der Wertebereich des Prediction-Wertes ebenso im Bereich zwischen min und max. Der Wertebereich kann somit frei gewählt werden.

Der Prediction-Wert wurde wie in Formel 6.4 durch eine Kombination der einzelnen Teil-Recommender und den Bewertungen des Benutzers berechnet. Die Teil-Recommender wurden linear kombiniert (Formel 6.6), sodass die Gewichte der einzelnen Teil-Recommender in der Summe 1 ergaben. Für die Berechnung eines Prediction-Wert wurde für jeden Titel die Ähnlichkeit mit dem zu berechnenden Titel mit der Bewertung des Benutzers multipliziert. Diese Bewertung des Benutzers musste in der oben beschriebenen Skala enthalten sein. Durch diese Berechnungsweise konnte ein Titel einen Prediction-Wert auf der Skala zwischen min und max haben.

$$p(u,t) = \frac{\sum_{j \in Tracks} sim(t,j) \cdot r(u,j)}{\sum_{j \in Tracks} sim(t,j)} \tag{6.4}$$

$$\begin{aligned} sim(t_1,t_2) = {} & (tracksim(t_1,t_2) \cdot tracksimWeight) + \\ & (timesim(t_1,t_2) \cdot timesimWeight) + \\ & (tagsim(t_1,t_2) \cdot tagsimWeight) \end{aligned} \tag{6.5}$$

$$1 = tracksimWeight + timesimWeight + tagsimWeight \tag{6.6}$$

Falls ein Benutzer bisher nur sehr wenige Interpreten gehört hat und große Teile seiner Hörereignisse von einem einzelnen Interpreten stammten, würde der Recommender diesen Interpreten bevorzugen. Diese Bevorzugung wäre zum Beispiel daran zu erkennen, dass die gesamten Empfehlungen aus Titeln eines einzelnen Interpreten bestünden. Zur Vermeidung dieses Verhaltens wurde eine Filterstrategie für die Empfehlungen implementiert, welche zu jedem Interpreten nur zwei Titel auswählte, welche die höchsten Prediction-Werte hatten.

Nachdem die Prediction-Werte der Titel berechnet wurden, wurde die gewünschte Menge an Titeln gewählt. Für die Empfehlungen wurden die Titel mit den höchsten Prediction-Werten ausgewählt.

6.4. Serendipität

In dieser Arbeit wurde eine Methode implementiert, welche Hörverhalten verschiedener Benutzer mit einbezog, um bei den Empfehlungen neuer Titel einen Serendipitätseffekt zu erzielen.

Die implementierte Methode setzte sich aus zwei Schritten zusammen: Sobald ein Benutzer u neue Serendipitätstitel berechnen ließ, wurden die fünf zuletzt gehörten Titel des Benutzer ausgewählt. Daraus entstand die Menge L, welche die letzten fünf Titel beinhaltete. Anschließend wurde zu jedem Titel der Menge L die letzten fünf Benutzer bestimmt, welche den Titel gehört hatten. Diese über die Titel ausgewählten Benutzer wurden in der Menge U' vereinigt. Zuletzt wurden Übereinstimmungen zwischen den in U' enthaltenen Benutzer gesucht. Wie in Abb. 6.6 dargestellt, wurden die gehörten Titel der Benutzer aus U' selektiert. U' enthält in

diesem Beispiel die Benutzer $\{u_A, u_B, u_C, u_D\}$. Ein Titel, welcher von einer hohen Anzahl an Benutzern gehört wurde, wurde Titeln mit wenigen Benutzern aus U' vorgezogen. Bei diesem Verfahren wurden diejenigen Titel beachtet, welche nicht von Benutzer u gehört wurden.

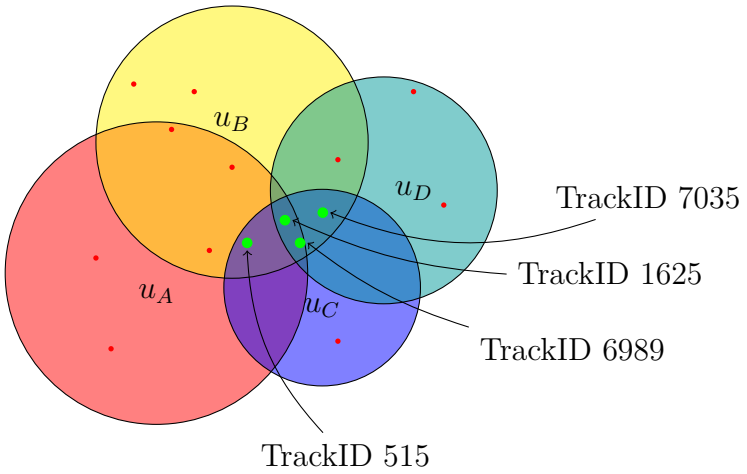


Abbildung 6.6.: Ausschnitt von Schnittmengen des Hörverhaltens der Benutzer u_A, u_B, u_C, u_D

Zusätzlich wurde die Anzahl der Hörereignisse berücksichtigt, welche von Benutzern der Menge U' erstellt wurden. Hierdurch entstand eine Liste an Titeln (siehe Tab. 6.5). Aus dieser berechneten Liste wurden anschließend die Titel mit der höchsten Anzahl an Benutzern ausgewählt. Bei gleicher Anzahl an Benutzern wurde die Anzahl der Hörereignisse der Menge U' berücksichtigt und eine höhere Anzahl bevorzugt.

trackID	**Interpret**	**Titel**	**#Benutzer in U'**	**Hörereignisse**
515	Robbie Williams	Feel	4	35
1625	DJ Bobo	Everybody	4	28
6989	Spice Girls	Wannabe	3	36
7035	Aqua	Barbie Girl	3	12

Tabelle 6.5.: Beispiel einer Serendipitätsmenge

Die beschriebene Methode wurde in TRecs implementiert und lieferte in den ersten Tests gute Ergebnisse bei der Auswahl von Serendipitätseinträgen. Die Einträge standen in einem Zusammenhang zu den vergangenen Hörereignissen, erhielten jedoch von den Teil-Recommender keine hohen Prediction-Werte. Da bei jeder Berechnung der Einträge zu jedem der zuletzt angehörten Titel eine Hörhistorie

erstellt werden musste, stieg der Berechnungsaufwand und damit auch die Berechnungszeit deutlich an. So benötigte die Berechnung neuer Serendipitätseinträge in den Tests mehr Zeit als die Berechnung neuer Empfehlungen. Da der Testrechner durch die Berechnung neuer Empfehlungen ausgelastet war, konnte die Berechnung der Serendipitätseinträge nicht gleichzeitig gestartet werden. Folglich mussten die Empfehlungen und die Serendipitätseinträge nacheinander berechnet werden. Eine Vorberechnung der vorgestellten Methode war nicht möglich, da sich die Hörer der kürzlich gehörten Titel, insbesondere bei einer großen Benutzerzahl, schnell änderte. Infolgedessen wurde eine weitere Methode zur Berechnung von Serendipitätseinträgen implementiert.

Die zweite Methode nutzte die bereits berechneten Prediction-Werte. So konnte darauf verzichtet werden, neue Berechnungen durchzuführen, welche die Berechnungszeit steigen lassen würden. Hier wurde für die Serendipität anstelle eines *top-k* Ansatzes wie bei den Empfehlungen ein *bottom-k* Ansatz gewählt. Berücksichtigt wurden die Titel, welche vom Benutzer noch nicht bewertet wurden und einen minimalen Prediction-Wert hatten. Abhängig von der Bewertunggsskala wurden Titel ignoriert, welche einen sehr geringen Prediction-Wert erhielten. Sehr geringe Prediction-Werte wurden nur dann berechnet, wenn die Vorlieben des Benutzers nicht mit der Hörergruppe des Titels übereinstimmten. Wenn ein Titel keinerlei Übereinstimmung mit den Vorlieben des Benutzers hatte, konnte davon ausgegangen werden, dass ihm der Titel nicht gefallen würde. Bei einem besseren Ergebnis hatte der Titel eine minimale Übereinstimmung mit den Vorlieben des Benutzers.

6.5. Prototyp

TRecs wurde als Server implementiert, der Anfragen über *HTTP* beantworten konnte, um die vorgestellten Strategien zu evaluieren. Zusätzlich wurde eine Internetseite entwickelt, mit welcher der Benutzer interagieren konnte. Durch die Verwendung von Javascript war es möglich, asynchrone Anfragen an TRecs zu senden. Dies bedeutet, dass nach einem Klick auf die Oberfläche die Seite nicht erneut geladen werden musste, sondern Teile der Oberfläche ersetzt werden konnten. Dies beschleunigte die Interaktion mit dem Benutzer, da man nach eine Aktion des Benutzers zum Beispiel die Oberfläche manipulieren konnte. Diese Technik wird ***A****synchronous* ***J****avaScript* ***a****nd* ***X****ML* (AJAX) genannt [19].

In Abb. 6.7 ist der Aufbau des Prototyps schematisch dargestellt. Der Benutzer konnte mit einer Benutzeroberfläche interagieren, welche vom Webserver bereitgestellt wurde. Nachdem der Benutzer sich am System angemeldet hatte, wurde ihm über den Webserver eine neue Oberfläche angezeigt. Nachdem die neue Oberfläche geladen wurde, wurden keine weiteren Anfragen an den Webserver gesendet. Alle weiteren Interaktionen des Benutzers werden mittels Javascript und AJAX an TRecs gesendet, der diese Interaktionen speicherte und Nachrichten an die Benutzeroberfläche zurück lieferte. TRecs wurde dazu verwendet, neue Empfehlungen für einen Benutzer zu berechnen und Bewertungen von Benutzern in der Datenbank zu speichern. *YouTubeConnect* wurde im Rahmen dieser Arbeit entwickelt, um im Vorfeld ein Video der Internetseite YouTube[1] zu finden und dieses mit dem zugehörigen Titel zu verknüpfen.

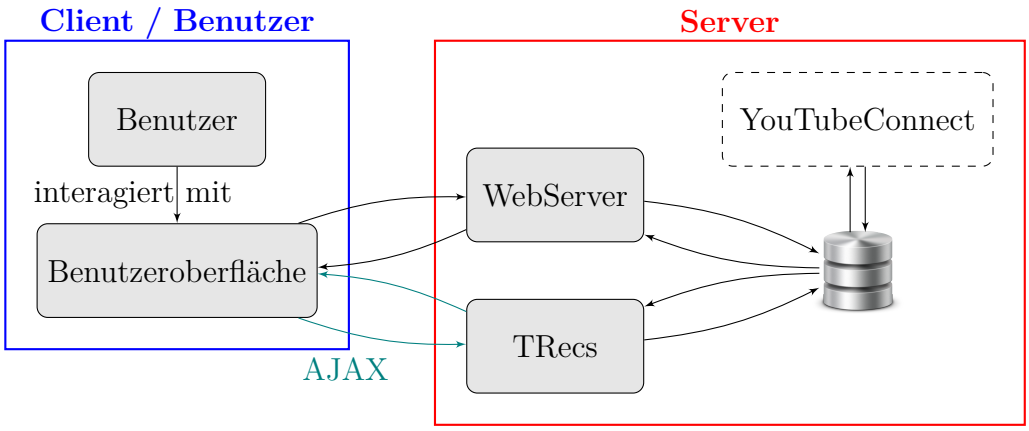


Abbildung 6.7.: Aufbau des Prototyps

Benutzer mussten sich registrieren, um sich Empfehlungen berechnen zu lassen (siehe Abb. 6.8). Dies wurde benötigt, um eine Aktion eindeutig einem Benutzer zuzuordnen und diesem auch Empfehlungen zu präsentieren, die auf seinen Vorlieben basierten. Hierbei wurde auch das Alter und das Geschlecht der Benutzer erhoben, welches für die Evaluation gespeichert wurde. Die Registrierung und Anmeldung an der Weboberfläche wurde mit Hilfe der Email des Benutzers durchgeführt.

Nach der ersten Anmeldung des Benutzers wurde er zur Einführung weitergeleitet, welche ihm die Möglichkeiten der Oberfläche erklärte. Diese Einführung ist in Anhang B dargestellt.

[1] http://www.youtube.com/ - Zugriff: 01.09.2012

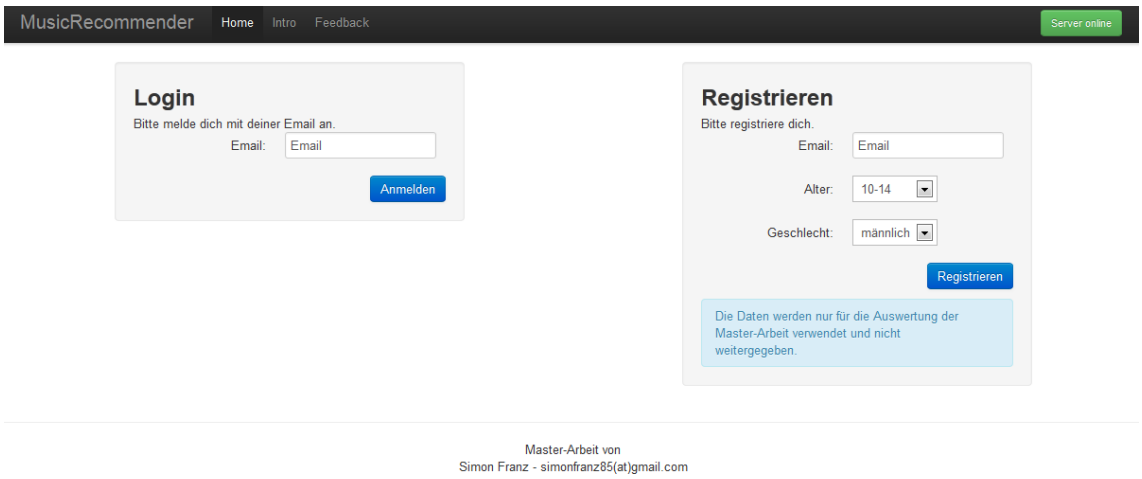


Abbildung 6.8.: Anmeldung und Registrierung der Benutzeroberfläche

Der Benutzer hatte die Möglichkeit, Titel anhand des Interpreten und des Titels zu seiner Hörhistorie hinzuzufügen (Abb. 6.9). Des Weiteren konnte er sich zufällige Titel anzeigen lassen, die er anschließend seinem Hörprofil hinzufügen konnte.

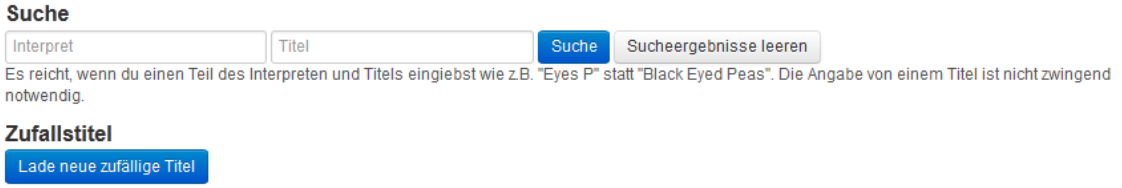


Abbildung 6.9.: TRecs Suchfunktion

Der Benutzer hatte unterschiedliche Möglichkeiten, einen Titel zu bewerten. Sobald er einen Titel bewertet hatte, wurde eine Anfrage an TRecs gesendet, welcher die Aktion in der Datenbank speicherte.

YoutubeConnect

Da davon ausgegangen werden konnte, dass viele Benutzer nicht alle 15.000 Titel kennen würden, wurde versucht, den Benutzer bei der Entscheidung, ob ihm ein Titel gefällt oder nicht, zu unterstützen. Hierzu wurde mit *YouTubeConnect* versucht, zu jedem Titel ein YouTube-Video zu finden. Mit Hilfe der YouTube API[2] konnte jeder Titel mit Interpreten bei YouTube gesucht werden. Zu jeder Suchanfrage wurde eine Ergebnisliste durch die YouTube-API zurückgegeben. Da YouTube-Videos meist von normalen Benutzern hochgeladen werden, gibt es kein Standard-Namensschema

[2]http://gdata.youtube.com/ - Zugriff: 01.09.2012

des Videonamens. Somit wurde jedes Video im Suchergebnis auf eine Ähnlichkeit zwischen *Interpret - Titel* und *Titel - Interpret* überprüft. Des Weiteren wurde die oftmals genutzte Angabe *(Official Video)* aus dem Videonamen entfernt. Die Ähnlichkeit des Videonamens und des Titels wurde mit Hilfe der Levenshtein-Distanz berechnet. Die Jaro-Winkler Distanz ist war diesen Zweck ungeeignet, da sie insbesondere die Ähnlichkeit am Anfang der Zeichenkette bevorzugt. Da bei den Videos jedoch häufig Schreibfehler am Anfang eines Videonamens auftreten, wurde hier die Levenshtein-Distanz eingesetzt. Bei der Levenshtein-Distanz wird die Anzahl an Einfüge-, Lösch- und Ersetz-Operationen gezählt, die benötigt wird, um eine Zeichenkette in eine zweite umzuwandeln.

Da in Deutschland sehr viele Videos gesperrt sind, musste anschließend überprüft werden, ob die übereinstimmenden Videos in der Ergebnisliste für deutsche Besucher gesperrt waren. Dies konnte automatisiert durchgeführt werden, indem durch den Testrechner die YouTube-Seite des Videos aufgerufen und überprüft wurde, ob eine Sperr-Meldung angezeigt wurde. Konnte sowohl eine Übereinstimmung mit dem Videonamen festgestellt als auch die Verfügbarkeit in Deutschland positiv überprüft werden, wurde das erste passende Video ausgewählt und mit dem Titel verknüpft.

Von den 15.000 Titeln konnten bei ca. 10.600 Titeln ein Video gefunden werden, welches bei YouTube verfügbar war. Dies bedeutet, dass zu ca. 71% der Titel ein YouTube-Video gefunden wurde. Falls zu einem angezeigten Titel kein YouTube-Link verfügbar war, wurde ein Button angezeigt, mit welchem der Benutzer auf das Suchergebnis des Titels und Interpreten von YouTube geleitet wurde. Falls ein YouTube-Video vorhanden war, konnte der Benutzer entweder das Video in einem neuen Fenster öffnen oder das Video mittels eines Play-Buttons direkt auf der Seite wiedergeben. In Abb. 6.10 ist in der Link-Spalte zu erkennen, dass zu dem fünften Titel kein YouTube-Video gefunden und dementsprechend eine YouTube-Suche angeboten wurde. Bei den ersten vier Titeln wurde die Möglichkeiten angeboten, das Video in einem neuen Fenster oder auf der aktuellen Seite abzuspielen.

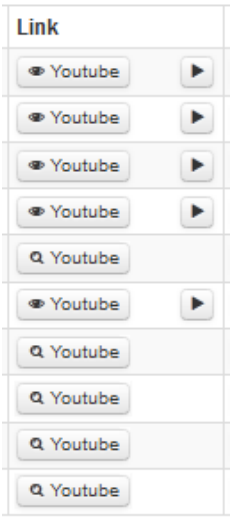


Abbildung 6.10.: Eingeblendete Links zu YouTube-Videos

Wenn der Benutzer die Möglichkeit nutzte, Videos direkt auf der Seite anzusehen, wurde die YouTubeID des Titels ausgelesen und die URL des einzubindenden Videos erstellt. Anschließend wurde ein Menüeintrag angelegt, welcher in eine Playlist hinzugefügt wurde (Abb. 6.11). Befanden sich keine Titel in der Playlist oder wurde gerade kein Video wiedergegeben, startete die Wiedergabe des aktuellen Videos sofort. Anderenfalls wurde es in die Playlist eingereiht und automatisch, nach allen vorherigen Videos, abgespielt.

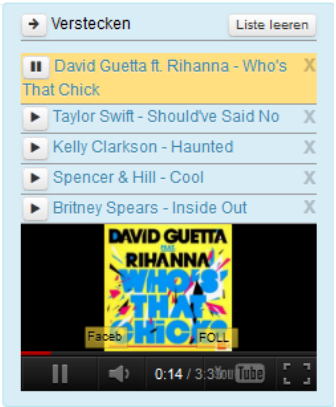


Abbildung 6.11.: Playlist des integrierten Video-Players

TRecs

Nachdem der Benutzer seine Vorlieben durch die Auswahl einiger Titel angegeben hatte, konnte er sich neue Empfehlungen berechnen lassen. Durch Javascript wurde

eine Anfrage an TRecs gesendet, der dadurch die Berechnung der Empfehlungen für den aktuellen Benutzer begann. Falls gerade mehrere Benutzer gleichzeitig angemeldet waren und zeitgleich neue Empfehlungen wollten, wurde die Berechnung für den ersten Benutzer gestartet und dem zweiten Benutzer eine Meldung zurückgegeben, dass seine Empfehlungen als nächstes berechnet werden. Hierdurch gelang es, dem Benutzer eine Rückmeldung zu geben, dass seine Berechnungen noch nicht gestartet waren. Des Weiteren konnten die Berechnungen der Empfehlungen für einen Benutzer beschleunigt werden, da immer nur die Empfehlungen eines Benutzers berechnet werden musste.

Die Berechnung der Prediction-Werte wurde beschleunigt, indem mehrere Threads genutzt wurden. Sobald neue Empfehlungen für einen Benutzer berechnet werden mussten, wurden die Titel in Blöcke zu je 500 Titeln zusammengefasst und einem Thread übergeben, welcher die Prediction-Werte berechnete. Der Wert von 500 Titeln hat den Vorteil, dass bei der Iteration über die Gesamtmenge nicht lange gewartet werden musste bis der erste Thread anfing zu rechnen. Des Weiteren konnte hierdurch mit einer hohen Anzahl an Threads gerechnet werden. In Abb. 6.12 ist die Anzahl der verwendeten Threads in Abhängigkeit zur Anzahl der zu berechneten Prediction-Werte pro Thread dargestellt. Hier ist zu erkennen dass 30 parallel arbeitende Threads zur Berechnung der Prediction-Werte am schnellsten waren. Die benötigte Zeit pro Durchlauf der 15.000 Titel wurde durch sieben Durchläufe gemittelt.

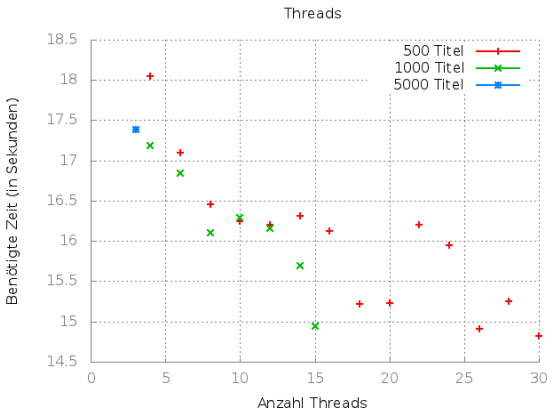


Abbildung 6.12.: Anzahl der Threads im Vergleich zur Geschwindigkeit der Berechnung der Prediction-Werte

Mit einem definierten zeitlichem Maximum konnte gewährleistet werden, dass neue Empfehlungen innerhalb einer bestimmten Zeit nach dem Start der Berechnungen angezeigt wurden. Hierzu wurde ein Countdown implementiert (Abb. 6.13), der dem Benutzer verdeutlichte, dass seine Empfehlungen nach einem bestimmten Maximum angezeigt werden. Wenn innerhalb dieser Zeit nicht alle Prediction-Werte der Titel berechnet werden konnten, wurden die Threads beendet und mit der Menge der bereits berechneten Prediction-Werte weiter gearbeitet.

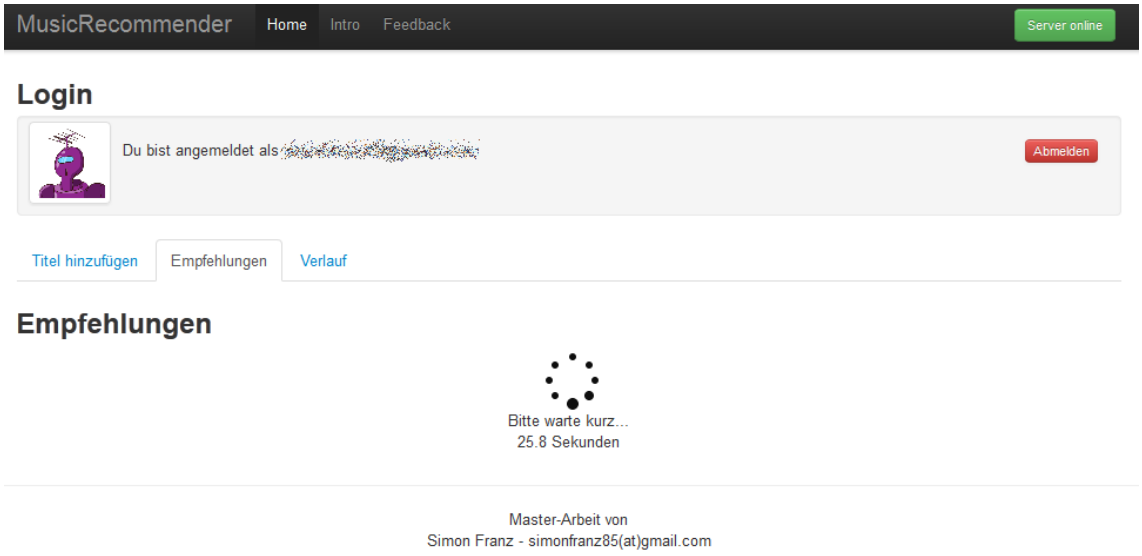


Abbildung 6.13.: Hinweis an einen Benutzer

Benutzeroberfläche

Sobald dem Benutzer neue Empfehlungen auf der Seite angezeigt wurden, konnte dieser, wie bereits bei den manuell hinzugefügten Titeln, die Titel bewerten. Neben den YouTube-Links wurden bei den Empfehlungen weitere Informationen angezeigt. Hierzu gehörte der Prediction-Wert, welcher in der Oberfläche als *Score* bezeichnet wurde. Je nach Übereinstimmung wurde der Titel mit einer unterschiedlichen Anzahl an Sternen versehen. Da die Titel absteigend nach ihrem Prediction-Wert sortiert wurden, wurden die besten Empfehlungen am Listenanfang angezeigt. Indem der Benutzer mit der Maus über die Sterne fuhr, konnte er auch den Prediction-Wert als Fließkommazahl einsehen (Abb. 6.14). Der berechnete Prediction-Wert konnte mit Hilfe einer Skala visualisiert (Tab. 7.2) werden, wie in Kapitel 7 näher beschrieben wird.

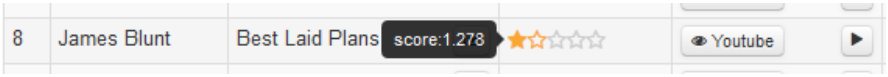


Abbildung 6.14.: Angezeigte Informationen einer Empfehlungsliste

Zusätzlich zu den angezeigten Scores der einzelnen Titel hatte der Benutzer die Möglichkeit, eine Erklärung zu öffnen, in welcher angezeigt wurde, wieso der Titel empfohlen wurde. So wurden in einem Erklärungsfenster alle für die diese Empfehlung relevanten Titel angezeigt, welche der Benutzer bereits gehört hatte (Abb. 6.15). Es wurden in der Erklärung nicht nur die Ähnlichkeiten zwischen dem aktuellen Titel und den gehörten Titeln dargestellt, sondern beim Überfahren der Ähnlichkeit mit dem Mauszeiger auch Informationen dazu gegeben, aus welchen Einzelwertungen der Teil-Recommender sich der Wert zusammensetzte. In Abb. 6.15 ist exemplarisch dargestellt, aufgrund welcher vom Benutzer bewerteten Titel ein Titel empfohlen wurde.

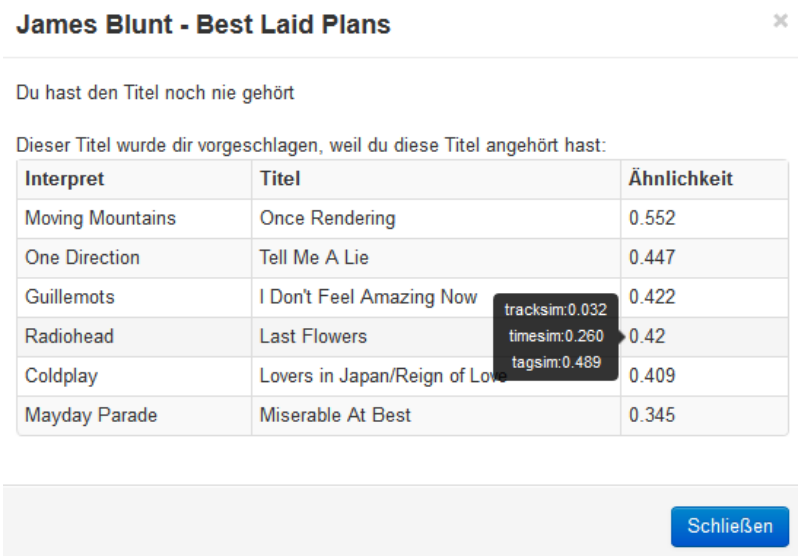


Abbildung 6.15.: Angezeigte Informationen einer Empfehlungsliste

Die Serendipitätseinträge wurden unter die Empfehlungen gemischt und an einer zufälligen Stelle in der Liste positioniert. Da die Serendipitätstitel keine Empfehlung darstellten und somit der Prediction-Wert nicht angezeigt wurde, wurden diese Titel mit einer Markierung *Neuheit* (Abb. 6.16) gekennzeichnet.

Der Benutzer hatte die Möglichkeit seinen Verlauf zu laden, um die zuletzt bewerteten Titel anzusehen (Abb. 6.17). Dies diente dazu, dem Benutzer eine Übersicht

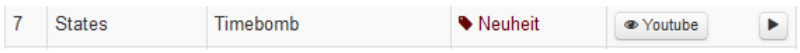


Abbildung 6.16.: Kennzeichnung eines Serendipitätseintrags

über die zuletzt bewerteten Titel zu geben und konnte insbesondere beim Eintragen der ersten Titel eine Hilfe darstellen.

Nr.	Interpret	Titel	Zeit	Gespeichert als
1	Anneke Van Giersbergen & Agua de Annique	Wonder	14. Mai 2012 21:00	Liebe ich
2	Regina Spektor	Laughing With	14. Mai 2012 20:59	Liebe ich
3	Goldmund	All Four Holy Mountains	14. Mai 2012 20:59	Höre ich
4	Moving Mountains	Once Rendering	14. Mai 2012 20:58	Höre ich
5	Guillemots	I Don't Feel Amazing Now	14. Mai 2012 20:58	Höre ich
6	Coldplay	Hurts Like Heaven	14. Mai 2012 20:58	Höre ich
7	IAMX	Tear Garden	14. Mai 2012 20:58	Höre ich
8	Coldplay	Lovers in Japan/Reign of Love	14. Mai 2012 20:57	Höre ich
9	R.E.M.	Radio Free Europe	14. Mai 2012 20:57	Liebe ich
10	U2	Sweetest Thing /The	14. Mai 2012 20:57	Liebe ich

Abbildung 6.17.: Ausschnitt der zuletzt bewerteten Titel

Mit Hilfe der Weboberfläche wurden die Anmeldezeiten der Benutzer protokolliert. Sobald sich ein Benutzer über die Weboberfläche angemeldet hatte, wurde durch Javascript jede Minute eine Nachricht gesendet, welche den derzeitigen Benutzer als aktiven Benutzer kennzeichnete. Hierdurch konnte zum einen die später durchgeführte Evaluation überwacht als auch Auswertungen über die Verweildauer jedes Benutzers gemacht werden.

Für die spätere Evaluation, welche in Kapitel 7 beschrieben ist, wurde des weiteren ein Formular implementiert, über welches Hinweise, Wünsche oder Anregungen gesendet werden konnten. Die Möglichkeit des Feedbacks stand nicht nur den registrierten Benutzern, sondern auch nicht registrierten Benutzern zur Verfügung, um etwaige Probleme über das Feedbackformular mitteilen zu können. Das gesammelte Feedback ist in Anhang C einzusehen und wird im nächsten Kapitel diskutiert.

6.6. Zusammenfassung

In diesem Kapitel wurde TRecs vorgestellt. Es wurde auf die Eigenschaften der jeweiligen Teil-Recommender eingegangen und beschrieben, wie die Ähnlichkeiten

zweier Titel berechnet wurden. Es wurde eine Vorberechnung implementiert, um die benötigte Berechnungszeit während der Laufzeit von TRecs zu minimieren. Diese berechnete für jeden Recommender die Ähnlichkeiten zwischen den unterschiedlichen Titeln.

Neue Empfehlungen wurden mit Hilfe mehrerer Threads berechnet, durch welche die Berechnungszeit reduziert werden konnte. Zusätzlich wurde eine maximale Berechnungsdauer implementiert, damit die Berechnungen neuer Empfehlungen nicht länger als definiert benötigten.

Ebenso wurde auf die Berechnung der Serendipitätseinträge eingegangen. Diese Titel dürfen nicht als Empfehlung gesehen werden. Vielmehr sollten sie als glücklicher Zufall angesehen werden. Obgleich der Benutzer seine Vorlieben in der Musik hat, wurde versucht, mit Hilfe dieser Serendipitätseinträgen seinen Musikgeschmack mit anderen Titeln zu erweitern, die ihm ebenfalls gefallen könnten.

Zuletzt wurde der entwickelte Prototyp, der in Kapitel 7 evaluiert wird, vorgestellt und die Interaktionsmöglichkeiten der Benutzer beschrieben.

7. Evaluation

Im folgenden Kapitel wird die Evaluation des im Kapitel 6 vorgetellten TRecs vorgestellt. Es wurden vier unterschiedliche Gruppen gebildet, welche im Folgenden näher beschrieben werden, um TRecs zu testen und verschiedene Einstellungen miteinander zu vergleichen. Insgesamt registrierten sich über 140 Personen, um an der Evaluation teilzunehmen.

7.1. Versuchsdesign

Die Evaluation wurde mit vier unterschiedlichen Gruppen durchgeführt. Die Gruppen unterschieden sich in der Gewichtung der Teil-Recommender. Ziel dieser Gruppeneinteilung war, herauszufinden, welche Einstellungen von TRecs die besten Empfehlungen generieren konnte. Bei den ersten drei Gruppen wurden die Empfehlungen basierend auf einer starken Gewichtung auf jeweils einem Teil-Recommender berechnet. Bei Gruppe 4 wurden alle drei Teil-Recommender gleichmäßig gewichtet. Die Gewichtung der einzelnen Teil-Recommender der Gruppen ist in Tab. 7.1 dargestellt.

	Track-Recommender	**Time-Recommender**	**Tag-Recommender**
Gruppe 1	0.8	0.1	0.1
Gruppe 2	0.1	0.8	0.1
Gruppe 3	0.1	0.1	0.8
Gruppe 4	0.34	0.33	0.33

Tabelle 7.1.: Gewichtung der Recommender pro Gruppe

Damit TRecs neue Empfehlungen eines Teilnehmers berechnen konnte, musste Vorwissen über die Vorlieben eines Benutzers bereits vorhanden sein. In einer ersten Testphase stellte es sich für Personen als schwierig heraus, Titel zu nennen, die ihren Vorlieben entsprachen. Jeder Benutzer musste unabhängig von seinen musikalischen Kenntnissen zehn Titel angeben, die seinen Vorlieben entsprachen. Auf

dieser Grundlage wurden neue Empfehlungen berechnet. Durch die Wahl von nur zehn Titeln wurde die Zeit, welche ein Teilnehmer zwischen der Registrierung und der Berechnung der ersten Empfehlungen benötigte, so gering wie möglich gehalten.

Da die Spieldauer eines Titels nur zwischen drei und vier Minuten liegt, sollte die benötigte Berechnungszeit für neue Empfehlungen so kurz wie möglich gewählt werden. Im besten Fall sollte der Teilnehmer die Möglichkeit haben, einen Titel anzuhören und bereits währenddessen neue Empfehlungen bewerten zu können. In der Evaluation wurde die maximale Berechnungszeit deshalb auf 30 Sekunden gesetzt. Dies schränkte zwar die Berechnungsmöglichkeit ein, sollte dem Teilnehmer jedoch so kurz vorkommen, dass er vor dem Bildschirm auf die Berechnungen warten konnte.

Es wurde die in Kap. 6.4 vorgestellte zweite Methode genutzt, um die Serendipitätseinträge innerhalb der maximal gesetzten Berechnungszeit zu berechnen. Diese hatte den Vorteil, dass zu jedem Titel der bereits berechnete Prediction-Wert genutzt werden konnte und die Serendipitätseinträge hieraus entnommen werden konnten. Die erste Methode stellte hierbei zwar eine gute Methode dar, um zusätzliche Serendipitätseinträge zu berechnen, jedoch wurde in dieser Evaluation eine kürzere Berechnungszeit vor der Güte der Serendipitätseinträge bevorzugt, um die Motivation der Teilnehmer nicht durch lange Berechnungszeiten zu senken.

Zur Berechnung der bereits in Kapitel 6 beschriebenen Prediction-Werte wurde die Skala, wie in Tab. 7.3 abgebildet, genutzt. Dem Benutzer standen hierdurch vier unterschiedliche Bewertungsmöglichkeiten zur Verfügung. Hierbei ist zu beachten, dass es sich um eine Likert-Skala mit drei Einträgen handelte. Die Bewertungsdimension teilte sich in *Liebe ich*, *Höre ich* und *Mag ich nicht*. Die Bewertungsmöglichkeit des Ignorierens wurde genutzt um Titel zu markieren, welche der Teilnehmer im Moment nicht bewerten wollte. Damit stand dem Benutzer eine neutrale Bewertung zur Verfügung. Mit Hilfe dieser Bewertungsmöglichkeit konnte der Benutzer den Titel für die nächsten zehn Empfehlungslisten ausblenden.

Durch die verwendete Skala wurde der Wertebereich für die Prediction-Werte auf -1 bis 3 festgelegt. Die in Tab. 7.2 abgebildete Visualisierungsskala wurde eingeführt, um dem Benutzer eine visuelle Repräsentation des Prediction-Wertes zu liefern. In Abhängigkeit zu dem berechneten Prediction-Wert wurde das passende Symbol angezeigt.

Symbol	Wertebereich
☆☆☆☆☆	Unter 1
★☆☆☆☆	Zwischen 1 und 1,1
★☆☆☆☆	Zwischen 1,1 und 1,2
★★☆☆☆	Zwischen 1,2 und 1,3
★★☆☆☆	Zwischen 1,3 und 1,4
★★★☆☆	Zwischen 1,4 und 1,5
★★★☆☆	Zwischen 1,5 und 1,6
★★★★☆	Zwischen 1,6 und 1,7
★★★★☆	Zwischen 1,7 und 1,8
★★★★☆	Zwischen 1,8 und 1,9
★★★★★	Zwischen 1,9 und 2
★★★★★🔥	Zwischen 2 und 2,5
★★★★★🔥🔥	Zwischen 2,5 und 3

Tabelle 7.2.: Symbole und Wertebereiche des Prediction-Wertes

Für die Berechnung der Prediction-Werte wurde die in Tab. 7.3 dargestellte Skala verwendet.

Wertung	**Beschreibung**	Wertung
Liebe ich	Titel, die vom Benutzer sehr gerne gehört werden	3
Höre ich	Titel, die vom Benutzer gehört werden	2
Mag ich nicht	Titel, die der Benutzer nicht mag	-1
Ignorieren	Titel, welche der Benutzer gerade nicht bewerten will. Diese wurden für die nächsten zehn Empfehlungslisten nicht weiter berücksichtigt.	-

Tabelle 7.3.: Bewertungsmöglichkeiten pro Titel

Die Bewertungsskala aus Tab. 7.3 wurde sowohl bei der Bewertung von Titeln eingesetzt als auch bei der Angabe der ersten zehn Titel. Abb. 7.1 zeigt, wie dem Benutzer die Angabe der ersten zehn Titel anhand von Zufallstiteln präsentiert wurde.

Abschließend wurde der Teilnehmer darum gebeten, die Empfehlungslisten (Abb. 7.2), welche aus sieben Empfehlungstitel und drei Serendipitätseinträgen bestand, zu bewerten. In Abb. 7.3 sind die Fragen aufgezeigt, welche der Teilnehmer beantworten sollte. Zum einen wurde die Wertung des Teilnehmers der Empfehlungsliste abgefragt. Hier standen dem Benutzer die Antworten *schlecht*, *geht so*, *gut*, *sehr gut* zur Verfügung, welche mit einer Wertung zwischen eins bis vier gespeichert wurden. Zum anderen wurde abgefragt, wie sehr die Titel den Teilnehmer überrascht haben. Wenn

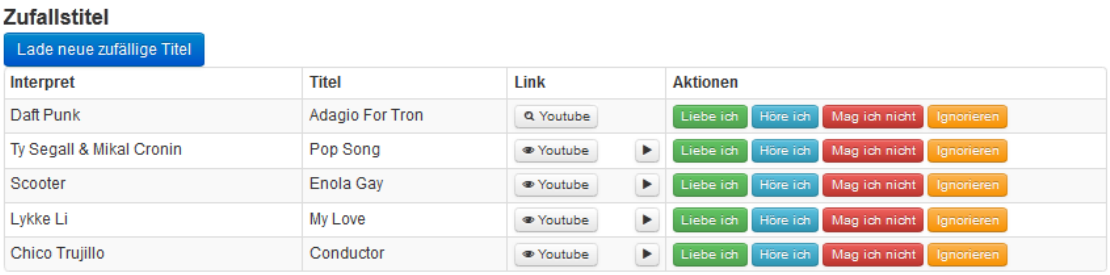


Abbildung 7.1.: Anzeige der Bewertungsmöglichkeiten mehrerer Titel

der Teilnehmer bei den Angaben seiner Vorlieben zehn Titel des selben Interpreten angab, war es wenig überraschend, wenn weitere Titel dieses Interpreten angezeigt wurden. Ebenso galt dies für Interpreten, die dem ursprünglichen Interpreten sehr ähnlich waren. Daher bezog sich die Frage auf die Serendipitätseinträge, welche in der Liste angezeigt wurden. Den Teilnehmern wurde auch hier eine Likert-Skala zur Bewertung vorgegeben. Die möglichen Antworten waren *nein, gar nicht*, *ein wenig*, *ja* und *sehr*, welche ebenso in einem Wert zwischen eins und vier abgebildet wurden.

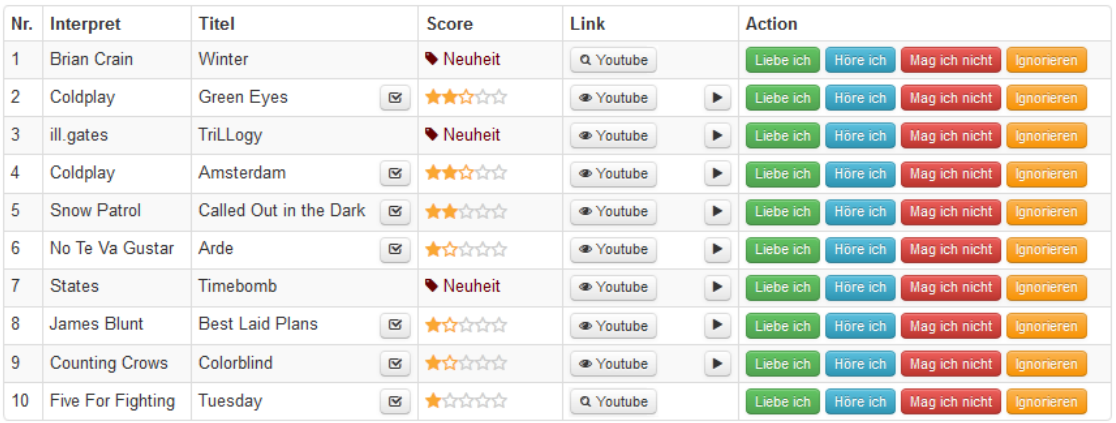


Abbildung 7.2.: Beispiel einer Empfehlungsliste

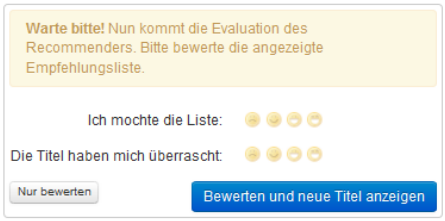


Abbildung 7.3.: Bewertung einer Empfehlungsliste

7.2. Ergebnis

Während der Evaluation von TRecs registrierten sich 144 Benutzer. Diese wurden gleichmäßig auf die vier unterschiedlichen Gruppen aufgeteilt. Beide Geschlechter wurden gleichmäßig auf alle Gruppen aufgeteilt, um zu vermeiden, dass sich in einer Gruppe nur Teilnehmer eines Geschlechts befinden. Die Teilnehmer bestanden aus 108 männlichen und 36 weiblichen Teilnehmern. So konnten jeweils 36 Teilnehmer in eine Gruppe eingeteilt werden. Die meisten Teilnehmer wurden durch die Informatik-Mailingliste gewonnen. Zur Steigerung der Teilnehmerzahl wurde die Evaluation zusätzlich in verschiedenen sozialen Netzwerken veröffentlicht. Ein Großteil der Teilnehmer war zwischen 20-29 Jahre alt (Abb. 7.4). Diese Anteile spiegelten sich auch in den einzelnen Gruppen wider.

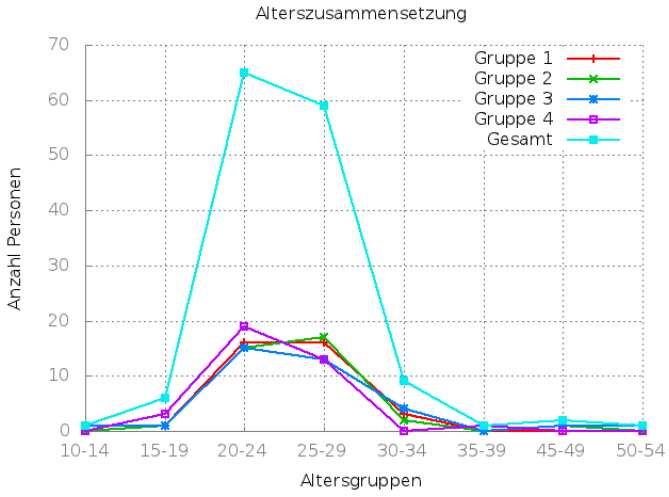


Abbildung 7.4.: Alter der Teilnehmer

Titelbewertungen

Die Liste der zehn empfohlenen Titel für Teilnehmer spaltete sich in zwei Arten von Titeln: sieben Empfehlungen und drei Serendipitätstitel. Durch die Anfrage neuer Empfehlungen wurden somit jeweils zehn neue Titel angezeigt. Von der Menge der registrierten 144 Teilnehmern wurden von 39 Teilnehmern keine Bewertung von Titeln durchgeführt, wodurch noch 105 Benutzer zur Evaluation übrig blieben. Von diesen Teilnehmern wurden insgesamt 4.386 Bewertungen von Titeln vorgenommen. Die Bewertung der Titel teilte sich in vier Bewertungen auf. Der Teilnehmer konnte jeden Titel, wie bereits beschrieben, mit *Liebe ich*, *Höre ich*, *Mag ich nicht* und *ignorieren* bewerten. Hinzu kam die Möglichkeit, Titel in einer Empfehlungsliste nicht zu bewerten. Sobald ein Teilnehmer einen Titel in der Empfehlungsliste jedoch nicht bewertet hatte, wurde er explizit darauf hingewiesen, dass er einen Titel noch nicht bewertet hatte. Der Benutzer konnte dennoch eine weitere Empfehlungsliste anfordern. Die Anzahl der verschiedenen Bewertungsmöglichkeiten sind in Abb. 7.5 dargestellt. So wurden ca. 7% (305) der angezeigten Titel nicht bewertet. Etwa 27% (1.206) der Titel wurden gebannt, sodass sie dem Teilnehmer nicht noch einmal empfohlen wurden. Weitere 17% (765) wurden von den Teilnehmern ignoriert, sodass sie zu einem späteren Zeitpunkt noch einmal angezeigt werden konnten. 17% (743) der Titel wurden von den Benutzern als favorisierte Titel markiert. Hinzu kommen weitere 31% (1.367) der Titel welche durch die Benutzer als *höre ich* markiert wurden. Sowohl die favorisierten Titel als auch die Titel, welche der Teilnehmer gehört hatte, konnten als gute Empfehlungen angesehen werden und ergaben zusammen 48% der angezeigten Titel.

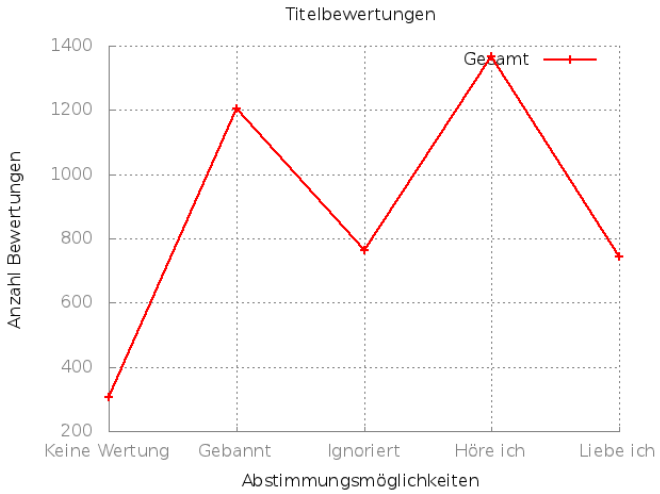


Abbildung 7.5.: Verteilung der Bewertungen aller Gruppen

Die Verteilung der einzelnen Gruppen wird in Abb. 7.6 dargestellt. Im Vergleich hierzu wird die Verteilung aller Teilnehmer in Abb. 7.5 visualisiert. Hier ist zu sehen, dass Teilnehmer der Gruppe 1 am aktivsten waren. Der Grund hierfür ist, dass in Gruppe 1 viele Teilnehmer waren, welche viele Empfehlungslisten und dadurch auch sehr viele Titel bewertet haben. Es ist zu erkennen, dass die Teilnehmer aus Gruppe 1 oft die Möglichkeit des Ignorierens gewählt haben.

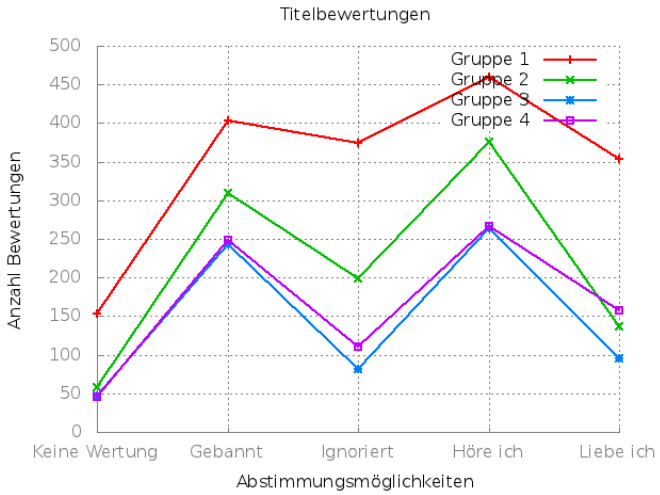


Abbildung 7.6.: Verteilung der Bewertungen pro Gruppen

Die Serendipitätseinträge wurden durch die Benutzer nicht positiv aufgenommen, was durch Abb. 7.7 gezeigt werden kann. Unter Berücksichtigung, dass in der Emp-

fehlungsliste sieben Empfehlungen und drei Serendipitätseinträge angezeigt wurden, ist zu erkennen, dass die Serendipitätseinträge deutlich seltener als geliebt oder als gehörte Titel gekennzeichnet wurden. Im Vergleich zu den Empfehlungstitel wurden diese deutlich öfter ignoriert und gebannt. Hieraus lässt sich schließen, dass die Serendipitätseinträge den meisten Teilnehmern nicht gefallen haben und die angewandte Methode sich nicht zur Selektion der Serendipitätseinträge eignete.

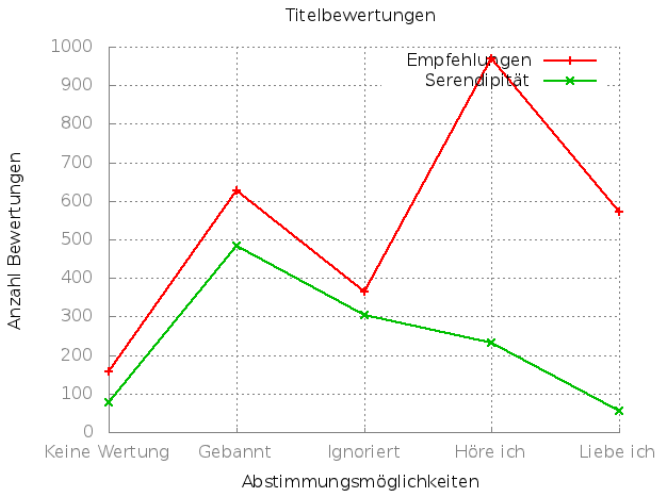


Abbildung 7.7.: Verteilung der Bewertungen aller Gruppen aufgeteilt nach Art der Empfehlungen

Einfluss der YouTube-Links

Positiv wirkte sich die Möglichkeit der YouTube-Links aus. In Abb. 7.8 ist dargestellt, wie sich die Bewertungen von Titeln mit und ohne YouTube-Links unterschieden. Erkennbar ist, dass die YouTube-Links keinen Unterschied bei nicht bewerteten Titeln machten - im Gegensatz zu den gebannten Titeln, wo ein großer Unterschied zu erkennen ist. Dies lässt sich gut im Zusammenhang der ignorierten Titel erklären: Wenn ein Titel für den Benutzer unbekannt war, so wurde dieser häufiger ignoriert. War der Titel dem Benutzer jedoch bekannt oder stand auf YouTube zum Anhören bereit, konnte direkt entschieden werden, ob der Titel gefällt oder nicht. Dadurch erklärt sich die hohe Anzahl der gebannten Titel, welche bei YouTube angehört werden konnten. Die Bewertungsmöglichkeiten der geliebten Titel und der gehörten Titel bilden zwischen den Titeln mit YouTube-Link und ohne einen deutlichen Unterschied. So wurden Titel mit YouTube-Link deutlich häufiger als geliebte Titel hinzugefügt als Titel ohne YouTube-Link. Dies kann darauf zurückgeführt werden, dass sobald der Teilnehmer auf den Link klicken konnte, er sich direkt eine Meinung

bilden oder sich besser an einen Titel erinnern konnte. Noch größer ist die Beeinflussung des YouTube-Links bei der Kategorie *Höre ich*. Hier sieht man, dass die Anzahl der gehörten Titel mit YouTube-Links ungefähr die vierfache Anzahl der Titel ohne YouTube-Links bildeten. Es zeigte sich, dass die YouTube-Links bzw. die eingebetteten Videos den Benutzer unterstützten, die Titel zu bewerten. Insbesondere bei der *Mag ich nicht*, *Liebe ich* und *Höre ich* Kategorie ist eine hohe Differenz zwischen Titeln ohne und mit YouTube-Link erkennbar.

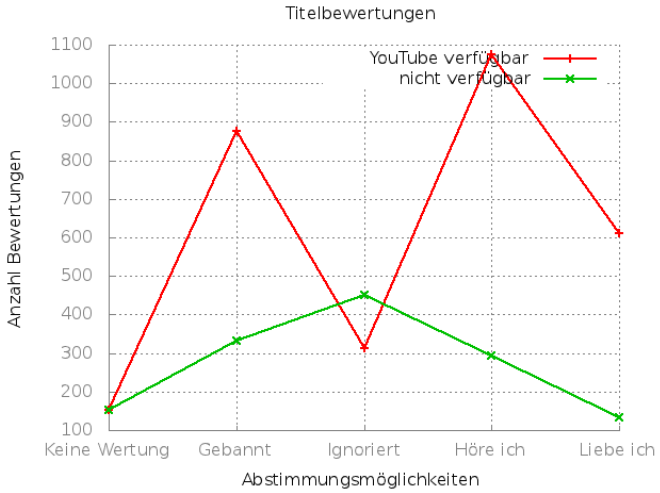


Abbildung 7.8.: Einfluss von YouTube auf Abstimmung

Insgesamt wurden in der Evaluation 439 Empfehlungslisten durch die Teilnehmer bewertet. Auch wenn die Anzahl der Listenbewertungen hoch erscheint, so liegt sie weit hinter der Anzahl der generierten Empfehlungslisten. Insgesamt wurden von allen Teilnehmern über 1.400 Empfehlungslisten generiert. Dies bedeutet, dass ca. 30% der generierten Listen bewertet wurden. Es ist zu vermuten, dass viele Benutzer nach einer Berechnung der Empfehlungen zwar die Empfehlungen betrachtet und potentiell die Titel bewertet haben, jedoch nicht die Empfehlungsliste bewertet wurde. Hierdurch wurden die bewerteten Titel zwar für die Berechnung neuer Empfehlungen berücksichtigt, jedoch nicht in die Evaluation mit aufgenommen.

Gruppenbewertungen

Bei der Evaluation von TRecs wurden vier unterschiedliche Gruppen gebildet. So konnten pro Gruppe unterschiedliche Aussagen getroffen werden. Erwartungsgemäß sollten die Empfehlungslisten für die Teilnehmer mit jeder weiteren Liste besser werden. Dies sollte sich in den Bewertungen der Listen widerspiegeln. Diese Ent-

wicklung konnte in den ausgewerteten Ergebnissen nicht eindeutig bestätigt werden, was auf die Menge der ausgewählten Titel zurückzuführen sein könnte. In Abb. 7.9 sind die Durchschnittswerte der Benutzer über die Listennummer abgebildet. Es wurden lediglich die ersten fünf Listenbewertungen berücksichtigt, da bei nachfolgenden Listen die Anzahl der Bewertungen zu gering war. Es lässt sich jedoch bei keiner der vier Gruppen eine Aussage darüber treffen, ob die Empfehlungslisten im Laufe der Zeit besser wurden. Zu vermuten ist, dass TRecs mehr Hörereignisse eines Benutzers benötigt, um diesem bessere Empfehlungen vorschlagen zu können. Da die Teilnehmerzahl und die Anzahl der bewerteten Empfehlungslisten und damit auch der bewerteten Titel pro Teilnehmer stark begrenzt waren, konnte TRecs die Vorlieben der Teilnehmer zwar berücksichtigen, jedoch stellten die bewerteten Titel nur ein Ausschnitt der Vorlieben der Teilnehmer dar. Des Weiteren ist zu vermuten, dass die Titelselektion nicht optimal gewählt wurde und hierdurch viele Titel, welche für einige Benutzer gute Ergebnisse dargestellt hätten, nicht selektiert wurden. Anstelle dessen wurden eher Titel selektiert, welche vielen Benutzern unbekannt waren und nur einem geringen Anteil der Benutzer gefiel. Hierdurch konnten für manche Teilnehmer keine optimalen Empfehlungen generiert. Bei der Betrachtung der durchschnittlichen Bewertung der Empfehlungslisten, die über alle Gruppen gebildet wurde, war jedoch eine positive Tendenz der Bewertungen erkennbar (Abb. 7.10).

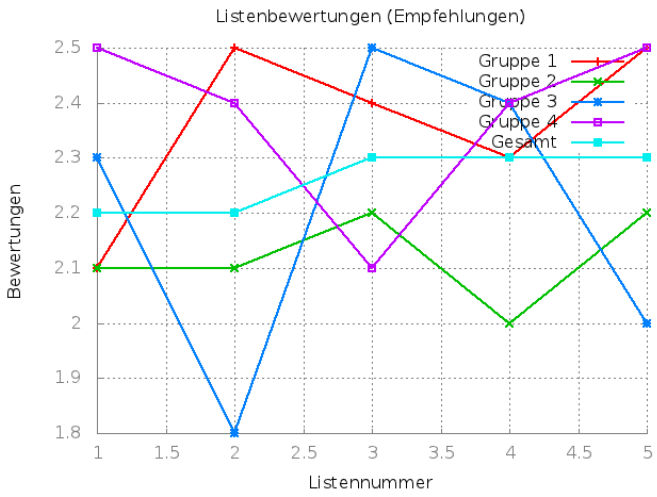


Abbildung 7.9.: Verteilung der Bewertungen pro Gruppen (Liste 1-5)

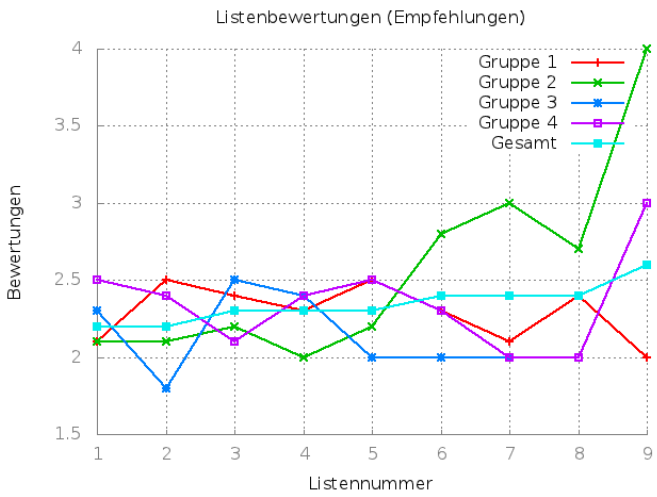


Abbildung 7.10.: Verteilung der Bewertungen pro Gruppen (Liste 1-9)

Ähnliches wie bei den Empfehlungen gilt auch für die Serendipitätseinträge. Da durch die Evaluation eine Bewertung über die Serendipitätseinträge abgefragt wurde, konnte hier eine Auswertung vorgenommen werden, inwieweit diese den Benutzer positiv überrascht haben. Die durchschnittliche Bewertung der fünften Empfehlungsliste liegt genau in der Mitte der angebotenen Bewertungsskala. Es ließ sich somit keine positive oder negative Beeinflussung der Teilnehmer erkennen.

Im Hinblick auf die Anzahl der bewerteten Empfehlungslisten lassen sich eindeutige Unterscheide zwischen den Gruppen erkennen (siehe Abb. 7.11). Es zeigte sich, dass durch Gruppe 1 die meisten Listen bewertet wurden. Da die Teilnehmer in der Einführung der Evaluation gebeten wurden, mindestens vier Empfehlungslisten zu bewerten, ist nach der vierten Liste ein deutlicher Abfall der Anzahl der Listenbewertungen erkennbar. Zu erkennen ist, dass insbesondere Gruppe 3 innerhalb der ersten vier Listen die höchste Absprungrate hatte. Dies lässt die Vermutung zu, dass, obwohl die erste Liste von 24 Teilnehmern aus Gruppe 3 bewertet wurde, die Teilnehmer mit den Empfehlungslisten nicht zufrieden waren und die Motivation deutlich sank, weitere Empfehlungslisten zu bewerten. So wurde die vierte Empfehlungsliste von nur noch zehn Teilnehmern bewertet. Alle anderen drei Gruppen hatten in den ersten vier Listen einen ähnlichen Abfall der Nutzerzahlen. So senkte sich die Anzahl der Listenbewertungen innerhalb der ersten vier Listen bei den drei Gruppen um fünf Listenbewertungen. Sowohl bei Gruppe 3 als auch Gruppe 4 ist zu erkennen, dass direkt nach der vierten abgestimmten Liste die Anzahl der Listenbewertungen deutlich absank. So wurden durch Teilnehmer der Gruppe 3 maximal

sieben Empfehlungslisten bewertet. Bei Gruppe 4 wurden maximal 12 Listen bewertet. Die Anzahl der Bewertungen von Empfehlungslisten nimmt bei Gruppe 2 linear ab. Durch Teilnehmer der Gruppe 2 wurden maximal 14 Listen bewertet. Ein deutlicher Rückgang der Listenbewertungen ist auch bei Gruppe 1 zu sehen, jedoch sinkt die Anzahl der Listenbewertungen ab der Liste 7 deutlich langsamer. Die Anzahl der bewerteten Listen pro Gruppe lässt vermuten, dass die Teilnehmer der Gruppe 1 am motiviertesten waren, mehr als die geforderten vier Listen zu bewerten. Die Motivation konnte unter anderem an der Qualität der einzelnen Empfehlungslisten liegen. Es konnte durch die unterschiedliche Aktivität der Teilnehmer der Gruppen davon ausgegangen werden, dass die Teilnehmer der Gruppe 1 und Gruppe 2 wesentlich zufriedener mit dem Ergebnis und folglich motivierter waren als die der Gruppe 3.

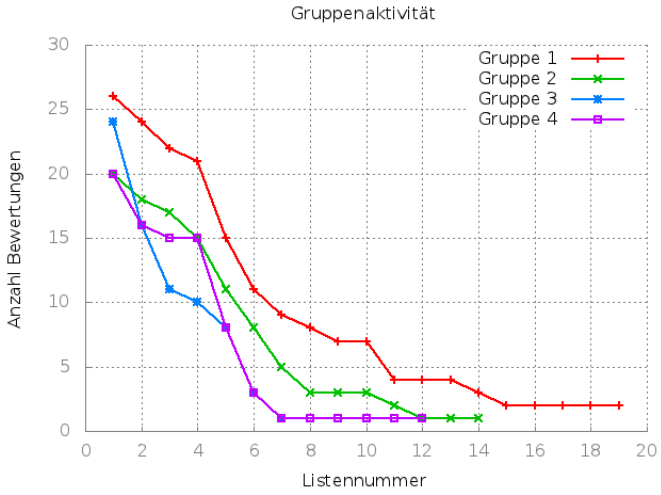


Abbildung 7.11.: Anzahl der bewerteten Listen pro Gruppe

Verweildauer

Nachdem sich Benutzer an der Benutzeroberfläche angemeldet hatten, wurde für jeden Benutzer eine Sitzung angelegt. Diese Sitzung wurde automatisch alle 60 Sekunden aktualisiert. So konnte gespeichert werden, wie lange ein Benutzer eingeloggt war. Durchschnittlich über alle Gruppen hinweg blieb ein Benutzer fast 23 Minuten angemeldet und hat in dieser Zeit mit der Oberfläche interagiert. Hierbei zählen neben der Bewertung von Titeln und der Berechnung neuer Titel auch das Ansehen und Hören der YouTube-Videos, welche auf der Seite eingebettet waren.

Die durchschnittliche Aufenthaltsdauer nach Gruppen wird in Tab. 7.4 gezeigt. Es ist zu erkennen, dass durch Teilnehmer der Gruppe 1, wie oben gezeigt, die meisten

Empfehlungslisten bewertet wurden, obwohl sie die geringste Aufenthaltszeit hatten. Eine mögliche Erklärung hierfür wäre, dass Teilnehmer der Gruppe 1 musikbegeistert waren und viele der empfohlenen Titel bereits kannten. Wenn einem Teilnehmer ein Titel bereits bekannt war, so musste dieser seltener YouTube-Links benutzen, um zu entscheiden, ob ihm der Titel gefällt oder nicht. Dies wurde durch die Anzahl der Youtube-Klicks pro Gruppe bestätigt. So wurden von Teilnehmern der Gruppe 1 lediglich bei 42% der Titel auf YouTube geklickt. Bei den übrigen Gruppen war dieser Anteil deutlich höher (57%, 60%, 59%).

	Durchschnittliche Aufenthaltszeit
Gruppe 1	1201 Sekunden
Gruppe 2	1471 Sekunden
Gruppe 3	1419 Sekunden
Gruppe 4	1457 Sekunden

Tabelle 7.4.: Aufenthaltszeit der Benutzer nach Gruppen

Durch Implementierung der Vorberechnungen konnten die Empfehlungslisten alle innerhalb der geforderten 30 Sekunden berechnet werden. Die durchschnittlich benötigte Zeit zur Berechnung der Empfehlungen eines Benutzers lag bei 12,946 Sekunden. Die längste Berechnungszeit war 18 Sekunden, wohingegen die schnellste Berechnung nur 11,244 Sekunden benötigte. Somit wurden jedem Benutzer immer die besten Empfehlungen berechnet.

Benutzerfeedback

Das Feedback-Formular wurde von mehreren Teilnehmern genutzt. Alle abgeschickten Einträge sind in Anhang C zu sehen. Oftmals wurde von den Teilnehmern angemerkt, dass die Suche zum Eintragen der zu Beginn benötigten Titel nicht die erwarteten Ergebnisse lieferte. Viele Teilnehmer hätten sich eine Funktion gewünscht, welche automatisch ähnliche Schreibweisen des Titels vorschlug (“Meinten sie ...”). Diese Funktion ist auch bei Google[1] realisiert und schlägt bei Schreibfehlern automatisch die richtige Schreibweise vor.

Neben Problemen bei der Benutzung der Oberfläche, welche nur von einzelnen Benutzer bemängelt wurden, wurde öfter gemeldet, dass viele Titel in einer unzureichenden Qualität bei YouTube vorlag oder ein Fehler bei YouTube auftrat, wie zum Beispiel eine Sperrung des Videos in Deutschland. Da die YouTube-Videos zu

[1] http://www.google.de - Zugriff: 01.09.2012

Beginn der Evaluation gesucht wurden, konnte es vorkommen, dass im Laufe der Evaluation Videos durch Rechteinhaber gesperrt wurden. Bei der Verlinkung der YouTube-Videos wurde die Qualität des Videos nicht berücksichtigt, sodass mitunter auch gleichnamige Cover-Versionen von Youtube-Nutzern verlinkt wurden.

Die ausgewählten Titel des Recommenders wurden ebenso von einigen Teilnehmern bemängelt, da ihre Vorlieben nicht durch diese Titel abgedeckt wurden. Dies kann darauf zurückgeführt werden, dass durch die Selektionsstrategie nicht jeder Musikgeschmack ausreichend abgedeckt wurde. Die Auswahl der Musiktitel hat in dieser Arbeit die gesamte Evaluation beeinflusst, da die Empfehlungen maßgebend von den Titeln abhängig waren. Es wurden zwar viele Titel ausgewählt, die von den Benutzern positiv bewertet wurden und bereits bekannt waren, jedoch wurden durch die angewandte Selektionsstrategie auch viele unbekannte Titel ausgewählt, die nur von sehr wenigen Benutzern gehört wurden.

Mit Hilfe der Feedbackfunktion wurden auch Kommentare zu den Bewertungsmöglichkeiten geschrieben. So reichten einigen Besuchern die Bewertungsmöglichkeiten nicht aus und wünschten sie sich zusätzlich eine feinere Aufteilung der Bewertungen. Als Beispiel wünschten sich einige Teilnehmer eine *nicht toll, aber auch nicht schlecht*-Bewertungsmöglichkeit. Mit der in der Evaluation gewählten Bewertungsmöglichkeiten wurde der Benutzer dazu gezwungen, sich klar für oder gegen den Titel zu entscheiden. Durch das Hinzufügen des Ignorierens konnte der Benutzer Titel, die er im Moment nicht bewerten wollte, für eine spätere Bewertung markieren. Die Anzahl der Bewertungsmöglichkeiten wurde gewählt, um den Benutzer bei der Abstimmung nicht durch zu viele unterschiedliche Möglichkeiten zu verwirren.

7.3. Zusammenfassung

Obwohl die Evaluation von TRecs nicht eindeutig zeigen konnte, dass eine bestimmte Gruppe die Zufriedenheit der Teilnehmer über die ersten Empfehlungslisten steigerte, konnte gezeigt werden, dass im Durchschnitt über alle Gruppen eine Verbesserung der Bewertungen feststellbar war. Obwohl versucht wurde, so viele Personen wie möglich zu motivieren, an der Evaluation teilzunehmen, waren nur 105 Teilnehmer aktiv und haben mehrere Empfehlungslisten bewertet. Da die Teilnahme an der Evaluation freiwillig war, konnten die Teilnehmer jederzeit abbrechen und keine weiteren Empfehlungslisten bewerten, was dazu führte, dass viele Benutzer bereits

nach den ersten Listen abbrachen. Für eine detailliertere Aussage, ob TRecs für Teilnehmer über die Zeit bessere Empfehlungen generiert, hätten mehr Benutzer an der Evaluation teilnehmen müssen. Ebenso wäre es wichtig gewesen, dass die Benutzer deutlich mehr Titel bewerteten. Eine größere Auswahl an Titeln hätte zudem bessere Ergebnisse erzielen können. Durch die Anzahl der Teilnehmer konnte hier jedoch nur eine Tendenz gemittelt über alle Gruppen gefunden werden, welche bestätigte, dass sich die Empfehlungen über die Zeit verbesserten. Welche Gewichtung, die in den Gruppen benutzt wurde, die besten Empfehlungen berechnen konnte, kann nur anhand des Aktivitätsgrades bestimmt werden. Da alle Teilnehmer gleichmäßig über alle Gruppen aufgeteilt wurden und die Teilnehmer der Gruppe 1 am aktivsten waren, lässt sich vermuten, dass die berechneten Empfehlungen der Gruppe 1 die Teilnehmer am besten dazu motivierten, eine weitere Liste zu bewerten.

In Gruppe 1 nahm der Track-Recommender den größten Einfluss auf die Empfehlungen. Hierdurch lässt sich vermuten, dass dieser Teil-Recommender ein Schlüsselelement für gute Empfehlungen in diesem Szenario darstellte. Der Tag-Recommender und Time-Recommender wurden beide gleich gewichtet und gemeinsam zu 20% berücksichtigt. Als Empfehlung für eine Gewichtung der Teil-Recommender sollte der Track-Recommender möglichst schwer gewichtet werden, wohingegen sowohl der Tag-Recommender als auch der Time-Recommender eine untergeordnete Rolle einnehmen und daher leichter gewichtet werden sollten.

Über das angewandte Verfahren zur Bestimmung von Serendipitätseinträge kann keine Aussage getroffen werden, da die Ergebnisse über alle Gruppen hinweg sehr unterschiedlich waren. Auch hier wäre es wünschenswert gewesen mehr Teilnehmer gehabt zu haben, um eine Tendenz erkennen zu können.

8. Ausblick

Ziel dieser Arbeit war es, Benutzern neue Empfehlungen aufgrund ihres Hörverhaltens zu berechnen. Hierzu wurden die Benutzer zu Beginn der Evaluation aufgefordert, ihre Vorlieben mittels zehn Titel zu definieren. Diese Anzahl wurde gewählt, um die Zeit zwischen der ersten Angabe eines Titels und den ersten Empfehlungen so gering wie möglich zu halten. Die Ergebnisse zeigten jedoch, dass die Angabe von nur zehn Titeln nicht ausreichte, um die Vorlieben eines Benutzers hinreichend festzulegen. Denn gerade bei der Musik-Domäne können die Vorlieben eines Benutzers sehr unterschiedlich und breit gefächert sein.

Weiterhin muss die Qualität der Ergebnisse des Recommenders mit den zur Verfügung stehenden Musiktiteln in Verbindung gesetzt werden. Von dem in Kapitel 4 beschriebenen Crawler wurden ca. 4,7 Millionen Titel geladen. Nach der Kontrolle auf doppelte Einträge in Kapitel 5 blieben hiervon etwa 3,6 Millionen Titel übrig. Wenn die Menge der zur Verfügung stehenden Titel reduziert wird, muss ein Kompromiss zwischen bekannten und eher unbekannten Musiktiteln gefunden werden. Für die Empfehlungen sollten zum einen Titel enthalten sein, die von einem Großteil der Benutzer gehört werden, wie zum Beispiel Titel aus den Charts, welche nach der Spielhäufigkeit im (öffentlichen) Radio und den Verkäufen generiert werden. Zum anderen sollten auch unbekanntere Interpreten mit eingeschlossen werden, um den Benutzern neue Titel vorzustellen. Die in dieser Arbeit genutzte Selektionsstrategie berücksichtigte beide Aspekte, indem sie zum einen Titel mit Hilfe von Benutzern ausgewählt hat - somit eine ähnliche Strategie, wie bei der Generierung der Charts - und zum anderen unbekanntere Titel über die Tags auswählte. Es stellte sich jedoch heraus, dass die Anzahl der unbekannten Titel sehr hoch war und die unbekannten Titel häufig in der Evaluation empfohlen wurden. In dem durch die Selektionsstrategie entstandenen Datensatz konnte nicht überprüft werden, ob Titel, welche die Evaluation positiv beeinflusst hätten, nicht ausgewählt wurden. In zukünftigen Arbeiten sollte die gewählte Selektionsstrategie daher verbessert oder auf eine andere Strategie zurückgegriffen werden.

Für Recommender, welche mit Testdaten arbeiten, ist die Bildung einer Auswahl der zur Verfügung stehenden Objekte eine Möglichkeit, Empfehlungen in kurzer Zeit zu berechnen. Hierbei sollte jedoch darauf geachtet werden, dass charakteristische Merkmale erhalten bleiben. In einem Produktivsystem ist dies jedoch keine Möglichkeit, da dem Benutzer eventuell Objekte verborgen bleiben, die für ihn eine optimale Empfehlung darstellen könnten. Wenn man den Ansatz dieser Arbeit mit 15.000 Titeln auf ein Szenario mit ca. 3,6 Millionen Titeln überträgt, wäre ein einzelner Rechner, wie er in dieser Arbeit verwendet wurde, nicht in der Lage, diese Berechnungen in angemessener Zeit durchzuführen. Aus diesem Grund sollten, bei der Verwendung großer Datensätze verteilte Rechensysteme, wie z.B. Map Reduce, eingesetzt werden. Dabei greifen mehrere Rechner auf eine gemeinsame Basis zu und können die Berechnungen parallel auf mehreren Rechner ausführen. Diesen Ansatz verfolgt Apache Hadoop[1], welches ein Framework für nebenläufige Berechnungen für große Datenmengen bereitstellt. Für die Berechnung von Empfehlungen in sehr großen Datenmengen stellt Apache Mahout[2] ein geeignetes Werkzeug dar, welches unter anderem auch Hadoop nutzen kann, um die Berechnung von Empfehlungen zu beschleunigen. Durch die Nutzung von verteilten Rechensystemen kann die Geschwindigkeit der Berechnungen in zukünftigen Arbeiten deutlich gesteigert werden.

Der Time-Recommender berechnete Empfehlungen anhand der Hörhäufigkeit an den unterschiedlichen Tagen im Jahr. Die Methode, mit welcher Ähnlichkeiten zwischen Titeln berechnet wurden, lieferte eine hohe Ähnlichkeit bei Titeln, die zum Beispiel auf der selben CD waren. Bei diesen Titeln stellte der Teil-Recommender eine hohe Ähnlichkeit fest, da die Titel meist in einer gewissen Reihenfolge abgespielt wurden und somit gleich oft gehört wurden. Die Ähnlichkeit von zwei Titeln von unterschiedlichen Interpreten auf unterschiedlichen CDs wurde jedoch meist als sehr gering berechnet. Eine mögliche Lösung stellt hier die Verwendung zusätzlicher Informationen dar. So könnte der Einfluss der Abspielreihenfolge von CDs und Hörhäufigkeit von Titeln auf einen Tag bezogen gemindert werden, indem zusätzlich die Anzahl der Hörereignisse pro Woche und Monat berücksichtigt werden. So könnte eine hohe Ähnlichkeit zwischen zwei Titeln festgestellt werden, die zwar nicht am selben Tag gehört werden, jedoch beide sehr oft im gleichen Monat gehört werden. Zusätzlich wäre eine Negativ-Wertung denkbar, um Titel des gleichen Interpreten oder der gleichen CD weniger stark zu gewichten.

[1] http://hadoop.apache.org/ - Zugriff: 01.09.2012
[2] http://mahout.apache.org/ - Zugriff: 01.09.2012

Die Eingabe der zu Beginn benötigten Titel wurde in der Evaluation durch die Benutzer getätigt. Der Benutzer musste an dieser Stelle relativ spontan zehn Titel nennen, welche seine Vorlieben darstellten. Insbesondere bei der Suche der zehn Titel traten immer wieder Tippfehler der Benutzer oder abweichende Schreibweisen auf. Für eine zukünftige Evaluation sollte eine "Meinten sie"-Methode zur Verfügung gestellt werden, welche die Eingabe des Benutzers überprüft und bei Schreibfehlern ähnliche Titel anzeigt. Zur Reduzierung der Benutzerinteraktion zu Beginn wäre es auch interessant, die Vorlieben des Benutzers anhand eines eventuell angelegten Last.fm-Benutzerprofils zu erkennen. Eine ähnliche Methode könnte auch mit einem YouTube-Konto verfolgt werden, bei dem die zuletzt angesehenen Videos analysiert werden und Kenntnisse über gehörte Musiktitel erworben werden. Dies hätte zur Folge, dass der Recommender einem Benutzer direkt nach der Eingabe seines Benutzernamens eines unterstützten Musikdienstes Empfehlungen präsentieren könnte. Dadurch müsste der Benutzer keine zusätzliche Zeit aufwenden, um seine Vorlieben einzugeben und wäre motivierter die Empfehlungen zu bewerten.

In der Evaluation (Kapitel 7) konnte gezeigt werden, dass die gewählte Methode, um Serendipitätseinträge zu bestimmen, keine eindeutigen Folgerungen zuließen. In weiterführenden Arbeiten sollte daher die erste beschriebene Methode zur Serendipitätsauswahl von Kap. 6.4 verwendet werden. Da in dieser Arbeit der Einsatz eines Recommenders in Echtzeitsystemen auf einem Arbeitsplatzrechner getestet wurde, konnte die erste Methode nicht verwendet werden, da die nötige Berechnungszeit das festgelegte Maximum überschritten hätte. In zukünftigen Arbeiten kann die erste Methode unter Verwendung von verteilten Rechensystemen angewandt werden, um die nötige Berechnungszeit auszugleichen.

Die unterschiedlichen Bewertungsmöglichkeiten der Evaluation wurden so gewählt, dass der Benutzer sich klar für oder gegen einen Titel entscheiden musste, um diesen zu bewerten. Er hatte außerdem die Möglichkeit, einen Titel in der Evaluation zu ignorieren, sodass dieser Titel erst zu einem späteren Zeitpunkt erneut angezeigt wurde. Die gewählte Skala, die den Benutzern zur Verfügung stand, wurde auf sehr einfache Aktionen ausgelegt. Da die Skala der Bewertungsmöglichkeiten ohne weiteres ausgetauscht werden könnte, stellt sich für zukünftige Arbeiten die Frage, ob eine feinere Abbildung der Vorlieben, wie zum Beispiel eine Skala von null bis zehn, bessere Empfehlungen für den Benutzer generieren würden oder diese den Benutzer eher verwirren würden.

Zur Verbesserung der Evaluationsmöglichkeiten ist denkbar, dass die in dieser Arbeit entwickelte Oberfläche als Grundlage für kommende Recommender Systeme genutzt werden kann. Durch eine Trennung zwischen Oberfläche und dem Recommender wäre es möglich, mit Hilfe einer Oberfläche mehrere von unterschiedlichen Personen programmierte Recommender zu evaluieren. Für eine solche Oberfläche müsste in weiteren Arbeiten eine API definiert werden, welche sowohl von den Recommendern als auch von der Oberfläche eingesetzt werden würde. Wie bereits in dieser Arbeit verwendet, empfiehlt es sich, in zukünftigen Evaluationen eine direkte Kommunikation der Oberfläche und des Recommenders zu nutzen.

9. Zusammenfassung

Die zunehmende Verfügbarkeit von Musik-Streaming-Diensten in den letzten Jahren bietet neue Anreize sowohl für die Betreiber als auch für die Nutzer. So können die Betreiber die Interessen ihrer Hörer feingranularer analysieren, womit sich personalisierte Dienste realisieren lassen. Eine weitverbreitete Musikplattform in Deutschland ist Last.fm, welche es sowohl erlaubt, Musik über das Internet zu hören als auch die lokal abgespielten Titel an Last.fm zu melden (sogenanntes Scrobbeln). Viele dieser Daten sind bei Last.fm öffentlich über eine Schnittstelle abrufbar und bilden die Grundlage für den in dieser Arbeit entwickelten hybriden Recommender Track Recommender System (TRecs).

TRecs basiert auf einer Kombination von drei verschiedenen konfigurierbaren Metriken: der Ähnlichkeit von Titeln basierend auf gemeinsamem Hörverhalten verschiedener Benutzer, ähnlichen Tags, mit denen diese Titel annotiert sind, und dem zeitlichen Abspielverhalten der Titel. Die Berechnung der Empfehlungen wurde aufgrund der aufwändigen Laufzeit der verwendeten Algorithmen in zwei verschiedene Phasen aufgeteilt: eine Vorberechnungsphase, in der die Ähnlichkeiten für die drei Metriken zwischen den verschiedenen Titeln bestimmt wurden, und eine Online-Phase, die Empfehlungen in Echtzeit auf einem Einzelplatzsystem (unter 30 Sekunden) berechnete. Trotz der Einführung einer separaten Vorberechnungsphase war es erforderlich, die Menge der für die Evaluation betrachteten Musiktitel von über 4,5 Millionen gesammelten Titeln auf 15.000 Titel zu reduzieren. Die Datenselektion dieser Untermenge wurde aufgrund einer Analyse des Ursprungsdatenbestandes durchgeführt. Dabei wurde die Selektion derart durchgeführt, dass sowohl eine Teilmenge der bekanntesten Titel als auch exotischere Titel ausgewählt wurden.

Die Evaluation von TRecs erfolgte über eine Web-Oberfläche, die Benutzern nach initialer Angabe von zehn bewerteten Titeln neue Titel-Empfehlungen lieferte. Bewertet wurde anschließend die Qualität der einzelnen Empfehlungen und die der gesamten Empfehlungsliste. Dabei berücksichtigte das System die bisher vom Benutzer

gemachten Bewertungen für die Berechnung neuer Empfehlungslisten. Zu jeder personalisierten Empfehlung konnte sich der Benutzer eine detaillierte Aufschlüsselung der einzelnen Beiträge der Recommender-Metriken zum Gesamtergebnis anzeigen lassen. Für die Evaluation wurden vier Gruppen mit jeweils unterschiedlichen Gewichtungen der einzelnen Metriken definiert. An der Evaluation nahmen über 140 Teilnehmer teil, die ca. 4.400 Titel bewerteten. Es zeigte sich, dass die von den Benutzern empfundene Qualität der Empfehlungen mit steigender Anzahl an abgegebenen Bewertungen leicht anstieg. Insbesondere die Gruppe 1, bei der die Ähnlichkeit von Titeln basierend auf gemeinsamem Hörverhalten besonders stark gewichtet wurde, führte zu besseren Ergebnissen. Jedoch ließ sich in der Evaluation in den einzelnen Gruppen kein klarer Trend erkennen, dass sich durch die steigende Anzahl der Bewertungen die empfundene Qualität der Empfehlungen signifikant verbesserte. Ein möglicher Grund hierfür könnte die im Verhältnis zur verfügbaren Anzahl an Musiktiteln kleine Auswahl der für die Evaluation verwendeten Titel sein.

In weiterführenden Arbeiten stellt die Untersuchung von Recommender-Algorithmen auf dem gesamten Musiktitelbestand ein möglicher Ansatzpunkt dar, um die Qualität der Empfehlungen weiter zu steigern. Hierfür bietet sich insbesondere das MapReduce-Framework[1] an, für das es beispielsweise mit Mahout[2] erste Bibliotheken mit entsprechenden Algorithmen gibt.

[1] http://hadoop.apache.org/mapreduce/ - Zugriff: 01.09.2012
[2] http://mahout.apache.org/ - Zugriff: 01.09.2012

A. Verwendete Datenbanktabellen

track: Informationen über Titel

trackID:	Eindeutige ID des Titels
name:	Titelname des Titels
url:	URL des Titels von Last.fm
artistID:	ID des Interpreten

artist: Informationen über Interpreten

artistID:	Eindeutige ID des Interpreten
name:	Interpretenname
url:	URL des Titels von Last.fm

tag: Informationen über Tags

tagID:	Eindeutige ID des Tags
name:	Tagname
url:	URL des Tags von Last.fm

trackTags: Verbindungen zwischen Titeln und Tags

trackTagID:	ID des Verbindung zwischen Titel und Tag
trackID:	ID des Titels
weight:	Gewicht der Verbindung

user: Informationen über Benutzer

userID:	Eindeutige ID des Benutzers
name:	Benutzername
url:	URL des Benutzers von Last.fm

recentTracks: Hörereignisse der Benutzer

recentTrackID:	Eindeutige ID des Hörereignisses
trackID:	ID des Titels
userID:	ID des Benutzers
timestamp:	Zeitpunkt (im Format Unix-Timestamp)

bannedTracks: Gebannte Titel der Benutzer

bannedTrackID:	Eindeutige ID des Ereignisses
trackID:	ID des Titels
userID:	ID des Benutzers
timestamp:	Zeitpunkt (im Format Unix-Timestamp)

lovedTracks: Favorisierte Titel der Benutzer

lovedTrackID:	Eindeutige ID des Ereignisses
trackID:	ID des Titels
userID:	ID des Benutzers
timestamp:	Zeitpunkt (im Format Unix-Timestamp)

userUsers: Freundschaften zwischen Benutzern

userUserID:	Eindeutige ID der Verbindung
userID1:	ID eines Benutzers
userID2:	ID eines Benutzers

B. Einführung

Willkommen

Was gibt's zu tun?

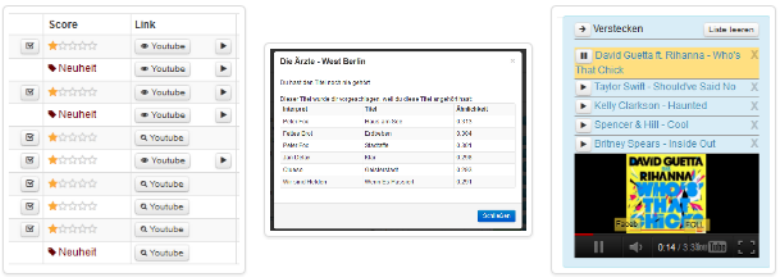
1. Zuerst musst du zehn Titel auswählen. Entweder kannst du direkt nach Titeln suchen oder du kannst aus einer zufällig angezeigten Liste Titel auswählen.

Wenn du einen Titel gefunden hast, kannst du ihn bewerten. Hierzu stehen folgende vier Möglichkeiten zur Auswahl:

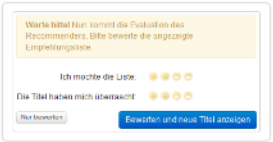
- Liebe ich - Titel, die du sehr gerne hörst.
- Höre ich - Titel, die du aktuell hörst.
- Mag ich nicht - Titel, die du hasst bzw. gar nicht magst.
- Ignorieren - Titel, die du gerade nicht bewerten möchtest. Diese werden für die nächsten zehn Empfehlungslisten nicht angezeigt.

2. Sobald du zehn Titel bewertet hast, kannst du Empfehlungen für dich generieren lassen.

Nach einem Klick bekommst du die Empfehlungen angezeigt. Um deinen Musikgeschmack zu verfeinern, musst du auch die Titel, die dir vorgeschlagen wurden, bewerten. Wenn du dich wunderst, wieso dir ein Titel empfohlen wurde, kannst du das Erklärungs-Feature benutzen. Es werden pro Durchgang zusätzlich Neuheiten angezeigt, die dir gefallen und deinen Musikgeschmack erweitern könnten.



3. Sobald du das Gefühl hast, dass du alle für dich relevanten Titel bewertet hast, kannst du dir neue Empfehlungen generieren lassen. Jedoch musst du zuvor noch bewerten, wie gut die empfohlenen Titel zu deinem Musikgeschmack gepasst haben und ob dich die Auswahl der vorgeschlagenen Titel überrascht hat.



4. Umso mehr Titel du bewertest, desto besser werden die Vorschläge, die dir das System geben kann.
Du kannst jederzeit zurück zu "Titel hinzufügen" und deinen Musikgeschmack genauer festlegen. Aber vergiss nicht, dass du dir hin und wieder neue Empfehlungen geben lässt und diese Empfehlungslisten auch bewertest. Du kannst so viele Empfehlunglisten generieren und bewerten, wie du willst. Du kannst dich jeder Zeit abmelden und zu einem anderen Zeitpunkt weitermachen. Diese Evaluation endet am 10. Juni 2012.
Für eine möglichst genaue Datenauswertung ist es notwendig, dass du mindestens vier Listen bewertest. Umso mehr Listen du bewertest, umso detaillierter kann ich die Daten auswerten. Vielen Dank!

C. Evaluationsfeedback

Teilnehmer 97555	Das man nicht nur nach Titel suchen kann sondern immer zumindest einen Teil des Interpreten eingeben muss ist sehr nervig... bei vielen Liedern kenne ich den Titel aber nicht den Interpreten
Unbekannter Teilnehmer	Uebersetze es auf English und dann kann ich es auch mit meinen Freunden sharen!
Teilnehmer 97569	Hey Simon,ist eine gute Idee! Sitze öffters mal vor Youtube und suche Lieder die mir gefallen. Aber deren Vorschläge sind nicht immer mein Fall. War überrascht, wie mir die Liste beim zweiten mal gefallen hat!Gibt es eine “meinten Sie vielleicht:...” Option auf der Seite?Well Done!!!
Unbekannter Teilnehmer	Ich scheitere schon an der Zufallsliste. Er zeigt mir keine 10 Lieder an, wenn ich einen Titel angeklickt habe und ihn bewertet habe, komme ich nicht weiter. Meine Geduld ist, was ein “Nicht-Gehorchen-Bei-Computern” betrifft, äusserst gering. Habe daher abgebrochen.
Teilnehmer 97585	- Leider zu viele Titel unbekannt (vor allem ältere Stücke)- Vorschläge wurden mit zunehmender Anzahl der Bewertungen vorwiegend schlechter (?!) - Suche mit leerem Interpreten-Feld liefert zwar eine Fehlermeldung, hängt sich dann aber auf. - Empfehlungsliste sollte bis zum nächsten Neugenerieren gespeichert bleiben (einmal hat sich die Seite beim Bewerten aufgehängt und nach F5 war die ganze Liste weg ...)
Unbekannter Teilnehmer	Ein “Geht so”-Knopf wäre noch nett (Titel, die man an sich nicht schlecht findet, aber gerade nicht hören würde)

Unbekannter Teilnehmer	Gute Arbeit - der Anfang war schwierig, weil die Track-Liste zwar lang, aber bei weitem nicht vollständig ist :)
Unbekannter Teilnehmer	Leider kamen zu viele Lieder von Künstlern, bei denen ich angegeben habe, dass ich es liebe, auch genau von denselben Alben. Diese Lieder kenne ich natürlich dann schon alle.LG
Teilnehmer 97625	Leider waren viele Interpreten, die ich höre gar nicht in deiner Datenbank. So wurde das Ergebnis beeinflusst, da ich sehr wenig als “höre ich viel” angegeben habe.
Teilnehmer 97578	geht nicht
Teilnehmer 97574	Viele lieder gehen nicht auf youtube tritt ein fehler auf
Unbekannter Teilnehmer	Login scheint nicht zu funktionieren. Jedenfalls habe ich mich heute Nachmittag registriert und wollte jetzt Eingaben machen. Es kommt - im Gegensatz zu einer anderen Mailadresse - kein Hinweis, dass ich nicht Registriert sei, sondern es passiert einfach nichts.
Teilnehmer 97653	- es wird einem nicht angezeigt, wie viele Listen man schon bewertet hat. - etwas zwischen ich höre es und Mag ich nicht wäre nicht schlecht gewesen - die Erklärungen haben öfter nicht funktioniert- Muse [The Resistance] hatte sich irgendwie eingebrannt und kam bei jeder Erklärung- wenn man die Songs bei Youtube suchen muss findest es oft nicht das richtige teilweise wird auch nicht Titel und Interpret eingegeben. Daher ist es schwer nachzuvollziehen ob man den Titel mag oder nicht da man nicht weiß ob es sich um den selben handelt - und der gain für den Endbenutzer sollte noch mehr herausgestellt werden

Teilnehmer 97653	eins noch ich konnte leider am Anfang nicht alle Titel die ich wollte hinzufügen, weil diese nicht gefunden wurden es handelte sich um gängige Titel
Unbekannter Teilnehmer	Erklär-Button funktioniert nicht :((getestet unter: Firefox, Chrome)
Teilnehmer 97659	– Die Kategorien “Liebe ich”, “Höre ich”,... sind nicht so differenziert. Mir hat ein “nicht toll, aber auch nicht schlecht”-Feld gefehlt (habe stattdessen immer ignorieren gewählt)... Und ich habe nun “Liebe ich” gedrückt, wenn es mir gefallen hat. Allerdings finde ich, dass man meistens nach einem ersten Hören eines neuen Liedes eigentlich meistens noch nicht sagen kann, wie gut es einem gefällt, daher die Wahl des Namens für diese Kategorie für mich etwas seltsam. – einige Videos haben leider so schlechte Qualität bzw. sind teilweise unter dem Titel bei Youtube nicht direkt auffindbar, dass man sie nicht bewerten kann -> von mir “ignoriert” Fazit daraus: für dich ist es nicht so durchschaubar, aus welchem Grund man “Ignorieren” wählt, was jedoch auch interessant wäre!??!?!Allgemein eine sehr interessante Idee und ich fand es toll, dass mit der Zeit die Vorschläge immer besser wurden, dein System scheint also zu funktionieren;) Ich habe mir übrigens gleich ein paar der Vorschläge gemerkt - es hat sich also gelohnt=)

Literaturverzeichnis

[1] G. Linden, B. Smith, and J. York, "Amazon. com recommendations: Item-to-item collaborative filtering," *Internet Computing, IEEE*, vol. 7, no. 1, pp. 76–80, 2003.

[2] M. Piotte and M. Chabbert, "The pragmatic theory solution to the netflix grand prize," *Netflix prize documentation*, 2009.

[3] R. Bell, Y. Koren, and C. Volinsky, "The bellkor 2008 solution to the netflix prize," *Statistics Research Department at AT&T Research*, 2008.

[4] R. Likert, "A technique for the measurement of attitudes." *Archives of psychology*, 1932.

[5] B. Drinkwater, "A comparison of the direction-of-perception technique with the likert method in the measurement of attitudes," *The journal of social psychology*, vol. 67, no. 2, pp. 189–196, 1965.

[6] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 17, no. 6, pp. 734–749, 2005.

[7] G. Kauntz, "Social networking for shared music collections."

[8] D. Turnbull, L. Barrington, and G. Lanckriet, "Five approaches to collecting tags for music," in *Proceedings of the 9th International Conference on Music Information Retrieval*, 2008, pp. 225–230.

[9] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl, "Item-based collaborative filtering recommendation algorithms," in *Proceedings of the 10th international conference on World Wide Web*. ACM, 2001, pp. 285–295.

[10] D. Turnbull, L. Barrington, D. Torres, and G. Lanckriet, "Semantic annotation and retrieval of music and sound effects," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 16, no. 2, pp. 467–476, 2008.

[11] E. Coviello, R. Miotto, and G. Lanckriet, "Combining content-based auto-taggers with decision-fusion," 2011.

[12] M. Tiemann and S. Pauws, "Towards ensemble learning for hybrid music recommendation," in *Proceedings of the 2007 ACM conference on Recommender systems*. ACM, 2007, pp. 177–178.

[13] J. Herlocker, J. Konstan, and J. Riedl, "Explaining collaborative filtering recommendations," in *Proceedings of the 2000 ACM conference on Computer supported cooperative work*. ACM, 2000, pp. 241–250.

[14] R. Burke, "Hybrid recommender systems: Survey and experiments," *User modeling and user-adapted interaction*, vol. 12, no. 4, pp. 331–370, 2002.

[15] ——, "Hybrid web recommender systems," in *The adaptive web*. Springer-Verlag, 2007, pp. 377–408.

[16] W. Cohen, P. Ravikumar, and S. Fienberg, "A comparison of string distance metrics for name-matching tasks," in *Proceedings of the IJCAI-2003 Workshop on Information Integration on the Web (IIWeb-03)*, 2003, pp. 73–78.

[17] M. Jaro, "Probabilistic linkage of large public health data files," *Statistics in medicine*, vol. 14, no. 5-7, pp. 491–498, 1995.

[18] D. Kleber, D. Maldonado, D. Scheuregger, and D. Ziprik, "Aufbau des anschriften-und gebäuderegisters für den zensus 2011," *Wirtschaft und Statistik*, vol. 7, no. 2009, pp. 629–640, 2009.

[19] J. Garrett *et al.*, "Ajax: A new approach to web applications," 2005.

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